Applying Computational Intelligence to a Real-World Container Loading Problem in a Warehouse Environment

Ayodeji Remi-Omosowon

School of Science & Technology

Nottingham Trent University

A thesis submitted in partial fulfilment of the requirements of Nottingham Trent University for the degree of

Doctor of Philosophy

September 2017

To my loving family.

Acknowledgements

I would like to acknowledge my supervisors, Dr Richard Cant and Dr Caroline Langensiepen, for their support in both academic and life matters. Their continuous guidance was a crucial factor in my development as a scientist, and indeed in the completion of this thesis. I would also like to thank my examiners, Dr Colin Wilmott and Professor Robert John, for their valuable comments.

I acknowledge Professor Lars Nolle for introducing me to Combinatorial Optimisation and its real-world applications; and for suggesting the pursuit of a research degree in this area. I also acknowledge Dr Taha Osman for the valuable comments and insights provided during the course of this research. Given its real-world application, his suggestions provided a constant reminder of the importance of a thoroughly scientific approach, with a focus on academic contributions, over and above the related software engineering contributions of the research. He is also responsible for my interest in the Semantic Web.

I acknowledge with love my family and friends who have been very supportive and helpful throughout this challenging journey. They helped make it pleasant and enjoyable. I particularly thank my wife for her patience, and my father, the other academic in the family, for his advice and encouragement.

Funding for this project has been provided by the Logistics Department in NSK Europe Ltd. I recognise the very significant roles played by NSK's United Kingdom Distribution Centre team, my manager: Mr Neil Dodd, and his manager: Mr Mark Carter, and I gratefully acknowledge with thanks all the support that I have received.

Abstract

One of the problems presented in the day-to-day running of a warehouse is that of optimally selecting and loading groups of heavy rectangular palletised goods into larger rectangular containers while satisfying a number of practical constraints. The research presented in this thesis was commissioned by the logistics department in NSK Europe Ltd, for the purpose of providing feasible solutions to this problem. The problem is a version of the Container Loading Problem in the literature, and it is an active research area with many practical applications in industry. Most of the advances made in this area focus more on the optimisation of container utility i.e. volume or weight capacity, with very few focusing on the practical feasibility of the loading layout or pattern produced. Much of the work done also addresses only a few practical constraints at a time, leaving out a number of constraints that are of importance in real-world container loading. As this problem is well known to be a combinatorial NP-hard problem, the exact mathematical methods that exist for solving it are computationally feasible for only problem instances with small sizes. For these reasons, this thesis investigates the use of computational intelligence techniques for solving and providing near-optimum solutions to this problem while simultaneously satisfying a number of practical constraints that must be considered for the solutions provided to be feasible. In proposing a solution to this problem and dealing with all the constraints considered, an algorithmic framework that decomposes the CLPs into sub-problems is presented. Each sub-problem is solved using an appropriate algorithm, and a combination of constraints particular to each problem is satisfied. The resulting hybrid algorithm solves the entire problem as a whole and satisfies

all the considered constraints. In order to identify and select feasible container layouts that are practical and easy to load, a measure of disorder, based on the concept of entropy in physics and information theory, is derived. Finally, a novel method of directing a Monte-Carlo tree search process using the derived entropy measure is employed, to generate loading layouts that are comparable to those produced by expert human loaders. In summary, this thesis presents a new approach for dealing with real-world container loading in a warehouse environment, particularly in instances where layout complexity is of major importance; such as the loading of heavy palletised goods using forklift trucks. The approach can be used to deal with a number of relevant practical constraints that need to be satisfied simultaneously, including those encountered when the heavy goods are arranged and physically packed into a container using forklift trucks.

Contents

List of Figures

List of Tables

List of Algorithms

Publications, Posters, Presentations

The following have been produced as a direct result of the research presented in this thesis.

Journal Articles (in preparation)

An Entropy-guided Monte Carlo Tree Search approach for generating optimal Container Loading layouts. Ayodeji Remi-Omosowon, Richard Cant, Caroline Langensiepen. [1]

Applying Gamification to the Container Loading Problem. Ayodeji Remi-Omosowon, Richard Cant, Caroline Langensiepen.

Refereed and Accepted Conference Papers

Remi-Omosowon A, Cant R, Langensiepen C. *Hybridization and the Collaborative Combination of Algorithms.* UKSim-AMSS 16th Int. Conf. Comput. Model. Simul., Cambridge, United Kingdom: 2014, p. 404. doi:10.1109/UKSim.2014.60.

Remi-Omosowon A, Cant R, Langensiepen C. *Deriving an Entropy Measure for 2D Container Layouts.* IEEE UKSim-AMSS 17th Int. Conf. Comput. Model. Simulation, UKSim2015, Cambridge, United Kingdom: 2015, p. 1038. [2]

[1] Submitted to Omega: first reviewer accepted, second rejected; re-submission planned.
[2] Awarded a joint Best Paper Award along with 8 other papers out of 98 papers.

Remi-Omosowon A, Cant R, Langensiepen C. *Applying Gamification Principles to Container Loading in a Warehouse Environment.* IEEE UKSim-AMSS 18th Int. Conf. Comput. Model. Simulation, UKSim2016, Cambridge, United Kingdom: 2016, p. 79-86. [1]

Posters

Applying Computational Intelligence to Optimisation Problems in a Warehouse Environment. Case Study: The Container Loading Problem. Ayodeji Remi-Omosowon. Displayed at the 2015 Science & Technology Annual Research Conference, Nottingham Trent University, Nottingham, United Kingdom. May 2015.[2]

Applying Computational Intelligence to the Container Loading Problem. A Hybrid approach based on the Collaborative Combination of Algorithms. Ayodeji Remi-Omosowon. Displayed at the 2014 Science & Technology Annual Research Conference, Nottingham Trent University, Nottingham, United Kingdom. May 2014.

Formal Presentations and Talks

Load Optimisation Project: Computational Intelligence applied to the Container Loading Problem. Ayodeji Remi-Omosowon. End-of-Project Presentation given at NSK Europe Ltd. Newark-on-Trent, United Kingdom, 3rd August 2016.

Applying Gamification Principles to Container Loading in a Warehouse Environment. Ayodeji Remi-Omosowon. UKSim-AMSS 18th International Conference on Modelling and Simulation, Cambridge, United Kingdom, 06 April 2016.

Applying Computational Intelligence to Optimisation Problems in a Warehouse Environment. Case Study: The Container Loading Problem. Ayodeji Remi-Omosowon. 2015 Science & Technology Annual Research Conference, Nottingham Trent University, Nottingham, United Kingdom. 07 May 2015.

[1] Awarded a joint Best Paper Award along with 3 other papers out of 60 papers.
[2] Awarded the first prize for the Best Poster on display.

Deriving an Entropy Measure for 2D Container Layouts. UKSim-AMSS 17th International Conference on Modelling and Simulation Cambridge, United Kingdom, 25 March 2015.

Load Optimisation Project Report. Ayodeji Remi-Omosowon. Project Progress Presentation at NSK Europe Ltd. Newark-on-Trent, United Kingdom, 15 January 2015.

Hybridization and the Collaborative Combination of Algorithms. Case Study: The Container Loading Problem. UKSim-AMSS 16th International Conference on Modelling and Simulation, Cambridge, United Kingdom, 26 March 2014.

Computational Optimisation for Practical Applications. Remi-Omosowon Ayodeji, NTU Physical Sciences, Engineering and Computing Research Centre Seminar Series. Nottingham Trent University, Nottingham, United Kingdom, 22 January 2014.

Computational Optimisation for Practical Applications. Prof Dr Lars Nolle; Giovanna Martinez Arellano; Ayodeji Remi-Omosowon. Workshop at the AI2013 33rd SGAI International Conference on Artificial Intelligence in Cambridge. 10 December 2013.

Technical Reports

Load Optimisation Project. Project Summary. Ayodeji Remi-Omosowon. NSK Europe Ltd. Newark-on-Trent, United Kingdom, 11 May 2015.

Published Software Packages

pyeasyga. A simple and easy-to-use implementation of a Genetic Algorithm library in Python. https://github.com/remiomosowon/pyeasyga. Published June 2014.

Chapter 1

Introduction

The Container Loading Problem (CLP) is a long-standing problem of real-world importance, particularly in the logistics and distribution domain. It describes the general problem of packing a set of three-dimensional boxes into a larger three-dimensional rectangular container such that some given objective function is maximised e.g. maximising either the weight or volume capacity of the container.

In addition to maximising the given objective function, many real-world applications of the CLP also have relevant practical constraints applicable to the problem. These constraints could arise either from health and safety considerations relating to the problem, or from the nature of the goods being loaded and the practicalities involved in their handling.

While the addition of practical constraints to the CLP traditionally makes the already difficult problem even more complex, many of the constraints are relevant constraints that must be satisfied completely before solutions to a real-world version of the problem are considered viable. It is therefore obvious that in order to add value to the real-world applications of the CLP, more attention must be placed on solution approaches that tackle these practical real-world constraints.

This change in focus can be seen in recent work in the area, with several different approaches e.g. Ramos et al. [2016b], Bruns et al. [2016], Sheng et al. [2017], Le and Knust [2017], Männel and Bortfeldt [2017], Sridhar et al. [2017], Mostaghimi et al. [2017], Huang et al. [2016], Ramos et al. [2016a], Costa and Captivo [2016], Moura and Bortfeldt [2016], etc., making a significant impact on the consideration and specific inclusion of many different real-world constraints

to achieve feasible solutions. A gap still exists however in the consideration of the feasibility and complexity of the resulting layouts generated by various solution approaches, especially as experienced when dealing with heavy items that often need to be handled with limited manoeuvrability using a forklift truck, or some other means, via a single entry point into a container. This constraint affects the packing density that can be achieved as well as the overall simplicity of the layout generated. In this thesis, we will investigate solution approaches to the CLP that aim to satisfy a given number of practical constraints in addition to achieving the goal of producing loading layouts that pack items in a constrained space and have a consistency or are simple enough to be easily reproduced by forklift truck drivers loading real-world shipping containers from one end.

1.1 Background and Motivation

The Container Loading Problem (CLP) describes the problem of packing a given set of three-dimensional rectangular boxes into a larger three-dimensional rectangular container such that a given objective function is maximised, and if applicable, some other constraints are satisfied. A solution to this problem is said to be viable if all the selected boxes are packed completely within the walls of the container and there is no overlap between any of the packed boxes.

The particular class of the CLP that this thesis aims to solve is motivated by the loading problem that occurs in the UK distribution centre of an engineering and manufacturing company called NSK Europe Ltd. The problem is that of optimally loading a set of heavy palletised goods selected from a larger set of goods, each with a different size, weight and possible orientation, into any one container, while attempting to maximise the weight capacity of the container and to satisfy a number of relevant practical constraints. The problem can be generally classified using the typology defined by Wäscher et al. [2007] as a 'Single Large Object Placement Problem', further characterised with having certain restrictions placed on the items to be loaded i.e. additional constraints considered because of the nature of heavy palletised goods.

Forklift trucks are used to move, stack and load heavy palletised goods. A typical loading operation requires multiple trips from the warehouse to the con-

tainer and back. This operation can take up to an hour to complete in a perfect loading scenario. Loading mistakes or delays in deciding how to load a given selection of goods can further increase the time spent loading. It is therefore of great benefit to investigate and devise a means of reducing the time spent deciding what goods to load and how best to load them. Special care also needs to be taken when stacking goods on top of each other and when arranging goods in the container. This is to prevent any potential damage to goods that could arise from incorrect or non-optimal packing. Again, it is beneficial to ensure that the goods are packed optimally so that the possibility of any damage is greatly limited.

For this particular CLP instance, the following practical constraints were identified and must be considered in order for any solution provided to be considered feasible. [1]

- **maximum container weight limit:** when goods are selected for packing into containers, the total weight of all the selected goods must not exceed the maximum weight capacity of the container.
- **complete shipment of pallet groups:** pallets are often separated into logical groups e.g. a specific customer order; if any pallet that is a member of such a group is loaded into a container, all other pallets belonging to the same group must be loaded into the same container as well.
- **pallet stability:** pallets that are stacked on top of each other must have a combined height not greater than a given maximum stack height; this is to provide load stability when the stacked pallets are lifted off the ground and moved from place to place.
- **complete surface area support:** pallets not placed directly on the container floor, must be completely supported by the top surface area of the pallets they are placed on, i.e., no over-hanging of pallets is allowed.

[1] Note that in the rest of this chapter, and indeed the rest of this thesis, the term 'pallets' is used to refer to the phrase 'palletised goods' or the term 'goods', and all are used interchangeably. When there is a need to refer to the actual pallet base used to load the goods, the distinction will be made.

- **orientation and rotation constraints**: due to the manner in which palletised goods are packed, i.e., packed on pallet bases that facilitate the movement of the goods by forklift trucks, the palletised goods may only be picked up and packed in any one of two horizontal orientations; and as different types of pallet bases are used to pack the goods, some pallets may be rotated and packed in either of two possible orientations, while others, due to the nature of the pallet base used, might not allow rotation and allow packing in only one orientation.

- **stackability:** pallets can generally have other pallets placed on them, i.e., they are 'stackable'; restrictions, however, can be placed on a pallet so that no other pallet can be placed on it, e.g., a customer instruction to not double stack certain pallets, or to mark specific pallets as 'fragile'; if a pallet is identified in this manner as a 'non-stackable' pallet, it must not have any other pallet placed on it.

- **load bearing weight:** when a pallet is placed on another stackable pallet, the weight of the pallet placed on top, in the stack, must not be greater than the weight of the (bottom) pallet it is placed on.

- **maximum stack weight:** to keep in line with forklift truck regulations, when a pallet is stacked on another pallet, the combined weight of both pallets must not exceed the weight of the maximum load that the forklift truck is allowed to carry

Initial attempts have been made by the company to solve the problem using bespoke 'made-to-order' and proprietary 'off-the-shelf' software. These attempts were not successful because the solutions obtained were unfeasible as there were unable to satisfy the required practical constraints.

This thesis, in an attempt to provide a solution to this problem, investigates the use of computational intelligence techniques for optimally selecting a subset of heavy palletised goods for loading in reasonable time; and for generating many different loading arrangements (layouts), each a potential candidate for loading. The concept of entropy as a measure of disorder is adopted and applied to the generated loading layouts (see Remi-Omosowon et al. [2015]). This causes feasible

layouts to be rated better than non-feasible layouts. The best-rated layouts are then presented.

This thesis also investigates how loaders interact with the presented results in a gamified loading environment (see Remi-Omosowon et al. [2016]). The feedback from this interaction is used to explore the viability of a simulated loading environment as a learning aid. The premise is that if loaders can spend time loading in a simulated environment, the knowledge obtained can be transferred to loading in real-life situations and typically speed up the learning process.

More information about the problem being solved, the constraints considered, and the solution approaches considered can be found in Chapter 3.

1.2 Aims and Scope

This thesis presents a solution approach for a class of real-world container loading problems with good loading characteristics at low computational cost. In addition, the resulting layouts produced are also simple to understand and easy to reproduce by warehouse operatives driving forklift trucks. This class of problem is characterised with the loading of heavy palletised goods (hence the need for movement by forklift trucks) and a number of identified practical constraints that must be satisfied for a solution to be deemed feasible. The main aim of this thesis is to provide this solution approach as a contribution to container loading research.

A secondary aim is to examine and evaluate user interaction with the results of the otherwise complicated algorithms at work in the proposed solution approach, in a system where gamification principles have been applied to retain and increase user engagement. The effects of such interactions on the adoption of the overall loading system in the workplace are also discussed.

The problem considered is a variant of the Single Container Loading Problem (SCLP). No explicit solution is provided for the related Multiple Container Loading Problem (MCLP). A naive solution is however proposed in the concluding chapter, for the MCLP, involving the use of repeated applications of the solution approach provided for the SCLP.

1.3 Structure of the Thesis

The research presented in this thesis covers three themes. Some of the chapters presented sit directly within a given theme, while the rest span across the different themes.

The first theme covers the algorithmic optimisation of pallet selection. To this end, a hybrid algorithm for solving container loading problems, that focuses on obtaining a selection of pallets that can fit completely into a container, with the total weight utility of the container close to the maximum allowed, is introduced in Chapter 4. Improvements to the algorithm, that allows it to fit in more pallets into the same container in less time, are presented in Chapter 5. The hybrid algorithm in both cases satisfies all of the required constraints.

The second theme explores strategies for the optimal placement of the selected pallets within a container such that the resulting layout is achievable using forklift trucks and is easy for humans to understand. Chapter 6 presents the initial work done to optimise pallet placement within a container. In it, we derive a measure to rate layouts in terms of their disorderliness. This measure is then applied to a stochastic process that generates lots of randomised layouts in order to find and select the best-rated layouts automatically. Chapter 7 presents an algorithm that uses the derived measure to drive the placement of individual pallets (or stacks) within a single layout. The algorithm employs some elements of randomness during its placement which appeared to give it an edge over purely deterministic placement. It improves on the approach used in Chapter 6 in that it uses the measure to drive the generation of a single 'best' layout; as opposed to generating a lot of random layouts, rating them all using the measure, and then finally selecting a best one. As such, it presents a much faster means of generating layouts whose placement is directed by a measure that leads to the reduction of disorder in the layouts. This, in turn, leads to layouts that are easier to understand by loaders and, by the same virtue, easier to pack using forklift trucks.

The third theme is concerned with the (human-)factors that come to play when technology is introduced into the workplace, and it describes the process of gamifying the presented container loading system to increase user engagement

with the system. This theme is presented in Appendix A: Applying Gamification principles to the Container Loading Problem, and covers the application of gamification principles to the loading system and the effects they had on the adoption of the loading system. The loading system, in this case, refers to the hybrid algorithm and all things used to interact with it e.g. inputs, outputs, formalisms used to describe input to or output from the algorithm, interfaces used to control the algorithm, the medium used to interact with the algorithm, etc. A side effect of this gamification process is that the loading system is made more accessible to the loaders via a simulated loading environment. This environment is easy for the loaders to use and understand, as it abstracts away the otherwise complicated algorithms working behind the scenes.

A review of the relevant literature and previous work undertaken in the field is presented in Chapter 2, and the final conclusions of the thesis are presented in Chapter 8.

1.4 Contributions of this Thesis

The contributions of this thesis are:

- The development of a placement algorithm that uses the concepts of entropy and Monte Carlo tree search to generate layouts that are easy to understand and load using forklift trucks.
- The development of a framework for algorithmic hybridisation wherein the problem to be solved can be decomposed into sub-problems and each can be solved using any number of exchangeable algorithms as long as any imposed constraints are satisfied.
- Providing a novel solution to the Container Loading Problem that satisfies a unique combination of practical constraints.
- The development of an optimal approach for solving Container Loading Problems involving palletised goods.
- The development of a novel approach for identifying feasible loading layouts based on the use of a derived entropy measure.

- Integrating the above contributions (i.e. otherwise complicated algorithms) into a system that presents results in a manner that can be easily interpreted and understood by humans.

- Extending the existing container loading benchmark data, which mostly only covers weakly heterogeneous problem instances that deal with relatively few constraints at a time, to strongly heterogeneous instances that cover a larger number of constraints at a time; these additional constraints reflect a wide spectrum of practical applications that have not yet been dealt with extensively in literature.

1.5 Related Academic Publications

A significant part of this thesis has been published in 3 peer-reviewed papers in international conference proceedings. An extended version of one of these papers is under preparation for submission as a journal article. Ideas presented in two of the papers have been extended to include work not present in the original papers; this too, is under preparation for submission as a journal article. Some of the work has also been presented in 2 posters displayed locally at the University's annual Science & Technology conference.

The following list relates the published work to the relevant chapters in which they are presented.

- Chapter 4: A Hybrid Algorithm for the Container Loading Problem
  - Remi-Omosowon A, Cant R, Langensiepen C. *Hybridization and the Collaborative Combination of Algorithms.* UKSim-AMSS 16th Int. Conf. Comput. Model. Simul., Cambridge, United Kingdom: 2014, p. 404. doi:10.1109/UKSim.2014.60.
  - *Applying Computational Intelligence to the Container Loading Problem. A Hybrid approach based on the Collaborative Combination of Algorithms.* Ayodeji Remi-Omosowon. Poster displayed at the 2014 Science & Technology Annual Research Conference, Nottingham Trent University, Nottingham, United Kingdom. May 2014.

- Chapter 6: Optimising Container Layouts for Real-World Packing
  - Remi-Omosowon A, Cant R, Langensiepen C. *Deriving an Entropy Measure for 2D Container Layouts.* IEEE UKSim-AMSS 17th Int. Conf. Comput. Model. Simulation, UKSim2015, Cambridge, United Kingdom: 2015, p. 1038.
  - *Applying Computational Intelligence to Optimisation Problems in a Warehouse Environment. Case Study: The Container Loading Problem.* Ayodeji Remi-Omosowon. Poster displayed at the 2015 Science & Technology Annual Research Conference, Nottingham Trent University, Nottingham, United Kingdom.
- Appendix A: Applying Gamification principles to the Container Loading Problem
  - Remi-Omosowon A, Cant R, Langensiepen C. *Applying Gamification Principles to Container Loading in a Warehouse Environment.* IEEE UKSim-AMSS 18th Int. Conf. Comput. Model. Simulation, UKSim2016, Cambridge, United Kingdom: 2016, p. 79-86.

Chapter 2

Literature Review

2.1 Introduction

This chapter presents a review of relevant literature on the study of the Container Loading Problem (CLP), drawing on research that is focused on understanding the importance and implications of the consideration of practical real-world constraints on the solution approaches employed in solving the CLP. The review begins by providing an introduction to the CLP as a combinatorial optimisation problem in Section 2.2. It then situates the CLP in the class of problems known in complexity theory as *NP*-Complete problems, thus providing a justification for the heuristics-based solution approach employed in this thesis to tackle the problem. Section 2.3 identifies and provides a discussion of the typologies defined for the CLP in literature as well as the various solution approaches employed when dealing with CLPs. Finally, section 2.4 presents the practical constraints, particularly real-world constraints, typically considered when solving CLPs.

2.2 Optimisation Problems

Optimisation problems are problems that seek to find a selection of *optimal* choices or parameters to achieve a specific goal. The selected parameters are usually used to determine the minimum or maximum value of some function. This function is known as an *objective function* and when we seek to minimise

its value, it is referred to as a *cost function*; on the other hand, when we seek to maximise its value, it is referred to as a *fitness function*.

Optimisation problems can be divided into two categories: 'continuous' and 'discrete' based on the type of parameters, or *variables* involved. When the variables are discrete, the problem is known as a combinatorial optimisation problem, and it has a search space that comprises different combinations of values, one of which maximises or minimises a given objective function.

As many combinatorial problems with theoretical and practical importance abound in literature, many algorithmic techniques and solution approaches have been studied for solving them. A number of these approaches are based on mathematical programming methods that seek to find exact solutions to the problem. While the use of these exact methods is successful on some problems, there exist a class of problems for which they are not suitable.

Formally, in complexity theory, there is a class of problems P, which are problems that are solvable in polynomial time. There is also a class of problems NP (Non-deterministic Polynomial), which are problems where the validity of their solutions can be verified in polynomial time, but for which the solution search typically takes exponential time. The non-determinism is a reference to the concept of a non-deterministic Turing machine exploring both branches of an if-statement simultaneously, resulting in an exponential search space, in polynomial time. Having defined both classes of problems, P and NP, NP-Complete problems can be defined as the class of problems in NP which are as hard as any other problem in NP. They have the property of having a solution that can be proved or validated in polynomial time, but not having any known algorithm that can solve them in polynomial time. For completeness sake, there is also a class of problems known as NP-Hard problems, which are problems at least as hard as any problem in NP but may not be in NP themselves.

NP-Complete problems are *intractable*, as their solution search space increases exponentially with regards to their input size. This leads to exact methods being only feasible for problem instances with very limited input size. To remedy this problem, and to provide solutions to practical problems with large input sizes in reasonable time, other solution approaches based on heuristics and approximation algorithms have been developed and studied, that place emphasis on

providing high-quality 'optimal' solutions that are good enough and produced in very reasonable time.

Examples of some well-known heuristic methods include hill-climbing and gradient descent algorithms, that seek to determine a maximum fitness (or minimum cost) by iteratively taking steps towards better neighbour solutions in their search space; simulated annealing, based on a physics metaphor that also uses an iterative procedure and seeks to reach a maximum fitness using small steps that have been simulated and tested to see if they obtain a better fitness before being selected; and genetic algorithms, inspired by the biological metaphor of evolution which also uses an iterative process to arrive at an optimum solution, whilst employing safe-guards that are intended to help prevent it from being stuck in a local optimum.

The problem considered in this thesis is a practical real-world version of a combinatorial optimisation problem known as the Container Loading Problem (CLP). When expressed as the problem 'how many of these boxes can I fit in this container?', the CLP is *NP*-Hard. However, when expressed as the decision problem 'can these set of boxes fit in this container?', it can be seen to belong to the *NP*-Complete class; as any provided solution can be very quickly (in polynomial time) verified, i.e., does the solution fit all the given boxes into the container? In addition, as the CLP is known to reduce to the well-known Knapsack Problem [Scheithauer, 1992], which is already known to be *NP*-Complete [Pisinger, 1995], it is itself also *NP*-Complete. Most practical instances of the (*NP*-Complete) CLP are actually relatively easy to solve because the fill levels considered are fairly small. However, as the version of the CLP considered here occasionally has to deal with a few tight cases of very high fill level, as well as satisfy several practical constraints that make the problem even more difficult, the general solution approach proposed makes uses of heuristics, instead of exact methods, to determine a solution to the problem in reasonable time.

2.3 The Container Loading Problem

The Container Loading Problem (CLP) is an active research area with numerous applications in the real world, particularly in the container transportation and

distribution industries [Dereli and Da, 2010]. As indicated earlier, it is well known to be a NP-Complete problem [Dowsland and Dowsland, 1992] where the direct application of known mathematical formulations found for it in literature have been proven to be computationally feasible for only problem instances of a very limited size [Nepomuceno et al., 2007]. For this reason, the majority of the studies performed in this area, this thesis inclusive, focus more on providing solutions to the problem using heuristic and meta-heuristic approaches.

2.3.1 Typology

The definition provided for the CLP above can be said to describe a general version of the CLP. Several distinctions exist in the characteristics of CLPs, such as a difference in objective functions or a difference in the constraints considered, that result in variants to the problem that are treated as individual problems in their own right, each with its own degree of complexity and its own applicable solution approaches and techniques.

Dyckhoff, in his seminal work [Dyckhoff, 1990], provided a comprehensive typology for organising and classifying different CLPs. This brought together all the terms and different notions used in prior research into packing problems and provided a consistent language for describing the different problem types and variations. His work also served to help further the development of research that focused on specific problem types. Wascher et. al. later developed an improved typology [Wäscher et al., 2007], partially based on Dyckhoff's original typology, to deal with some of the deficiencies in it that had become apparent over time. They introduced new categorisation criteria different from Dyckhoff's and a new consistent system of names for the new problem categories.

In a broad sense, both typologies define two main distinctions for the CLP. In the first, we have two categories the first of which involves the loading of an entire set or a subset of small objects into a single larger object where the main objective is to maximise the utility of the larger object. The second category involves loading an entire set of small objects into one or more large objects where the main objective is to minimise the number of large objects used to pack the entire cargo.

Another distinction is made on the type of small objects to be loaded. Where there is only one type of small object, the small objects are said to be homogeneous; otherwise, they are said to be heterogeneous. Another further distinction is made on heterogeneous types; when there are a few distinct types with many objects of each type, the objects are said to be weakly heterogeneous; and when there are many distinct object types with few numbers for each object type, the objects are said to be strongly heterogeneous.

Using these distinctions, Wascher et al.'s typology [Wäscher et al., 2007] divides CLPs into the following problem types: Identical Item Packing Problem, Placement Problem, Knapsack Problem, Open Dimension Problem, Cutting Stock Problem and Bin Packing Problem. These problems are further broken down and categorised into more refined sub-problems; we refer the reader to Wascher et al.'s typology for detailed descriptions of the problems and their sub-problems.

In this thesis, the class of the CLP examined belongs to the class categorized as the Single Large Object Placement Problem (SLOPP) which is a sub-category of the Placement Problem but with the distinction of dealing with weakly heterogeneous small objects and a single large object. This problem (in its 3-dimensional form) is commonly referred to as the Single Container Loading Problem (SCLP) in the literature [Bortfeldt et al., 2003; George and Robinson, 1980; Li et al., 2014; Zheng et al., 2015; Zhu and Lim, 2012].

2.3.2 Solution Approches

Several solution approaches have been proposed in the literature for solving the CLP. The two most common heuristic approaches provided are the layering and wall-building approach. The layering approach, as seen in Bischoff and Ratcliff [1995a], Gehring and Bortfeldt [1997] and Bortfeldt and Gehring [2001], is based on the concept of packing items in a loading configuration from the ground up in layers. The wall building approach, first proposed in George and Robinson [1980], with a variant introduced in Moura and Oliveira [2005], is based on filling the container with walls where the walls are rectangle blocks made up of boxes whose depth is determined by the first box placed in them. Eley [2002] also presents

an approach based on wall-building that builds blocks of identical oriented boxes using a greedy heuristic. Another heuristic approach provided in literature is the AND/OR-graph approach proposed by Morabito and Arenalest [1994] in which boxes are represented as nodes in a graph and a cut performed on a box is represented as an AND operation. A sequence of cuts is performed until all nodes found are final. The set of all the nodes and AND operators is the AND/OR-graph. These approaches form the foundation for many heuristic frameworks used for solving the CLP in literature.

Meta-heuristic approaches such as Tabu search [Bortfeldt et al., 2003; Liu et al., 2011], Genetic Algorithms [Gehring and Bortfeldt, 1997; Nepomuceno et al., 2007; Pires de Araujo and Pinheiro, 2010], Simulated Annealing [Dereli and Sena Das, 2010; Peng et al., 2009], and Ant Colony Optimisation [Li Wang et al., 2010; Yap et al., 2012] have also been used extensively. These approaches have been explored extensively in the literature, and are often hybridized to solve specific variants of the CLP. More recently, parallel versions of some meta-heuristic approaches have been investigated; an example can be seen in Bortfeldt et al. [2003]. Other research shows trends of hybridizing heuristic methods with other techniques.

In this thesis, the solution approach employed is a collaborative combination of several problem-specific heuristic and meta-heuristic approaches. The approach is inspired by that used in Gehring and Bortfeldt [1997], which incorporated a tower building phase that reduced the problem into a two-dimensional problem of packing towers onto a container floor thus resulting in a reduction in dimensionality and a simplification to the original problem. In the proposed approach, an initial search is carried out in the solution space to select items with a combined weight less than the container weight limit, using a genetic algorithm. 'Weight' is an important criterion for driving this stage, because in the problem instance considered, the container weight limit is met long before the container gets filled up. The selected items are then built up into stacks using a greedy algorithm, and the built-up stacks are then packed using initially a genetic algorithm integrated with several rectangle packing algorithms, and then with a deterministic packing algorithm, and finally with an Entropy-guided Monte-Carlo algorithm, each corresponding to the different approaches used at different times while iteratively

improving the solution quality obtained.

2.4 Container Loading in the Real World: Practical Constraints

Practical constraints in the container loading problem turn up in different forms. Some constraints are related to the enclosing container, while some are related to the items being packed, and in some instances where the items being moved are heavy, some constraints are related to the forklift trucks used to move the items. These constraints may occur in practice as 'hard' or 'soft' constraints. Hard constraints must be satisfied, while soft constraints can have violations tolerated up to certain limits, if not completely satisfied.

A quick review of the literature turns up three basic constraints that seem to be a fundamental part of the canonical problem definition for the CLP, and were not explicitly observed, are assumed to be implicitly observed by most. These include: (i) the packing of boxes orthogonally to the walls of the enclosing container, (ii) the placing of all boxes completely within the walls of the container, and (iii) making sure boxes do not intersect each other. Due to the ubiquity of these constraints, some might not refer to them as 'practical' constraints; indeed these particular three have been referred to as 'geometric' constraints by some. In addition to these constraints, some solutions also insist that the entire base of a box must be fully supported by either the container floor or another box, while some allow a little overhang when boxes are placed on other boxes.

Apart from these basic constraints, the rest of the practical constraints typically encountered are related to orientation, stacking, stability, weight distribution, weight capacity, multidrop prioritisation, and complete shipment of item groups. These are covered in detail in the review by Bortfeldt and Wäscher [2013], who divides them into categories in relation to the container, the individual items being loaded, the entire cargo being loaded, and the positioning of the cargo. I will briefly describe a number of these here.

In practice, a typical container has weight limits that must not be exceeded. In most container loading problems, the volume of the container is the objective

function that is maximised. When heavy goods are to be loaded, the weight limits are often met before the volume limits are encountered. In cases like these, the weight of the container becomes the objective function that is maximised as the weight limits become more restrictive than the volume limits. Weight limit constraints are typically treated as hard constraints. Another constraint related to weight is the weight distribution constraint. It requires the distribution of weight to be spread out almost evenly across the container floor. This is important in order to satisfy axle weight guidelines for the trucks that transport the containers, as well as to provide a stable load that reduces the movement of cargo when the container is in transit. When considered, weight distribution constraints are often treated as soft constraints.

The orientation constraint is the most common constraint considered in the CLP literature. In theory, there can be up to six different orientations possible i.e., three vertical orientations each having two horizontal orientations. In practice, however, the vertical orientation is often fixed e.g. goods that have stickers with directions to load 'This way up ↑', or items that have to be stood up a particular way. This results in there being only two possible orientations in which to pack items. Additional limits can also be placed on the horizontal orientations allowed. This can be seen for example in the problem considered in this thesis (described in Chapter 3) where different types of pallets are used to pack goods, and some of the pallets allow loading using forklift trucks from only two sides i.e., the front and the back, while others allow loading from all four sides. When we consider that we load the palletised goods orthogonally in the container, and account for symmetry, then the pallets that allow loading from all four sides give us two possible horizontal orientations while the pallets that allow loading from only two sides give us only one possible orientation in which the pallet can be packed. Orientation constraints are treated as hard constraints in the literature.

Stacking constraints are usually introduced as hard constraints to help prevent damage to the items being packed. They are concerned with restrictions on how boxes are placed on top of each other and are also referred to as 'load-bearing' constraints in the literature (see Junqueira et al. [2012b]) because they are concerned with the load-bearing strength of boxes and how much weight they can sustain before they get damaged. Several methods exist for dealing with

stacking constraints in practice. One such method is to always require heavier items to be placed below lighter ones; another separates items into groups of 'stackable' and 'non-stackable' items where stackable items are those that can have other items placed on them and non-stackable items can not. The choice of which items are stackable or not could be determined by the shape of the top surface of the item to be packed, for example, if the top surface is uneven. It could also be determined by a 'Do not double-stack' directive. Items might also be marked as 'fragile' with a meaning attached that directs loaders not to place the items on any other items and not to have other items placed on them. Another observed method considers the item density and places items of higher density below items of lower density.

The complete shipment constraint refers to the case where if an item that belongs to a subset of items is loaded into a container, all other items belonging to the same subset must be loaded as well. Examples of this might include the loading of different furniture parts, or as with the problem dealt with in this thesis, items belonging to the same customer order. In both examples, if a single part of the subset is loaded, the rest must be loaded as well. Another case for this constraint is mentioned in literature, the difference lying in the number of containers the items are loaded into. In this case, if an item that belongs to a subset of items is loaded into one of many containers used to load a given shipment, it is sufficient that the other items in the same subset are loaded as part of that shipment and not necessarily in the same container. When dealt with in literature (e.g. see Eley [2003]), complete shipment constraints are treated as hard constraints.

Loading priority constraints refer to a situation where a subset of items must be loaded from a given set of items. For example, this might be because of a deadline placed on the delivery of the particular subset of items, or because of the items having a higher delivery priority than others e.g. first class post items will be given more priority than second class post items. This constraint is usually treated as a hard constraint where we find that all high priority items must be loaded first before any low priority item is loaded.

Positioning constraints deal with the restriction on the position of items within a container. The literature distinguishes between 'absolute' and 'relative' posi-

tioning. With absolute positioning, items are restricted to (or restricted from) very specific locations, i.e. absolute positions, within a container. Relative positioning, on the other hand, restricts the placement of items relative to each other e.g. requiring that items belonging to the same customer order be placed next to each other within the container. In practice, situations that require the delivery of packed items to multiple locations exhibit both absolute and relative positioning constraints. The items that will be delivered to the same location are kept close together relative to each other, while groups of items are kept in absolute positions in the container such that each group can be unloaded according to the order of the locations being delivered to. Here, they are mostly treated as hard constraints.

The stability constraint deals with how stable the items are when being packed or unpacked, and how stable an entire packed load is when being moved. They are of significant importance in the literature and are often presented in the form of requiring that the bottom surface area of an item to be packed must be completely supported by either the top surface area of another item or the container floor. In some cases, where an item is placed on another item, partial support that results in a little overhang may be allowed (see Gehring and Bortfeldt [1997]; Tarantilis et al. [2009]). Interlocking arrangements in the load may also be used to reduce motion during transit. The use of 'filler' materials is also introduced in practice to plug any gaps left after loading to keep entire loads stable.

Pattern complexity constraints deal with how easy it is for generated loading patterns to be understood and implemented by human loaders or robots. It is of importance because complex patterns that achieve a very compact and high container fill may not be implementable by human loaders or loading robots without considerable extra effort. This results in loading patterns that are easy to describe and pack being more desirable in practice than complex patterns that might obtain a higher fill. An example of the complexity constraint in practice is the use of the 'guillotine' pattern. It is a pattern that can be obtained by a series of cuts made parallel to the walls of the container. It is frequently considered in the literature as it can easily be described and packed.

2.5 Conclusion

The solution approaches considered in the literature for the CLP can be broadly divided into 'exact and approximate algorithms' and 'heuristic and meta-heuristic' algorithms. While quite a lot of progress has been made in the area of the exact and approximate algorithms for the CLP [Zhao et al., 2014], considering that there have been only a few proposed so far, they have so far proven to be feasible only for problem instances of limited size. Heuristic and meta-heuristic approaches, on the other hand, have been shown to be a good vehicle for solving problems with practically-relevant constraints and typically provide good quality solutions in reasonable time for problems with realistic sizes. Despite this, the focus on real-world practical constraints in the literature has been very little in comparison to the body of work present. There appears to be more research dealing with an idealised form of the CLP which typically has just a few or no constraints present. Indeed, for this reason, Bortfeldt and Wäscher [2013] in their review remark that research in container loading has to be looked upon as being in its infancy with respect to the inclusion of practically-relevant constraints. They report that only 26 out of the 163 papers they reviewed considered 4 or more constraints simultaneously.

In this regard, one can see that the literature addressing the issue of dealing with multiple practical constraints simultaneously is scarce. This chapter, therefore, presents a review of the Container Loading Problem (CLP), situating it particularly in the context of its use practically in the real-world, as well as in the work environment. We consider the category of the CLP identified by the two main typologies for the CLP in the literature [Dyckhoff, 1990; Wäscher et al., 2007] as the 'Single Large Object Placement Problem' (also commonly referred to as the Single Container Loading Problem) and present a review of a number of relevant practical constraints, i.e., weight limit, orientation, pattern complexity, stability, complete shipment, loading priority, and stacking. The work presented attempts to deal with these constraints simultaneously. In cases where it appears a constraint is not explicitly dealt with, e.g. loading priority or positioning, a workaround is presented for dealing with the constraint using the same body of work presented in this thesis.

While a number of standard solution approaches for solving CLPs are presented, the focus in the rest of this thesis is motivated by real-world applications of the problem. In dealing with the problem, we realise that the real-world constraints encountered in practice are what make the problem different from the canonical CLP found in the literature. Indeed, one could argue that the combination of a number of given constraints turn each CLP into a unique problem different from other CLPs. To this end, non-standard solution approaches that deviate from or extend the norm might be expected to be better suited to problems with such unique combination of constraints. In the rest of this thesis, a number of approaches that have been applied successfully to other areas but have to the best of my knowledge not been previously applied to the area of Container Loading are investigated. These concepts, which include the use of 'entropy' as a measure and the use of 'Monte Carlo tree search' to guide search, are considered, and their implementation and use in the CLP is outlined in subsequent chapters.

Chapter 3

Problem Overview

3.1 Introduction

The problem this thesis attempts to solve is motivated by a real-world container loading problem that occurs in the United Kingdom distribution centre (UKDC) of an engineering and manufacturing company, NSK Europe Ltd., that is one of the largest bearing suppliers in the world. The UKDC serves as the logistics and distribution department for the company, and the problem instance and specific practical constraints considered in this thesis are therefore a result of a practical case study developed there.

In the UKDC, bearings are packed into boxes which are put into cartons and arranged on pallets which are then shrink-wrapped to be loaded as individual units onto shipping containers for transportation to customers. We will subsequently use the terms "pallets" or "palletised goods" interchangeably to refer to these individual shrink-wrapped units and use the term "pallet-base" to refer to the actual base on which the cartons have been arranged. A typical pallet is heavy, weighing 445 kg on average, and has to be moved about using forklift trucks. Pallets are loaded into 40 ft containers with dimensions presented in Table 3.1. There are 4 different pallet-base types currently used in the UKDC; these types are shown in Table 3.2. Each pallet-base type has different implications for the orientation in which they can be placed on a container floor using a forklift truck, and the way they can be arranged side-by-side within the confines of the

Table 3.1: Dimensions of the container used to load pallets in the UKDC

Container Dimensions	
Length	1203 cm
Breadth	235 cm
Maximum stack height	210 cm
Maximum weight capacity	25 999 kg

container (see Figures 3.1 and 3.2).

The problem belongs to the first category of Dyckhoff and Gerhard's classification [Dyckhoff, 1990], i.e., using a single container and a weakly heterogeneous rectangular box set. The problem is also categorised as the Single Large Object Placement Problem (SLOPP) according to Wascher et al's improved typology [Wäscher et al., 2007], which is a sub-category of the Placement Problem as defined in their typology, with the distinction of dealing with weakly heterogeneous small objects and a single large object.

Multiple pallets, potentially having different pallet-bases, may be logically grouped together as a single job. Single jobs must not be split across different containers and must have all of their constituent pallets loaded completely onto a single container. If they can't all be loaded, none of the pallets belonging to that job must be loaded. Forklift trucks are used to move, stack and load the heavy pallets, which are typically spread across different locations in the warehouse (see Figure 3.3); with a typical loading operation requiring multiple trips from the warehouse to the container and back. A loading operation can take up to an hour given a perfect loading scenario; with loading mistakes or delays in deciding how to load a given selection of goods further increasing the time spent loading. It is therefore of great benefit to investigate and devise a means of reducing the time spent deciding what goods to load and how best to load them.

The problem encountered in the UKDC is therefore that of optimally loading a set of heavy palletised goods selected from a larger set of goods, each with a different size, weight and possible orientation, into a single container, while attempting to maximise the weight capacity of the container and to satisfy a number of other relevant practical constraints. Due to the nature of the heavy palletised goods involved, additional constraints are considered due to the re-

Table 3.2: Pallet-base types: Dimensions and allowed orientations

Pallet-base	Length	Breadth	Orientations
STD	70 cm	80 cm	Both
NSK	105 cm	78 cm	Single
EURO	120 cm	81 cm	Both
EURO2	80 cm	60 cm	Single

Figure 3.1: Pallets on the 'STD' pallet-base type

strictions encountered with respect to how the pallets are physically placed and arranged on a container's floor using forklift trucks. As the value of the entire cargo in a fully loaded container can range from an average of £24 000 to £120 000, it is important to take care when stacking pallets on top of each other and to arrange the resulting stacks carefully in the container in a manner that allows for safe, easy and quick loading (or unloading) using forklift trucks; this should help prevent any potential damage to the palletised goods that could arise from incorrect or non-optimal packing.

Figure 3.2: A pallet with the 'NSK' pallet-base type (left) and the 'EURO' pallet-base type (right)

Figure 3.3: Palletised goods stored in various locations in a warehouse

3.2 Constraints

Discussion with the end users at the UKDC revealed some constraints on solutions that were absolute. Firstly, there was a maximum overall weight that must not be exceeded by the total weight of all the pallets loaded onto a single container. Secondly, no more than two pallets could be stacked and no upper pallet could be loaded across more than one lower pallet for health and safety reasons. Additionally, due to the physical limitations of manipulation via forklift trucks, some pallets had to be loaded in a single orientation, while others could be loaded in both possible horizontal orientations. Finally, the requirement to dispatch an entire container's cargo, to a particular destination means that pallets could be divided into groups containing any combination of pallets, such that if a pallet was selected to be packed into the container, all other pallets in the same group must be completely packed as well. Results that do not satisfy these constraints were not acceptable. In addition to these, the following practical constraints are also considered:

- pallets have a 'stackable' property that indicates if they can have other pallets placed on them; if a pallet is not stackable, it must not have any other pallet placed on it
- stackable pallets must not have a pallet with greater weight placed on them; only pallets with weight less than or equal to that of the stackable pallet are allowed to be placed on it
- for stability during container transit, pallets not placed directly on the container floor must be completely supported by the surface area of the pallets they are placed on
- for stability when being moved by forklift trucks, pallets that are stacked on top of each other must have a combined height less than or equal to a given maximum stack height
- to keep in line with forklift truck regulations, pallets stacked on top of each other must have a combined weight less than or equal to the maximum weight the forklift truck is allowed to carry

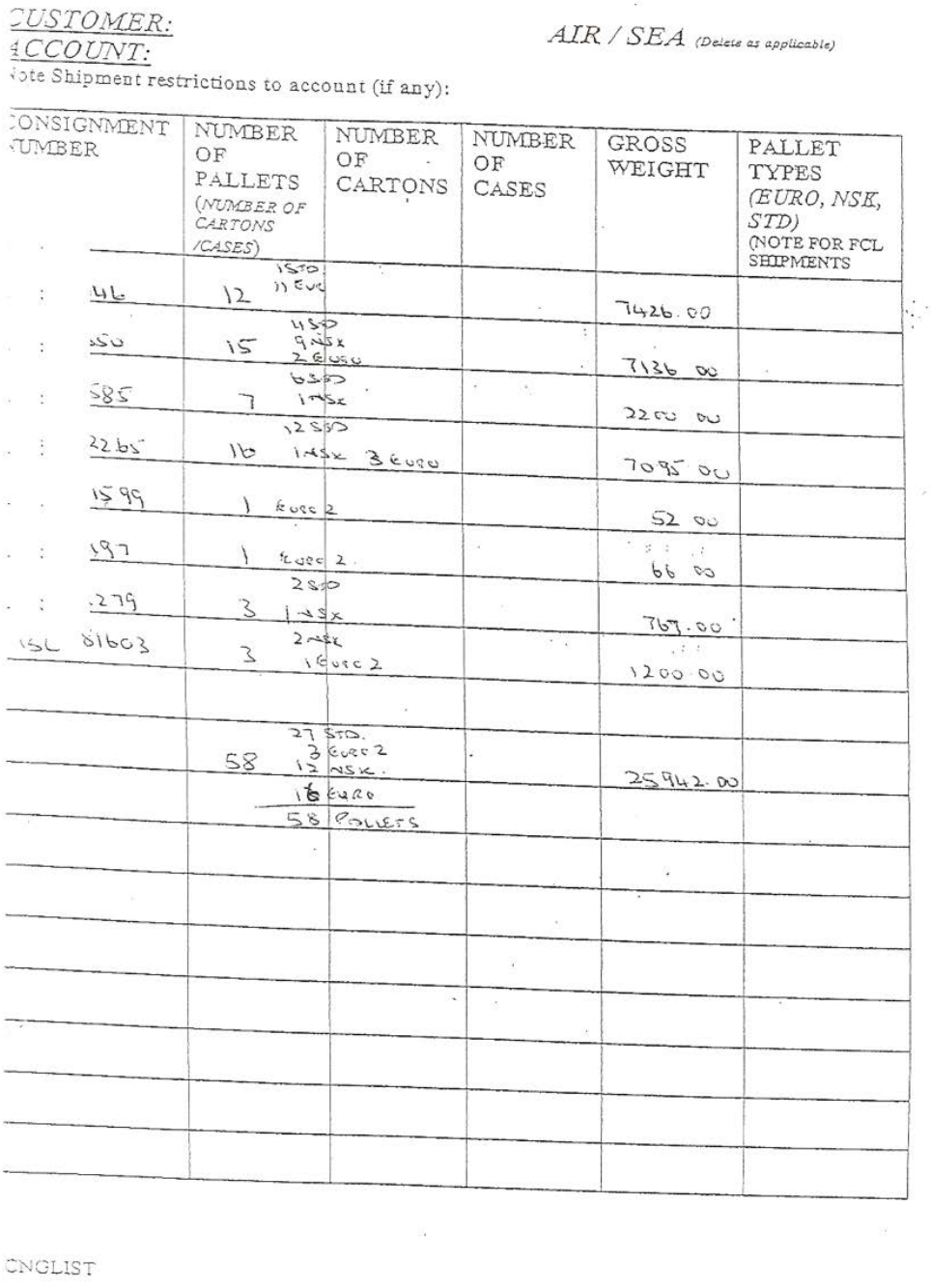
- finally, pallets must be packed orthogonally, parallel to the sides of the container and completely within the confines of the container's walls.

Results that do not satisfy these constraints are not feasible.

3.3 The Manual Process

The loading process in the UKDC begins when a sales operative has a list of packed pallets ready to be shipped out to customers. These packed pallets will often be separated into different logical groups each representing a single customer order, or job. Using a pen, paper and a calculator, the operative, considering only the pallet weights and volume, attempts to select a combination of jobs that have a combined weight close to or equal to the maximum weight capacity of a container. The selections total volume is also checked to see if it is less than or equal to the volume capacity of the container. Once such a selection is found (see Figure 3.4), it is sent to warehouse operatives with loading experience to determine if the selection will fit into the container or not. The loaders evaluate the selection considering not only the pallet weights and volume but also the types of pallet-base in use, which provides them with pallet size and orientation information. Armed with this knowledge, they tap into their know-how and prior experience to determine if they think the selection can fit into the container or not. If they think the selection will fit, they notify the sales operative who then books a container for loading and sends the loaders a detailed packing list (see Figure 3.5); otherwise, the entire selection process is repeated and the new selection is also checked to see if it will fit.

When a selection that fits has been identified, it is then up to the loaders to work out how best to load the selection into a container as efficiently as possible while satisfying required practical constraints. They determine this by drawing on their years of experience to come up with a loading plan. Once they have the plan, they generate pallet stacks and proceed to load the stacks on the container floor using forklift trucks.



CUSTOMER:
ACCOUNT:
Note Shipment restrictions to account (if any):

AIR / SEA (Delete as applicable)

CONSIGNMENT NUMBER	NUMBER OF PALLETS (NUMBER OF CARTONS /CASES)	NUMBER OF CARTONS	NUMBER OF CASES	GROSS WEIGHT	PALLET TYPES (EURO, NSK, STD) (NOTE FOR FCL SHIPMENTS
46	12 1 STD. 11 EURO			7426.00	
550	15 4 STD 9 NSK 2 EURO			7136 00	
585	7 6 STD 1 NSK			2200 00	
2265	16 12 STD 1 NSK 3 EURO			7095 00	
1599	1 EURO 2			52 00	
197	1 EURO 2			66 00	
279	3 2 STD 1 NSK			769.00	
ISL 81603	3 2 NSK 1 EURO 2			1200 00	
	58 27 STD. 3 EURO 2 12 NSK. 16 EURO 58 POLLETS			25942.00	

ENGLIST

Figure 3.4: Manual selection of customer orders that add up to 25 942 kg (close to the maximum container weight limit for a 40ft container: 26 000 kg)

SUMMARY PACKING LIST

14:10:04
Page : 1

MARKS:

Container Number	Type	Gross Wt.	Net Wt.	Length	Width	Height	Cube
)518/00001	PALLET	268.000	259.000	80	70	64	0.3584
)518/00002	PALLET	515.000	505.000	105	75	72	0.5670
)518/00003	PALLET	552.000	542.000	105	75	73	0.5749
)518/00004	PALLET	555.000	545.000	105	75	72	0.5670
)518/00005	PALLET	346.000	336.000	105	75	47	0.3701
)518/00006	PALLET	264.000	254.000	105	75	69	0.5434
)518/00007	PALLET	416.000	406.000	105	75	58	0.4568
)518/00008	PALLET	403.000	393.000	105	75	58	0.4568
)518/00009	PALLET	261.000	252.000	80	70	76	0.4256
)518/00010	PALLET	371.000	362.000	80	70	95	0.5320
)518/00011	PALLET	156.000	147.000	80	70	60	0.3360
)518/00012	PALLET	549.000	527.000	120	81	76	0.7387
)582/00001	PALLET	776.000	766.000	105	75	81	0.6379
)582/00002	PALLET	332.000	322.000	105	75	56	0.4410
)582/00003	PALLET	418.000	409.000	80	70	99	0.5544
)582/00004	PALLET	553.000	543.000	105	75	65	0.5119
)582/00005	PALLET	352.000	343.000	80	70	73	0.4088
)582/00006	PALLET	450.000	440.000	105	75	56	0.4410
)582/00007	PALLET	261.000	252.000	80	70	64	0.3584
)582/00008	PALLET	500.000	490.000	105	75	56	0.4410
)582/00009	PALLET	924.000	914.000	105	75	85	0.6694
)582/00010	PALLET	440.000	430.000	105	75	56	0.4410
)582/00011	PALLET	653.000	643.000	105	75	70	0.5513
)582/00012	PALLET	322.000	313.000	80	70	74	0.4144
)582/00013	PALLET	393.000	384.000	80	70	91	0.5096
)582/00014	PALLET	310.000	301.000	80	70	66	0.3696
)595/00001	PALLET	316.000	307.000	80	70	74	0.4144
)595/00002	PALLET	331.000	322.000	80	70	74	0.4144
)595/00003	PALLET	308.000	299.000	80	70	74	0.4144
)595/00004	PALLET	281.000	272.000	80	70	71	0.3976
)595/00005	PALLET	296.000	287.000	80	70	73	0.4088
)595/00006	PALLET	441.000	432.000	80	70	89	0.4984
)595/00007	PALLET	347.000	338.000	80	70	74	0.4144
)595/00008	PALLET	317.000	308.000	80	70	74	0.4144
)595/00009	PALLET	319.000	310.000	80	70	74	0.4144
)595/00010	PALLET	154.000	145.000	80	70	51	0.2856
)595/00011	PALLET	400.000	391.000	80	70	74	0.4144
)595/00012	PALLET	258.000	249.000	80	70	59	0.3304
)595/00013	PALLET	366.000	357.000	80	70	74	0.4144
)595/00014	PALLET	184.000	175.000	80	70	67	0.3752
)595/00015	PALLET	345.000	336.000	80	70	74	0.4144
)595/00016	PALLET	343.000	334.000	80	70	74	0.4144
)595/00017	PALLET	349.000	340.000	80	70	74	0.4144
)595/00018	PALLET	368.000	359.000	80	70	73	0.4088
)595/00019	PALLET	276.000	267.000	80	70	73	0.4088
)595/00020	PALLET	276.000	267.000	80	70	73	0.4088
)595/00021	PALLET	74.000	65.000	80	70	39	0.2184
.367/00001	PALLET	309.000	300.000	80	70	79	0.4424
.367/00002	PALLET	282.000	273.000	80	70	66	0.3696
.367/00003	PALLET	306.000	297.000	80	70	78	0.4368
.367/00004	PALLET	499.000	489.000	105	75	59	0.4646
.367/00005	PALLET	617.000	607.000	105	75	71	0.5591
.367/00006	PALLET	418.000	408.000	105	75	59	0.4646
.367/00007	PALLET	238.000	229.000	80	70	79	0.4424
.367/00008	PALLET	420.000	410.000	105	75	59	0.4646
.367/00009	PALLET	619.000	609.000	105	75	71	0.5591
.367/00010	PALLET	628.000	618.000	105	75	74	0.5828
.367/00011	PALLET	617.000	607.000	105	75	71	0.5591
.367/00012	PALLET	479.000	469.000	105	75	59	0.4646

Figure 3.5: A detailed packing list

3.3.1 Identified Shortcomings

The process as it stands is not as efficient as the management in the UKDC would like it to be. A significant amount of time is spent during the process when the sales operatives and the loaders are trying to determine and find a selection that can fit completely in a container. This part of the process is an important part to get right as a wrong fit will result in a lot more time wasted when selected pallets are packed up to a point and then have to be unpacked when it is found that the entire load will not fit completely in the container. In a bid to automate and optimise the manual process, the UKDC have embarked on research towards a computerised loading optimisation system in order to:

- increase overall loading speed and efficiency,
- save time and avoid conflict and confusion between operatives when they disagree on pallet selections, and how best to load them, reduce the total costs incurred when hiring containers, by optimally maximising the capacity of every loaded container to reduce the overall number of containers required for loading,
- reduce damage to goods that might arise from non-optimal packing within the container, thereby reducing costs that are normally associated with the replacement of damaged goods and customer fines for the receipt of damaged goods,
- provide greater customer satisfaction by speedily processing and loading customer orders for safe and prompt delivery,
- increase warehouse throughput, i.e., the more goods that can be loaded and shipped out of the warehouse, the higher the capacity to process new orders within the existing warehouse space, thus leading potentially to more business for the company,
- make loading know-how available to all operatives.

This should have the overall effect of significantly improving business performance and raising competitive edge while providing greater customer satisfaction.

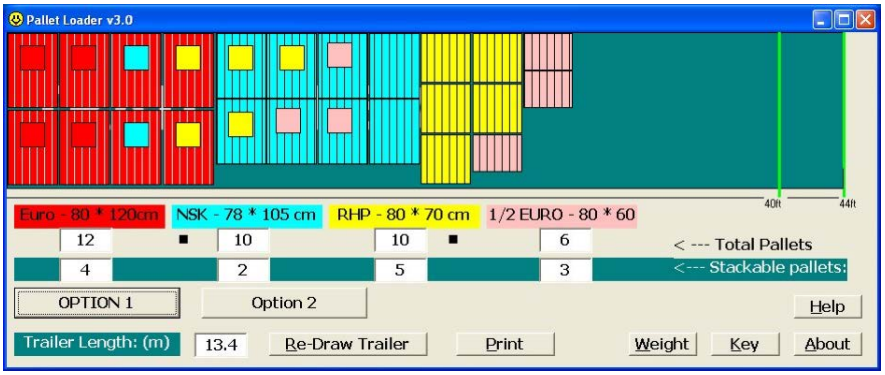


Figure 3.6: The bespoke 'Pallet Loader' software developed for the UKDC loading problem

3.4 Initial Solution Approach

Since as far back as 2005, the UKDC has been trying to optimize its loading process. Initial attempts made to solve the loading problem involved the use of proprietary off-the-shelf software. The main problem faced with this approach was that the systems examined were not flexible enough to accommodate and satisfy all of the required constraints. Subsequent attempts were then made to develop bespoke software for the problem. One notable attempt made in 2005 produced software called the 'Pallet Loader'. The software was built by a computer science work-placement student employed by the company at the time. A screenshot of the software in operation can be seen in Figure 3.6. It was also deemed unsuccessful because it could not deal with new pallet-bases with sizes different from those it already had hardcoded into it; it also assumed all pallets where stackable and could have other pallets placed on them, which in reality was seldom the case; it did not take into account individual pallet weights and as such could not deal with any weight-related constraints; and, it sometimes generated unfeasible loading layouts such that pallets with much larger bottom surface areas would be placed on pallets with a smaller top surface area.

3.5 Proposed Solution Approach

As alluded to earlier in the literature review, constraints encountered in practice are very often of great importance and a number of them must be completely satisfied simultaneously in order for any provided solutions to be considered feasible. The main issue faced by previous solution attempts was directly related to this inability to completely satisfy all of the considered constraints. This meant that while some solutions might have been provided, and on paper or in theory might have looked okay, they would have failed to make any sense or be of any value in practice.

Detailed discussions with the loaders led to a heuristic approach that was a natural fit for the problem as experienced in the UKDC. The approach involved separating the problem into sub-problems where each sub-problem is solved independently. The approach also managed to take care of all the considered constraints. The separation breaks down the problem into the sub-problems of (i) selecting pallets for loading into a container subject to a maximum weight constraint, (ii) stacking the selected pallets subject to a number of given constraints, and (iii) packing the selected stacks completely on the container floor. It was inspired by a similar separation that occurs in the manual process for solving the problem.

Mirroring this breakdown into sub-problems, allowed for a deeper focus on each individual sub-problem leading to the proposal of well-known solution approaches for solving each sub-problem. We also ended up with an easier way to cover all the constraints considered, as the constraints separate nicely into the different sub-problems. A consequence of this is that for each sub-problem, we find ourselves dealing with a smaller number of constraints than we would have if considering all of the considered constraints as a combined whole. This results in sub-problems that are easier to solve individually.

3.5.1 The Selection Problem

In the context of the overall loading problem in the UKDC, the selection problem is simply the problem of selecting a combination of jobs that have combined weight close to that given as the maximum for a container, i.e., the very first

part of the problem involving only the sales operative when he/she generates the manual selection list using just pen and paper (Figure 3.4). It essentially finds a selection of pallets whose total weight maximises the weight capacity for a specific container, and is equivalent to the one-dimensional knapsack problem Pisinger [1995] which can be modelled mathematically as:

$$\begin{aligned} &\textit{Maximise} && \sum_{i=1}^{n} v_i x_i \\ &\textit{subject to} && \sum_{i=1}^{n} w_i x_i \leq W, \qquad x \in \{0,1\} \end{aligned}$$

where x_i is a binary variable equal to 1 if item i should be included in the knapsack, or 0 otherwise; n is the total number of items available; v_i and w_i are the value and weight of item i respectively; and W is the maximum weight capacity of the knapsack.

Relating the knapsack problem back to the selection problem, the knapsack represents the container, and jobs are the items to be selected. The value (and weight) of a job is its weight. We therefore have x_i representing an individual job, n representing the total number of jobs, W representing the maximum weight capacity of the container, and v_i and w_i both representing the combined weight of all pallets in job i. The problem is then to obtain a selection of jobs with weights less than or equal to the weight capacity of the container.

A genetic algorithm (GA) is proposed to solve this problem as it is a well-known meta-heuristic approach adopted for solving difficult combinatorial optimization problems: see De Jong [1975], Davis [1989], Thierens and Goldberg [1994], Rudolph [1994], Palmer and Kershenbaum [1995], Reeves [1997], and Cheng et al. [2000] in Tang [2011]; and given the size of the problem, can be guaranteed to provide good results. Also, by solving this problem using a GA, we take advantage of the fact that at the end of a typical GA run, we are presented with multiple solutions of varying quality and fitness. These multiple solutions provide us with alternative selections that can be examined if the first selection examined is found not to fit; this mimics perfectly the situation in the manual

solution approach when the sales operative has to produce a new selection if the previous selection produced was found to be unsuitable by the loaders.

In the operation of the proposed GA when generating a selection of jobs, we have implicitly taken care of the practical constraint that requires keeping pallets belonging to the same job together. This is because the 'items' we pick for selection into our knapsack are the individual jobs (which themselves are made up of related pallets grouped together), rather than the individual pallets. As such, no extra work needs to be done to ensure that selected pallets are only selected if every other pallet belonging to its groups is selected as well. Another practical constraint that we deal with in the selection problem is that of ensuring that the maximum weight of the selected pallets is less than the maximum weight for the given container. This constraint is explicitly handled by the GA, as intermediate solutions that are found to violate the constraint during its operation are penalized, and only solutions that satisfy the constraint are considered.

As an alternative approach to using a GA, a Dynamic Programming (DP) approach was considered as it is also a well-known algorithm for solving the one-dimensional knapsack problem in pseudo-polynomial time [Martello and Toth, 1990; Pisinger, 1995]. Further examination however revealed that using this approach would require more time and computational effort, as we would have to do extra work to ensure that we are able to generate selections that add up to the same weight but are made up of different combinations of jobs; in the GA approach, we essentially got this for free. Additionally, we would also have to perform multiple repeated runs using the DP approach to target different weights (we looked at starting at the maximum and working our way down). Again, discussions around what step sizes to use when decreasing the target weights, and whether or not to use unit sizes thereby targeting every possible weight, or using fixed step sizes to decrement the weight, which could lead to missing out on good solutions for the weights in between, further made stronger the case for not using the DP approach. Another factor considered when ruling out the DP approach was that it is known to be memory intensive: Martello and Toth in Martello and Toth [1990] mention that they do not consider a DP approach in their computational experiments for the one-dimensional knapsack problem because of its

'excessive memory requirements'.

3.5.2 The Stacking Problem

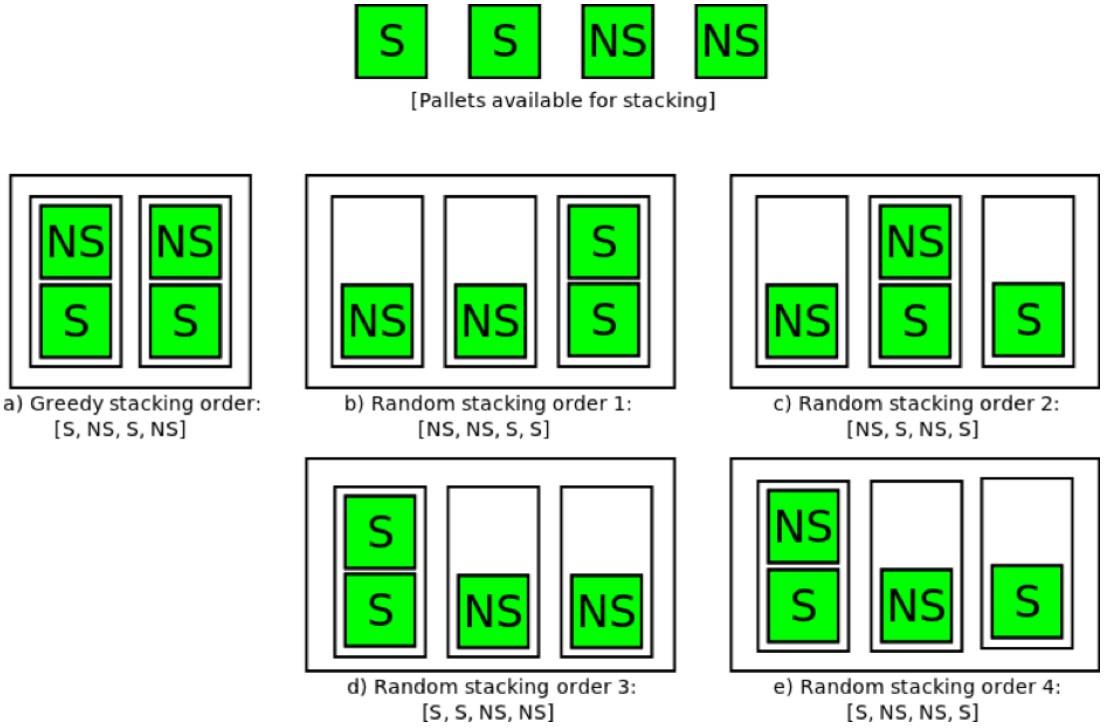
The stacking problem in the UKDC occurs when a loader receives both the selection and detailed packing lists from the sales operative. The problem involves the generation of stacks, which are simply pallets placed on top of each other, that are then moved as a unit using forklift trucks to be arranged on the container floor, all while satisfying the different number of stacking-related constraints. To solve the problem manually, the loader typically eyeballs the list of pallets taking note of the numbers of different pallet-bases used and the individual weights for each pallet. He/she then mentally works out the number of stacks that can be produced from the list while considering the related constraints. The different pallet-base sizes come into play because they determine the top and bottom surface areas of pallets that are considered in one of the constraints, and the weights are also used when satisfying the weight-related stacking constraints. After stacks are generated, they are then packed onto a container floor.

The proposed solution to the stacking problem is a greedy algorithm that is partly inspired by the approach employed in Gehring and Bortfeldt [1997]. We took insight from their tower-generation process and came up with a procedure to greedily generate stacks subject to the unique combination of constraints faced in the UKDC. The proposed greedy algorithm will separate pallets into 'stackable' and 'non-stackable' groups based on whether other pallets can be placed on them or not. Both groups will then have their pallets sorted in descending order by weight. At the core of its operation, the greedy algorithm will proceed to generate stacks from both groups by selecting pallets from the stackable group before the non-stackable group as bottom pallets for stacks; and by selecting pallets from the non-stackable group before the stackable group as the top pallets. This way, we ensure that all stackable pallets are considered before non-stackable pallets as bottom pallets in a stack so we do not end up in a situation where a stackable pallet that could have been used as a bottom pallet is wasted as a top pallet. Sorting both groups in descending order by weight ensures that heavier pallets are selected before lighter ones. If we get a situation where a non-stackable pallet

is selected as the bottom pallet of a stack, it would mean there were no more stackable pallets to consider. Once a bottom and top pallet are selected, we have a complete stack. If only a bottom pallet can be selected (i.e. either there are no more pallets to place on it, or the pallet itself is non-stackable and cant have other pallets placed on it), we will pack that single pallet on its own. Stacks (and single pallets) are generated in this manner until pallets from both groups have been exhausted.

During the stacking operation, the greedy algorithm ensures that the required constraints are satisfied. At the point where the top and bottom pallets of a stack have been selected, the algorithm checks to see that (1) the combined weight of both pallets is less than the specified maximum stack weight (this relates to the maximum load a forklift truck can carry), (2) the combined height of the pair is less than the specified maximum stack height (this relates to the stack's stability when it is being moved using a forklift truck), (3) the weight of the top pallet is less than or equal to that of the bottom pallet, and (4) the bottom surface area of the top pallet is less than or equal to that of the bottom pallet. Outside of these constraints, pallets that have been marked as 'fragile' or have customer instructions to not double-stack are both treated as non-stackable pallets.

As an alternative to the greedy approach, we could also have proposed any number of tower building algorithms commonly used in literature. However, due to the constraints we have to deal with, our version of tower building is more simplistic than the problems typically encountered in literature. As we can only stack a single pallet on another pallet and have a maximum of two pallets in a stack, the heuristics often employed in literature, that often attempt to stack multiple boxes on a single box while at the same time building the towers/stacks as high as the container's height, turn out to be overly complicated solutions for our much simpler problem. Using the greedy approach and making a quick comparison with a simple stacking method that builds up stacks in the order pallets are presented, we can see in Figure 3.7 that the proposed approach produces a better overall solution as it is able to generate stacks greedily in a manner that ensures that the surface area covered by the resulting stack footprint is minimised, hence increasing the chances that the set of stacks produced from the approach would fit completely within a container at a later packing phase.



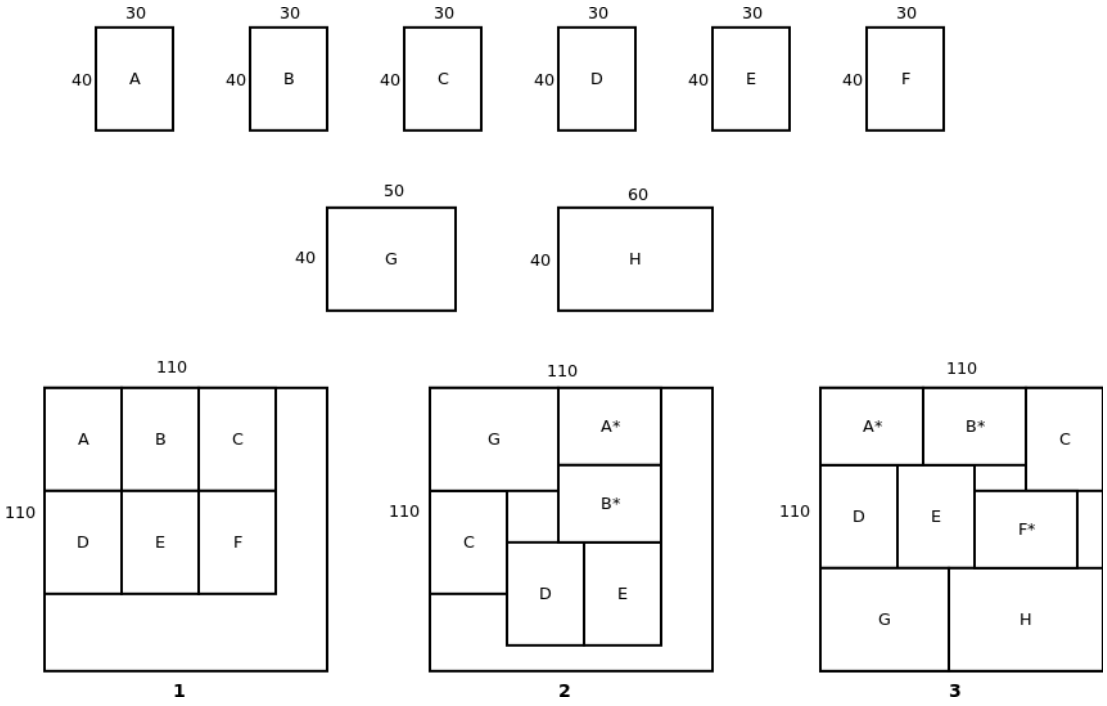
Stackable pallets are marked with a 'S' and non-stackable pallets are marked with a 'NS'.

Figure 3.7: Various stacking variations obtained using different stacking order

3.5.3 The Packing Problem

The packing problem identified in the UKDC is the problem of packing the resulting set of pallet stacks obtained from the prior stacking process completely into a container. When the pallet stacks are physically packed into a container, they are placed orthogonally to the enclosing walls. Forklift trucks are used to pack and arrange the stacks on the container floor. The forklift trucks are able to pick up and move pallet stacks by inserting their forks into slots that can be found on the pallet bases of the pallets. Some pallets have these slots in their pallet bases on only two (opposite) sides, while others have the slots on all sides. This allows pallet stacks to be lifted and packed in either only a single orientation or in any of two possible orientations (allowing rotations by 90°), both determined by the presence and position of the slots on their pallet bases. In the packing problem, experienced warehouse operatives are tasked with packing the set of pallet stacks, obtained from the stacking process, completely into a container. They rely on their experience to determine the order and orientation with which to place each pallet stack into the container. The specific order with which the different stacks are packed as well as the orientation each individual stack is placed in has an impact on if the final arrangement of stacks is able to fit in completely into a container (see Figure 3.8). The complexity arises from the fact that the pallet stacks are placed on several different types of pallet bases, with different sizes and different possible orientations. The loaders are able to deal with this using mostly past experience and their intuition.

This problem is equivalent to a two-dimensional rectangle packing problem (2D-RPP), which is the problem of packing a given set of small rectangular pieces into a larger containing rectangle. There are many variants of the 2D-RPP [Dowsland and Dowsland, 1992]. Majority assume rectangles have a single fixed orientation and allow no rotations during rectangle placement. Others allow rectangle rotations by 90° thus allowing placement in either of the two possible orientations. Some additionally may or may not impose that the small rectangle pieces are obtained through a sequence of edge-to-edge cuts parallel to the edges of the larger containing rectangle (e.g. Lodi et al. [1999]); this is generally referred to as a guillotine constraint in literature. In proposing a solution to the packing



Order for Packing 1: A B C D E F G H
Order for Packing 2: G A* B* C D E F H
Order for Packing 3: A* B* C D E F* G H
An asterisk (*) next to a rectangle means the rectangle is rotated and placed in its second orientation.

Figure 3.8: Packings obtained by the Rectangle Packing Algorithm for 8 rectangles based on the order and orientation of rectangles

problem, we concern ourselves only with the variant of the 2D-RPP that is the most similar to our own problem: i.e. the variant that completely packs a given set of rectangles orthogonally into a single larger rectangle, with the rectangles placed in one or more possible orientations.

An order-based genetic algorithm (GA) integrated with a rectangle packing algorithm was initially proposed for solving the problem. In its operation, the GA takes as its input the list of stacks from the stacking process, and generates several lists of varying order with each stack setup for packing in a randomised orientation. Each list is then packed in its prescribed order using the integrated rectangle packing algorithm. The order is important because the results obtained from the rectangle packing algorithm employed vary based on the order (and orientation) of the rectangles it is given to pack (illustrated in Figure 3.8). The algorithm stops when it either encounters a list that it is able to pack completely into a container or when it has gone through all the lists produced without finding such a list. The GA was proposed to solve this problem because it is a well-known solution approach to the Travelling Salesman Problem which is an order-based problem. It was also selected because of the earlier familiarity with GA operations gained when using them to solve the prior selection problem. The packing GA, in its operation, satisfies the constraint that allows boxes to be packed in any allowed orientation. If a box is allowed to rotate, it can be packed in any of two given orientations. If not, the box can only be packed in a single orientation. The packing GA also satisfies the constraints of packing boxes orthogonally into a container and of packing boxes completely within the walls of a container.

Subsequently, as part of the effort to improve the quality of layouts produced, an entropy-guided Monte Carlo tree search (MCTS) process was proposed for solving the same problem. The process employs a heuristic placement method that places pallets randomly in the spare space in a container and then moves them towards the already filled area of the container. The pallets are moved in a manner that aligns them as close as possible to previously loaded pallets. The MCTS process along with its placement method helps to facilitate the efficient utilisation of space within the container. Guiding the search process using entropy helps to reduce the level of disorderliness in the layouts resulting in layouts that are easier to understand. In its operation, the entropy-guided MCTS process

satisfies all the same constraints satisfied by the earlier proposed Packing GA.

3.6 Conclusion

The container loading problem (CLP) experienced in the UKDC differs from the canonical CLP commonly described and dealt with in literature. This is usually expected when dealing with real-world problems in practice. In solving the problem, the UKDC usually made use of a manual system for selecting and loading pallets into containers. This system relied on the expertise provided by a handful of experienced warehouse operatives. Earlier attempts had been made to automate this manual loading process. The attempts included the use of off-the-shelf and bespoke software solutions. Due to the unique nature of the problem, which is a result of the specific combination of practical constraints being considered, it was difficult to obtain solutions from these attempts as there was a lot of manual tuning and configuration required to accommodate and capture all the required constraints. In addition to this, solutions that were eventually obtained were practically unfeasible for loading in the real world. An example of this was the production of plans that were too complex to be loaded by forklift trucks. In another example, the way the pallets were stacked would cause damage to the boxes if loaded in practice. In other instances, some loading plans that looked almost feasible were generated and could be loaded in practice with little tweaks to the plan; however, it was always obvious from looking at such plans that a higher container utilisation could be obtained by using the loaders' expertise.

These inadequacies led to the commissioning of the research reported in this thesis. At the start of the research, the exact problem being solved and the manual process employed for solving the problem were both studied and investigated. This study revealed that the problem was different from the canonical form of the problem normally described in the literature and that the problem was made more difficult by the introduction of all the real-world constraints specific to the UKDC, that had to be satisfied for solutions provided to be considered feasible. This explained why the problem had been non-trivial to solve so far. The study of the manual process in addition to the study of relevant literature is what led to the overall hybrid framework provided for the proposed solution approach. In

order to simplify the problem, it was split into different sub-problems, i.e. the sub-problems of selection, stacking, and packing, that could each be solved independently of the other sub-problems and later combined together to provide a solution to the entire problem. This approach was inspired by the manual process and forms the foundation that all subsequent solution approaches presented in this thesis are built on. Once the sub-problems were identified, solving the overall problem became a matter of identifying and implementing known solution approaches to each of the sub-problems. While the whole problem being solved and all its combined constraints might have looked like a 'new' problem with no known solutions, each of the sub-problems had known solution approaches and had a smaller number of constraints to deal with individually.

Chapter 4

A Hybrid Algorithm for the Container Loading Problem

4.1 Introduction

In the previous chapter, we proposed a solution approach to the CLP experienced in the UKDC that uses a hybrid algorithm to solve the problem by first separating the problem into sub-problems, which are solved individually using different specialised algorithms and collaboratively combined to solve the entire problem while satisfying all of the relevant practical constraints encountered at each sub-problem stage. In this chapter, we present the proposed hybrid algorithm and show how the sub-algorithms are collaboratively combined to solve the entire problem. It is worth mentioning again that the hybrid algorithm is designed in a manner that allows each of the sub-algorithms to be easily interchanged with an equivalent algorithm. We refer the reader back to the previous chapter for the justifications we made for the choice of the individual sub-algorithms presented in this chapter.

The hybrid algorithm at its core is an iterative procedure that combines three different heuristic algorithms: i.e. a selection algorithm, a stacking algorithm, and a packing algorithm. As mentioned in the previous chapter, the selection algorithm solves a one-dimensional knapsack problem by selecting a combination of jobs (groups of pallets) for loading into a container, while maximizing the

Algorithm 4.1 Hybrid Algorithm

Input:
UnpackedPallets
L_c, B_c, W_{max} ▷ container length, breadth, and maximum weight
W_{st}, H_{st} ▷ maximum stack weight and height
Output:
PackedStacks

group *UnpackedPallets* into *PalletJobs* ▷ each job is a group of pallets
JobSelections ← SelectionAlgorithm(*PalletJobs*, W_{max})
for all *selection* ∈ *JobSelections* **do**
  generate list of pallets, *Pallets* from selection
  Stacks ← StackingAlgorithm(*Pallets*, W_{st}, H_{st})
  PackedStacks ← PackingAlgorithm(*Stacks*, L_c, B_c)
  if AllStacksFit(*PackedStacks*) **then**
    return *PackedStacks*
  end if
end for

weight capacity of the container; the stacking algorithm stacks a given list of pallets by sorting the selection of pallets obtained from the selection algorithm in descending order of weight and iteratively stacking pallets on top of each other in a greedy manner whilst keeping the stack height and weight below the specified maximum values; and the packing algorithm solves a two-dimensional rectangle packing problem by packing the list of stacks produced by the stacking algorithm into a container, and checking to see if the stacks are packed completely or not.

Algorithm 4.1 shows the operation of the hybrid algorithm. It begins by first preparing the required input data and running the selection algorithm. The resulting list of selections produced is iterated through, and for each selection, a list containing all the selected pallets is produced. This list of pallets is provided as input to the stacking algorithm which produces a list of stacks as output. The packing algorithm then takes as its input this list of stacks and packs the stacks in the container reporting if they could be completely packed or not. This iteration in the hybrid algorithm, through the list of selections, continues until the packing algorithm finds a list of stacks that can be packed completely in the container, or until all the job selections have been evaluated, at which point the

algorithm reports that no selection of pallets could be found that fits completely in the container. If a selection of pallets that fits completely is found, all of the identified practical constraints mentioned in Chapter 3 will have been satisfied.

4.2 The Selection Algorithm

The selection problem, in the previous chapter, was identified as being equivalent to the well-known one-dimensional knapsack problem, and a genetic algorithm (GA) was proposed to solve it. The proposed GA (see Algorithm 4.2) is a standard GA implementation with standard chromosome representation and genetic operators i.e. selection, crossover, mutation. As GAs are well known and widely used across literature, we will not discuss in detail its general implementation and operation; we instead refer the reader to Goldberg [1989] and Hopgood [2001] where such detail is covered extensively.

In the selection GA, a candidate solution is represented as a chromosome which is encoded as an array of bits. The length of the array is equal to the total number of available jobs. Each bit in the array represents a unique job and can have its value set to 0 or 1, representing whether a job has been selected for inclusion into a container (i.e., the bit value is set to 1), or not (i.e., the bit value is set to 0). Each job represents a logical group of pallets that must all be completely packed together in the same container. The quality of a chromosome or candidate solution, known as its fitness, is the total sum of weights for all the jobs selected for inclusion in the container. The weight of each job is, in turn, the total sum of weights for every pallet that belongs to that particular job. If the total weight of all the selected jobs in a candidate solution exceeds the container's maximum weight capacity, a penalty is applied and the fitness of the candidate solution is set to zero.

The selection GA begins its operation by generating an initial population of candidate solutions where each bit of the chromosome is randomly set to 0 or 1. The fitness of every chromosome in the population is then calculated, and the entire population is sorted according to fitness. This population is known as the first generation. To create a new population which will be known as the second generation, the GA concept of 'elitism' is first applied. This simply makes a copy

of the chromosome with highest fitness from the last generation and includes it as the first member of the new population. Two chromosomes are then selected from the previous population using 'tournament selection', which is one of several GA selection operators where a number of chromosomes are chosen at random from the population and placed in tournaments against each other. The winner of each 'tournament' (i.e. the chromosome with the best fitness) is selected for crossover. Based on a crossover probability, these two chromosomes, known as parent chromosomes, are either mated or cloned to produce two children chromosomes. The mating process is a standard one-point crossover operation which involves selecting a random point in both chromosomes and swapping their end parts. The cloning process simply creates an exact copy of a chromosome. Based on a mutation probability, the resulting children are mutated. In a mutation operation, a random position in the chromosome is selected, and its bit is flipped i.e. if the bit is 0, it is changed to 1, or vice versa. This process of selection, reproduction and mutation is repeated until the number of chromosomes in the current generation matches the population size of the previous generation, at which point the creation of the new generation is complete. Subsequent generations are created using the same procedure until a specified number of generations have been produced.

The output of the selection algorithm is the population of chromosomes in the last generation. Each chromosome in this population is a candidate solution that represents a varied combination of jobs selected for inclusion into a container. The selection of jobs identified in each chromosome satisfies the practical constraints identified for the selection problem in the previous chapter, i.e., the total number of pallets resulting from the selected jobs have a total combined weight less than the maximum weight capacity of the container. The chromosomes in the last population are sorted in descending order by fitness to ensure that when the hybrid algorithm is iterating through the population, it will stack and pack job selections with a higher fitness first. As the fitness measure is the total sum of weights for every selected pallet, when a candidate solution is found that can be successfully stacked and packed, we can be sure that we have packed a solution that provides the maximum weight utility for the container, and that no other solution exists in the population with a higher container weight utility.

Algorithm 4.2 The Selection Algorithm

Input:
PalletJobs ▷ list of pallet groups
W_{max} ▷ maximum container weight
Output:
JobSelections ▷ selections of pallet groups

```
InitialisePopulation(PalletJobs)
EvaluatePopulation(W_max)
while max generation count not reached do
    create new population
    apply elitism
    while total population count not reached do
        Parent_1, Parent_2 ← TournamentSelection(TournamentSize)
        Child_1, Child_2 ← Crossover(Parent_1, Parent_2)
        Mutate(Child_1)
        Mutate(Child_2)
        add Child_1 and Child_2 to new population
    end while
    EvaluateNewPopulation(W_max)
end while
JobSelections ← most recent population
return JobSelections
end

function EvaluatePopulation(W_max)
    for all individual ∈ population do
        individual.fitness ← 0
        for all bit ∈ individual do
            if bit == 1 then
                W_job ← weight of job at bit position
                individual.fitness ← individual.fitness + W_job
            end if
        end for
        if individual.fitness > W_max then
            individual.fitness ← 0          ▷ penalty for greater weights
        end if
    end for
end function
```

```
function TournamentSelection(n)                    ▷ n is the tournament size
    select n individuals from the population at random
    bestIndividual ← select individual with best fitness
    return bestIndividual
end function

function Crossover(Parent_1, Parent_2)  ▷ a parent is a list of pallet groups
    rand ← select random floating point number between 0 and 1
    if rand < crossoverProbability then
        crossoverPoint ← select a random pallet group's position
        create Child_1, Child_2 as two empty list of pallet groups
        for each pallet group in Parent do      ▷ Parent can be either parent
            position ← position of the pallet group
            if position < crossoverPoint then
                copy the value at position in Parent_1 to Child_1
                copy the value at position in Parent_2 to Child_2
            else
                copy the value at position in Parent_1 to Child_2
                copy the value at position in Parent_2 to Child_1
            end if
        end for
    end if
end function

function Mutate(Child)
    rand ← select random floating point number between 0 and 1
    if rand < mutationProbability then
        pos ← select random position in Child
        invert the value at position pos in Child          ▷ i.e. 0 → 1; 1 → 0
    end if
end function
```

4.3 The Stacking Algorithm

In the stacking problem, we are given a list of pallets to stack subject to several identified practical constraints. The proposed algorithm for this problem is a greedy algorithm. The greedy algorithm (see Algorithm 4.3) begins its operation by separating the given list of pallets into two categories: stackable and non-stackable. The pallets in each category are then sorted in descending order of

weight. A *stack* data structure is defined that can hold two pallets: a required *bottom* pallet, and an optional *top* pallet. This data structure is defined for convenience in the operation of the stacking process and also to help satisfy the constraint that requires that a stack of pallets should not contain more than two pallets.

The algorithm then proceeds to iteratively generate stacks by attempting to use stackable pallets before non-stackable pallets as bottom pallets and non-stackable pallets before stackable pallets as top pallets. While searching for a bottom pallet, stackable pallets are considered first so that there is room for another pallet to be placed as a top pallet. Since the list of stackable pallets is sorted by weight in descending order, heavier pallets are guaranteed to be selected before lighter ones. This way, if another pallet in the same list is selected as a top pallet, we are certain that it is not heavier than the bottom pallet. If a pallet is selected from the list of non-stackable pallets as a bottom pallet, no other pallet can be placed on it. While searching for a top pallet, non-stackable pallets are considered first to ensure that a stackable pallet that can be used as a bottom pallet is not wasted as a top pallet. If a pallet is found that satisfies all the required practical constraints, it is selected as a top pallet; otherwise, other pallets in the list are considered until a suitable pallet is found. If none is found, the algorithm proceeds to check the list of stackable pallets for a suitable pallet.

If a bottom and top pallet can be selected for a stack, the stack is complete and added to a stack list. If only a bottom pallet can be selected for a stack, with no suitable top pallet found, the stack is considered complete and added to the stack list. Stacks are generated and added to the stack list until there are no more pallets to be considered for stacking.

The output of the stacking algorithm is the resulting list of stacks. The greedy approach used for the stacking algorithm ensures that the total number of stacks generated is minimized. This is important because a lower number of stacks increases the chances of the entire stack list being packed completely onto the container floor.

Algorithm 4.3 The Stacking Algorithm

Input:
Pallets ▷ pallets from an individual selection
W_{st}, H_{st} ▷ maximum stack weight and height
Output:
Stacks

```
Stackable, NonStackable ← Split(Pallets)
Sort(Stackable)                          ▷ in descending order by weight
Sort(NonStackable)                       ▷ in descending order by weight
initialise StackList
while Stackable has pallets or NonStackable has pallets do
    create Stack
    if Stackable has pallets then
        select first pallet in Stackable
        remove selected pallet from Stackable
    else if NonStackable has pallets then
        select first pallet in NonStackable
        remove selected pallet from NonStackable
    end if
    bottomPallet ← selected pallet
    add bottomPallet to Stack

    if bottomPallet is stackable then
        topFound ← false
        for all pallet ∈ NonStackable do
            topPallet ← pallet
            satisfied ← CheckConstraints(bottomPallet, topPallet)
            if satisfied == true then
                topFound ← true
                topPallet ← selected pallet
                remove topPallet from NonStackable
                add topPallet to Stack
                break
            end if
        end for

        if topFound ≠ true then
            for all pallet ∈ Stackable do
                topPallet ← pallet
                satisfied ← CheckConstraints(bottomPallet, topPallet)
```

```
                if satisfied == true then
                    topPallet ← selected pallet
                    remove topPallet from Stackable
                    add topPallet to Stack
                    break
                end if
            end for
        end if
    end if
    add Stack to StackList
end while
return StackList
```

4.4 The Packing Algorithm

In the packing problem described in the previous chapter, we are faced with the problem of packing the set of pallet stacks obtained from the prior stacking process completely into a container. This problem was identified as being equivalent to a two-dimensional rectangle packing problem and an order-based genetic algorithm (GA) integrated with a rectangle packing algorithm was proposed for solving it.

The proposed GA's overall operation is similar to that of a standard GA, with some minor differences in chromosome representation and hence the implementation of the genetic operators. Candidate solutions are represented by chromosomes encoded as an order-based list of pallet stacks, with each pallet stack having an orientation property (0 or 1) that determines what orientation the stack will be packed in.

The GA's initial population is generated by randomly assigning the orientation of each stack in a given stack list to 0 or 1, and shuffling the order of the stacks in the list. The fitness of a chromosome is evaluated using the integrated rectangle packing algorithm which sets the fitness to the total number of stacks in the stack list that it is able to completely pack into the container. During the fitness evaluation process (see Algorithm 4.4), the algorithm checks if each stack (more precisely the bottom pallet of each stack) can be rotated. If the stack can be rotated, it is packed with the rectangle packing algorithm in the orientation assigned to the stack in the chromosome; if not, it is packed in the

stack's only acceptable orientation irrespective of the orientation assigned to it. In dealing with the rotations this way, we satisfy the fourth constraint identified and considered in Chapter 1:

- Boxes can either be rotated or not. If a box is allowed to rotate, it can only be placed in a container in any of two given orientations; otherwise, it can only be placed in a single orientation.

When generating subsequent populations, the 'random' selection method is used for selecting chromosomes for reproduction. We use random selection instead of other GA selection mechanisms that tend to explicitly favour to some degree the selection of the fittest members of the population (e.g. tournament or roulette selection) because at this point the only fitness measure available to us is a binary measure that indicates if a given stack list can be completely packed into a container or not; and if the stacks can be packed completely, the entire GA process is stopped as a solution would have been found. This type of measure is not a good selection criterion; hence, the 'random' selection method, which does not rely on using any measure as a selection criterion, was selected.

During reproduction, after two parent chromosomes are selected, we employ a one-point crossover operation, which selects a random point in both chromosomes and swaps the orientations of the stacks in their end parts. The resulting children chromosomes obtained from the swaps make up the next population. As in the standard GA, reproduction via crossover is subject to a crossover probability. In the mutation operation employed, the order of the pallet stacks that comprise a chromosome is shuffled. As with crossover, this mutation is also subject to a mutation probability. In the overall operation of the algorithm, if a chromosome is evaluated and found to have a fitness equal to the total number of stacks in the chromosome (i.e. all the stacks in the chromosome can be packed completely), the algorithm terminates as a solution has been found; otherwise, the algorithm runs till it reaches the maximum number of generations and terminates indicating that no solution could be found that could be packed completely into the container for this particular selection of stacks.

Algorithm 4.4 Packing Algorithm - Fitness Evaluation

Input:
StackList ▷ i.e. candidate solution / chromosome
Output:
fitness

```
fitness ← 0
for all stack ∈ StackList do
    if stack can rotate then
        stack.orientation ← suppliedOrientation
    else
        stack.orientation ← defaultOrientation
    end if

    canPack ← TryPack(stack)

    if canPack == true then
        fitness ← fitness + 1
    end if
end for
return fitness
```

4.4.1 The Simple Rectangle Packer

The rectangle packing algorithm employed as the integrated packer in the Packing algorithm is called the Simple Rectangle Packer. It corresponds to the 'Shelf Next Fit' algorithm in Jylänki [2010] and to the 'Next-Fit Decreasing Height' algorithm in Lodi et al. [2002] without the initial step that sorts items by non-increasing height. It is a simple algorithm optimised for runtime performance. It does sacrifice a bit of space efficiency for its high performance and low memory usage, but the time needed to pack a new rectangle is $\mathcal{O}(1)$ [Markus Ewald, 2009]. It achieves good results with near-uniform sized rectangles but will waste lots of space with rectangles of varying dimensions. As our pallet set is weakly heterogeneous (see Section 3.1), we selected this algorithm for its high performance expecting it to achieve good results when used as the integrated packer.

In its operation, the algorithm begins with a new row with height set to 0. For each rectangle it has to pack, it adds the rectangle to the current row

from left to right. If the rectangle's height is greater than the row's height and the entire packing area height has not been exceeded, the row's height is set to the rectangle's height. If the rectangle does not fit in the row (i.e. adding the rectangle makes it exceed the packing area width), the row is closed and another row is created. The process then repeats, adding rectangles to the row from left to right. When the packing area's height is exceeded, the algorithm is no longer able to pack any rectangles. The algorithm is presented in Algorithm 4.5. Its implementation in code can be found in Markus Ewald [2011a].

4.5 Experiments and Results

The performance of the hybrid algorithm described in this chapter was evaluated using historical anonymised data via computational experiments performed on a PC with an Intel Core i3 M330 CPU (2.13GHz) and 4GB RAM running the ArchLinux operating system. The data contained various records of pallet and container data, and the weight utilisation obtained when experienced warehouse operatives manually loaded the recorded pallets into the given container.

The data was presented as 15 problem sets consisting on average of 22 jobs and 331 pallets, with each problem set representing a specific instance of the packing problem as experienced by the warehouse operatives. A summary of the data is presented in Table 4.1 and each of the problem sets is presented in the tables in Appendix C: Hybrid Algorithm Problem Sets. The pallets are weakly heterogeneous with only 4 different types of pallets (see Table 3.2) used across all the problem sets. For each pallet in the problem set, weight, length, breadth and height data is provided. The data provided, however, did not consider the stacking constraint. As such, during the algorithm evaluation, all pallets in the problem sets are assumed to be stackable. In order to provide further comparison that would better reflect behaviour observed in real-world container loading, the problem set was extended to consider the stacking constraint by randomly generating and assigning the stackability constraint to pallets i.e. some pallets would be considered stackable, and would be able to have other pallets placed on them, while others would not. The resulting problem set, extended to consider the stacking constraints, is referred to as the 'extended' problem set, while the

Algorithm 4.5 The Simple Rectangle Packer

Input:
stack
Output:
canPack ▷ indicate if the stack can be packed or not
placement ▷ stack's placement point in the packing area

initialise *PackingArea*
$currentRow \leftarrow 0$
$rowHeight \leftarrow 0$
$usedRowWidth \leftarrow 0$

function TryPack(*stack*)
$canPack \leftarrow false$
$placement \leftarrow InvalidPlacementPoint$

▷ rectangle won't fit if it is larger than packing area
if $stack.width > PackingAreaWidth$ **or**
$stack.height > PackingAreaHeight$ **then**
$placement \leftarrow InvalidPlacementPoint$
$canPack \leftarrow false$
return *canPack*, *placement*
end if

▷ if packing area width is exceeded, start a new row
if $usedRowWidth + stack.width > PackingAreaWidth$ **then**
currentRow += rowHeight
$rowHeight \leftarrow 0$
$usedRowWidth \leftarrow 0$
end if

▷ if stack can't fit vertically, packing area is full
if $currentRow + stack.height > PackingAreaHeight$ **then**
$placement \leftarrow InvalidPlacementPoint$
$canPack \leftarrow false$
return *canPack*, *placement*
end if

▷ stack fits in current location
$placement \leftarrow Point(usedRowWidth, currentRow)$

usedRowWidth += $stack.width$

```
    if stack.height > rowHeight then
        rowHeight = stack.height
    end if

    canPack ← true
    return canPack, placement
end function
```

Table 4.1: Problem set summary

Problem set	Jobs	Pallets	Total weight (kg)
PS01	37	390	173947
PS02	35	299	118168
PS03	29	332	147675
PS04	25	279	121957
PS05	22	251	97070
PS06	25	298	121574
PS07	24	236	92502
PS08	18	181	71011
PS09	21	215	96328
PS10	19	173	66698
PS11	8	113	45081
PS12	17	161	71704
PS13	14	103	46892
PS14	13	116	44459
PS15	24	335	138405

original unmodified problem set is referred to as the 'normal' problem set.

The hybrid algorithm is run for 50 iterations across all the problem sets, and the best, average and worst-case computation time and container weight utilisation achieved across all iterations is computed and compared. The total computation time to run all the iterations is also computed. Solutions obtained from the hybrid algorithm were validated by experienced warehouse operatives, who would compare the presented solutions with past documented solutions, and would also draw from experience to work out if a provided solution was feasible or not. A summary of 50 of such validated solutions, each having a weight utilisation of 100% is presented in Appendix B: Verified Hybrid Algorithm solutions.

Results obtained from evaluating the algorithm on the problem sets in their normal and extended form are compared with results from the manual process employed at NSK. The comparisons made are based on the weight utilisation of the packings obtained by both approaches. The historical data provided did not capture the time it took the warehouse operatives to complete their loading operations; as such, we have assumed an approximate value of 5 minutes - which represented the fastest recorded time it took an experienced warehouse operative to work out a selection of pallets and verify that the selection fits.

4.5.1 Parameter Tuning

The selection and packing GAs have several parameters that control their performance and operation. As these parameters can take on a wide range of values, with the values having a direct impact on the performance/computational time of the GA, we performed experiments to obtain parameters that achieved a good balance of optimal results and computation time. From the experiments performed, it was found that increasing the generation size of the selection GA beyond 500 had a very little effect on the performance of the algorithm, but caused a linear increase in the computation time. A population size of 100 was observed to give optimal results on average but resulted in low performance for large problem sets. A population size of 200 is thus selected so that uniform performance is obtained across large and small problem sets. Increasing the population size beyond 200 had little or no effect on performance, and only resulted in an increase to computation time. After experimenting with values between 0 and 1, in steps of 0.1, for the crossover probability, and in steps of 0.05 for the mutation probability, it was found that the values 0.8 and 0.05 for the crossover and mutation probabilities respectively, provided a good balance between optimal results and computational time. Similar experiments were performed on the packing genetic algorithm and recommended parameters obtained.

Based on the experiments performed, the parameters for the crossover probability, mutation probability, population size and generation size of the selection GA were set to 0.8, 0.05, 200 and 500 respectively. Similarly, the parameters for the crossover probability, mutation probability, population size and generation

Table 4.2: Recommended genetic algorithm parameters

	Selection GA	Packing GA
Crossover probability	0.8	0.8
Mutation probability	0.05	0.2
Population size	200	100
Generation size	500	300

size of the packing GA were set to 0.8, 0.2, 100 and 300 respectively. These parameters, seen in Table 4.2, are used subsequently for evaluating the hybrid algorithm across all the problem sets.

4.5.2 Results and Comparisons

The results of evaluating the hybrid algorithm on the normal problem set, shown in Table 4.3, show that the hybrid algorithm achieves a higher weight utilisation than the manual process across all the problem sets. Even the worst-case solutions obtained from the algorithm are consistently better than the solutions from the manual process. The solutions are also obtained in reasonable time, with the worst-case recorded time of 21.91 seconds.

The results of evaluating the algorithm on the extended problem set, which considers and enforces the stacking constraint, can be seen in Table 4.4. Due to the introduction of the stacking constraint, a general increase in computation time is observed. The weight utilisation achieved by the algorithm is similar to that achieved on the normal problem sets in the best-case scenario, slightly lower in the average-case scenario, and significantly lower in the worst-case scenario. In comparison to the manual loading process, the algorithm performs better in the average and best case scenarios; but achieves lower weight utilisation for some of the problem sets in the worst-case scenario. The computation time achieved for the average and worst-case scenario is significantly higher than those obtained for the normal problem sets. This is expected as the introduction of the stacking constraint will require more computation to be performed in the Stacking algorithm.

Overall, on average, the hybrid algorithm evaluated on the extended prob-

Table 4.3: Results for the normal problem sets

Problem sets	Manual process		Hybrid algorithm					
	Weight utilisation (%)	Best Recorded Time (s)	Weight utilisation (%)			Time (s)		
			Worst	Average	Best	Worst	Average	Best
PS1	98.15	300	100	100	100	0.53	0.49	0.47
PS2	98.72	300	99.94	99.99	100	11.72	0.7	0.46
PS3	98.92	300	100	100	100	0.47	0.42	0.4
PS4	98.58	300	100	100	100	0.42	0.38	0.38
PS5	98.68	300	100	100	100	0.49	0.37	0.36
PS6	95.8	300	99.71	99.99	100	11.29	0.85	0.37
PS7	99.25	300	100	100	100	0.46	0.4	0.38
PS8	99.73	300	100	100	100	0.37	0.35	0.33
PS9	94.71	300	100	100	100	0.39	0.37	0.36
PS10	85.54	300	100	100	100	0.42	0.35	0.33
PS11	89.16	300	99.76	99.76	99.76	0.35	0.28	0.26
PS12	95.43	300	100	100	100	0.36	0.33	0.32
PS13	94.88	300	99.99	99.99	99.99	0.34	0.31	0.31
PS14	81.6	300	99.99	99.99	99.99	0.34	0.31	0.3
PS15	96.52	300	98.75	99.96	100	21.91	2.1	0.37
Averages:	95.04	300	99.88	99.98	99.98	3.32	0.53	0.36

Table 4.4: Results for the extended problem sets

Problem sets	Manual process		Hybrid algorithm					
	Weight utilisation (%)	Best Recorded Time (s)	Weight utilisation (%)			Time (s)		
			Worst	Average	Best	Worst	Average	Best
PS1	98.15	300	99.99	100	100	1.35	0.34	0.28
PS2	98.72	300	94.36	99.33	100	48.17	25.28	0.28
PS3	98.92	300	99.99	100	100	0.29	0.27	0.25
PS4	98.58	300	99.22	99.98	100	97.4	2.29	0.23
PS5	98.68	300	90.42	99.45	100	226.86	23.4	0.24
PS6	95.8	300	78.29	99.07	100	175.78	29.1	0.24
PS7	99.25	300	94.34	99.2	100	158.73	38.36	0.24
PS8	99.73	300	78	96.55	100	293.53	78.59	0.21
PS9	94.71	300	99.93	99.99	100	177.86	3.79	0.22
PS10	85.54	300	93.17	99.25	100	301.34	75.7	0.22
PS11	89.16	300	99.76	99.76	99.76	1.86	0.67	0.22
PS12	95.43	300	99.91	99.97	100	0.25	0.22	0.21
PS13	94.88	300	99.77	99.96	99.99	0.23	0.21	0.2
PS14	81.6	300	96.28	99.63	99.95	345.56	33.34	0.19
PS15	96.52	300	89.6	99.07	100	154.56	39.06	0.23
Averages:	95.04	300	94.2	99.41	99.98	132.25	23.37	0.23

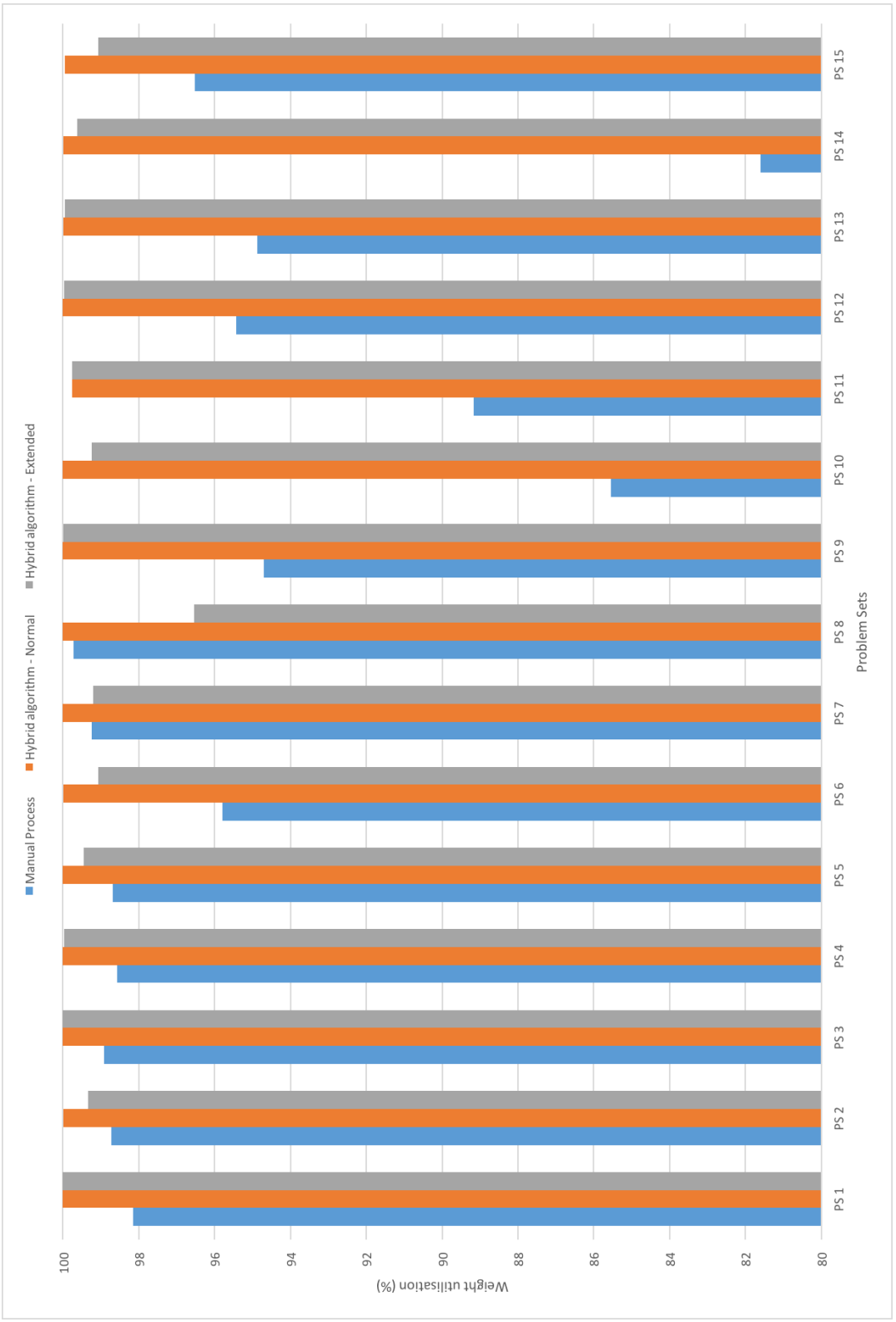


Figure 4.1: Comparison of weight utilisation across all problem sets

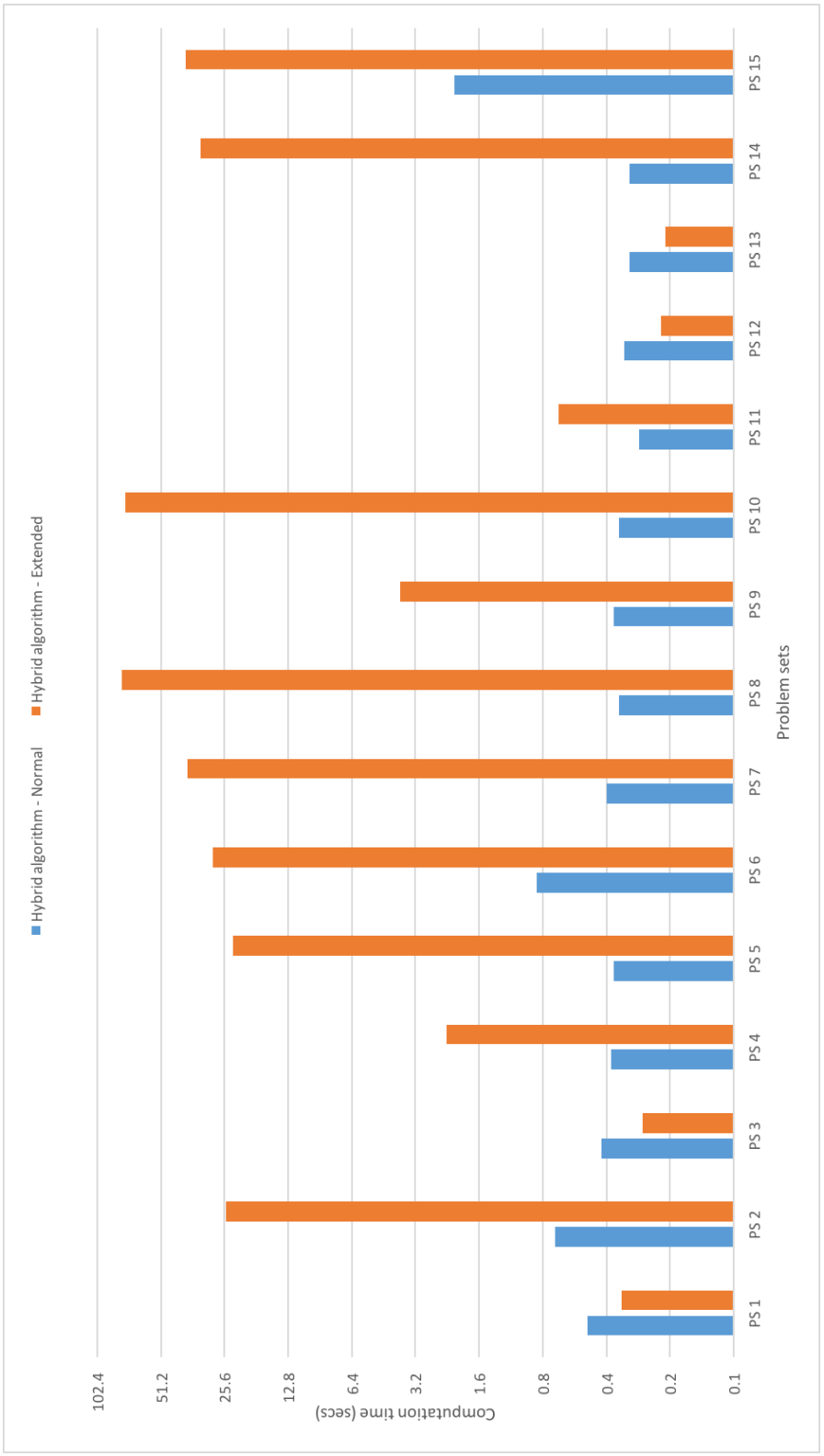


Figure 4.2: Comparison of computation time across all problem sets

lem set outperforms the manual loading method with average worst-case weight utilisation and computation time of 99.88% and 132.25 seconds respectively.

4.6 Conclusion

In the previous chapter, an approach to solving the Container Loading Problem (CLP) was proposed that divided the CLP into several sub-problems and solved each sub-problem individually in order to solve the problem as a whole. In this chapter, a hybrid algorithm was devised as a combination of several different independent algorithms that each solve one of the sub-problems, while satisfying a number of different practical constraints relevant to each sub-problem. The devised algorithm comprised of a genetic algorithm to solve a 'selection' problem; a greedy algorithm to solve a 'stacking' problem; and a genetic algorithm integrated with a rectangle packing algorithm to solve a 'packing' problem.

The algorithm was tested using problem sets made up of real-world historical data and the results showed that the algorithm achieved an average weight utilisation of 99.98% in 0.53 seconds on the examined problem sets. As the data examined did not consider stacking constraints, the initial algorithm runs assumed all pallets were stackable. To better reflect real-world loading, the problem sets were extended to include stacking constraints. With the introduction of a stacking constraint, the algorithm achieved an average weight utilisation of 99.41% in 23.37 seconds.

When the stacking constraint was introduced, the results obtained showed a slight reduction in the weight utilisation achieved with a noticeable increase in computation time. This performance was still deemed acceptable and practical, with the best- and average-case performance producing better results than the manual process and the worst-case performance producing results as good as those obtained by the manual process. In both cases, with and without the consideration of the stacking constraint, the results were very promising as all the solutions obtained were found to be feasible to load in a real-world scenario, and were computed in a reasonable time.

The quality of the weight utilisation and computation time obtained did not seem to be affected by the characteristics of the problem set. Performance did not

vary across problem sets with a high or low number of jobs/pallets. We did find that for some of the smaller problem sets, the weight utilisation obtained, e.g. 99.76% for problem set 11, while not the maximum possible for the container i.e. 100%, was the best possible utilisation that could be obtained for the combination of jobs and pallets considered.

In summary, the results obtained from the experiments performed on all the problem sets show that the proposed approach is valid, runs in reasonable time and produces better quality results than the existing manual process.

Chapter 5

Improvements to the Packing Algorithm

5.1 Introduction

In this chapter, we present improvements to the hybrid algorithm derived in Chapter 4. The improvements are made particularly to the packing algorithm (see section 4.4) employed for solving the packing sub-problem, in order to increase overall packing efficiency. You may recall that the hybrid algorithm is modular in design and composed of several algorithms, each employed to solve an individual sub-problem. This allows for the easy exchange of any of the algorithms with equivalent algorithms that can solve a particular sub-problem while satisfying all required constraints. The initial steps taken towards improvement therefore involved examining other equivalent algorithms for solving the packing sub-problem. As a start, instead of replacing the entire packing algorithm i.e. the packing genetic algorithm (GA) and its integrated rectangle packing algorithm, we replace only the GA's integrated rectangle packing algorithm. The new rectangle packing algorithm employed is called the Cygon Rectangle Packer. It replaces the Simple Rectangle Packer in the Packing GA. Based on observations of the operation and results of the modified Packing GA, the entire Packing GA is subsequently replaced with a simpler 'Sort-and-Pack' algorithm that uses the Cygon packer for its 'Pack' step.

5.2 The Cygon Rectangle Packing Algorithm

The Cygon Rectangle Packer is the integrated rectangle packing algorithm employed as a replacement for the Simple Rectangle Packer in the Packing GA. It is named after its author, Markus 'Cygon' Ewald and was selected because it is efficient in its space usage and offers good performance. It never exceeds $\mathcal{O}(n)$ time but generally achieves almost $\mathcal{O}(1)$ on average thus providing a very good compromise between space-efficiency and time-efficiency [see Markus Ewald, 2009].

In its operation, the packing algorithm always places rectangles as low as possible in the packing area. So, for any new rectangle that is to be added, it has to determine the X coordinate at which the rectangle can have the lowest overall height without intersecting any other rectangles. To quickly discover these locations, the algorithm uses a sophisticated data structure that stores the upper silhouette of the packing area. When a new rectangle needs to be added, only the silhouette edges need to be analysed to find the position where the rectangle would achieve the lowest placement possible in the packing area. The algorithm is presented in Algorithm 5.1 and its implementation in code can be found in Markus Ewald [2011a].

5.3 The Cygon Packer integrated Genetic Algorithm

As mentioned earlier, the Cygon rectangle packer is integrated into the Packing GA and used to replace the earlier employed 'Simple rectangle packer'. The new packer can be used in the same manner and handle the exact same inputs and constraints as the packer it is replacing. Because of this 'transparency' between the packers, the Packing GA is able to run and function as usual without any change to its operation. The only change expected is to the results obtained. Once the exchange is made, the Packing GA is able to immediately use the newly integrated packer as part of its evaluation process. The algorithm's operation can be seen in Algorithm 4.4.

Algorithm 5.1 The Cygon Rectangle Packer

```
Input:
  stack
Output:
  canPack                          ▷ indicate if the stack can be packed or not
  placement                        ▷ stack's placement point in the packing area

  initialise PackingArea
  initialise heightSlices          ▷ stores the height silhouette of the rectangles

  heightSlices.add( Point(0, 0) )

  function TryPack(stack)
    canPack ← false
    placement ← InvalidPlacementPoint

                                   ▷ rectangle won't fit if it is larger than packing area
    if stack.width > PackingAreaWidth  or
       stack.height > PackingAreaHeight then
          placement ← InvalidPlacementPoint
          canPack ← false
          return canPack, placement
    end if

    canPack, placement ← GetBestPlacementInPackingArea(stack)

    if (canPack is true) and (placement is valid) then
                                   ▷ mark the region of rectangle as being taken
      UpdateHeightSlices(stack, placement)
    end if
    canPack ← true
       return canPack, placement
  end function

  function GetBestPlacementInPackingArea(stack)
    bestSliceX ← 0
    bestSliceY ← 0

    highest ← heightSlices.getHighest()
    rightmost ← heightSlices.getRightMost()
```

```
    if highest + stack.height ≤ PackingAreaHeight then
        bestSliceY ← highest.Y
    end if

    if rightmost + stack.width ≤ PackingAreaWidth then
        bestSliceX ← rightmost.X
    end if

    while RectangleIntersectsInX(stack, bestSliceX) do
        bestSliceX++
    end while

    while RectangleIntersectsInY(stack, bestSliceY) do
        bestSliceY++
    end while

    if bestSliceX + stack.width ≤ PackingAreaWidth and
       bestSliceY + stack.height ≤ PackingAreaHeight then
            placement ← Point(bestSliceX, bestSliceY)
            canPack ← true
    else
            placement ← InvalidPlacementPoint
            canPack ← false
    end if

    return canPack, placement
end function
```

5.4 The Sort-and-Pack Cygon Packer

While examining the output from the Packing GA integrated with the Cygon packer, we observed that layouts that had larger stacks packed first tended to fit in more stacks into the container. 'Larger' in this context refers not to the weight, but instead to the 'size' or 'surface area' of the bottom part of the stack in contact with the container floor. As a result of this observation, a decision was made to swap out the entire Packing GA with a simpler 'Sort-and-Pack' algorithm, and to observe if this would produce comparable results.

The 'Sort-and-Pack' algorithm in its operation sorts all the stacks that have

Algorithm 5.2 Sort-and-Pack Hybrid Algorithm

Input:
UnpackedPallets
L_c, B_c, W_{max} ▷ container length, breadth, and maximum weight
W_{st}, H_{st} ▷ maximum stack weight and height
Output:
PackedStacks

1: group *UnpackedPallets* into *PalletJobs* ▷ each job is a group of pallets
2: *JobSelections* ← SelectionAlgorithm(*PalletJobs*, W_{max})
3: **for all** *selection* ∈ *JobSelections* **do**
4: generate list of pallets, *Pallets* from selection
5: *Stacks* ← StackingAlgorithm(*Pallets*, W_{st}, H_{st})
6: *PackedStacks* ← SortAndPackAlgorithm(*Stacks*, L_c, B_c)
7: **if** AllStacksFit(*PackedStacks*) **then**
8: **return** *PackedStacks*
9: **end if**
10: **end for**
11: **end**

been selected for packing in descending order by their 'size' (explained above). It then uses the Cygon packer to iteratively pack the stacks into the container. Its overall operation is similar to that of the Packing GA which it replaces: i.e. after it attempts to pack a given set of stacks, it reports if it was able to completely pack all the stacks into the container or not. If the stacks were completely packed, a solution has been found; otherwise, the iteration of the list of selections generated by the selection algorithm in the hybrid algorithm continues until either all the selections have been evaluated or a selection is found that can be packed completely into the container. The hybrid algorithm, modified to use the Sort-and-Pack algorithm for solving the packing sub-problem can be seen in Algorithm 5.2. It differs from the original algorithm in the replacement of the algorithm used for packing the stacks (line 6 of Algorithm 5.2). The reader is referred back to Chapters 4 and 3 for a description of the original hybrid algorithm (algorithm 4.1) and the packing sub-problem (section 3.5.3) respectively.

5.5 Experiments and Results

5.5.1 Comparisons of integrated Rectangle Packing Algorithms

Experiments were performed and comparisons made between the 'Simple', 'Cygon', and 'Arevalo' rectangle packers [see Markus Ewald, 2009], before the 'Cygon' packer was selected as the candidate algorithm for integration with the Packing GA. The three algorithms are from a family of rectangle packing algorithms provided as part of the 'Nuclex' framework [Markus Ewald, 2011b]: a set of fast and elegant components that take care of the grunt work required to implement certain features in a Microsoft XNA game. Their performance was evaluated using historical data separated into problem sets.

The results obtained (see Table 5.1) showed that the Cygon packer consistently outperforms the other two packers in both weight utilisation and computation time. With regards to the weight utilisation achieved, the Cygon packer performed better, consistently achieving 100% utilisation for the worst-, best- and average-case scenarios for both the normal and extended problem sets. The Arevalo and Simple packers followed closely, with the Arevalo packer producing a slightly better result than the Simple packer. With regards to computation time, the Cygon packer noticeably outperforms the other two packers in the worst- and average-case scenarios, with little or no difference in the best-case scenario.

5.5.2 Comparisons of the Packing GA and the Sort-and-Pack algorithm

(i) Layout Comparison: The Packing GA and the Sort-and-Pack algorithm employ the same Cygon rectangle packer for packing and operate in a very similar manner, which is why we are able to easily swap one out for the other. The resulting output from both algorithms however differ in the number of layouts produced. The Packing GA in its operation produces a multitude of layouts (determined by the population size of the GA), while the Sort-and-Pack algorithm produces only one. As the underlying rectangle packer used is the same for both,

Table 5.1: Results obtained for different rectangle packing algorithms

Problem sets	Rectangle packing algorithms	Hybrid algorithm					
		Weight utilisation (%)			Computation Time (s)		
		Worst	Average	Best	Worst	Average	Best
PS1	Simple	99.63	99.97	100	4.19	0.83	0.28
	Cygon	100	100	100	2.93	0.5	0.28
	Arevalo	99.83	99.99	100	10.76	0.84	0.29
PS2	Simple	99.1	99.94	100	3.82	0.7	0.28
	Cygon	100	100	100	0.33	0.3	0.28
	Arevalo	99.98	99.99	100	0.41	0.37	0.33

the type of layouts produced are very similar. The major difference in their operation lies in the way the rectangle packer is used to pack the stacks. In the Sort-and-Pack algorithm, the stacks to be packed are first sorted in decreasing order by size, before being presented to the Cygon packer. In the Packing GA, the stack order is shuffled randomly before the stacks are presented to the Cygon packer. This distinction meant that while both algorithms were capable of producing the same exact layouts in theory, in practice the Packing GA almost never arrives at the same order used for packing by the Sort-and-Pack algorithm; and even in the few cases in which it does, it takes significantly longer to do so.

(ii) Time Comparison: The Sort-and-Pack algorithm performs a lot faster than the Packing GA. This is because, during its operation, the Sort-and-Pack algorithm processes the entire set of input stacks just once. This is in contrast to the Packing GA, which by nature of its implementation has to process the same set of stacks multiple times during its operation, with the typical number of times roughly equal to the product of the GA's 'population size' and its number of 'generations'. As this difference is very clear-cut, no explicit comparison has been presented for the computation times of both algorithms. As alluded to in the previous paragraph, the Sort-and-Pack algorithm is able to arrive at and achieve layouts with a higher utilisation much faster than the Packing GA. In those situations where the same layout and utilisation is achieved, the Sort-and-

Pack algorithm produces the layout much faster than the Packing GA.

5.6 Conclusion

In this chapter, improvements to the hybrid algorithm presented in Chapter 4, made in order to improve the overall packing efficiency of the algorithm, are presented. The first improvement involved replacing the integrated rectangle packer used in the Packing genetic algorithm (GA) component of the hybrid algorithm, i.e., the 'Simple' rectangle packer, with a more space-efficient packer, i.e., the 'Cygon' rectangle packer. This resulted in an increase in the average container weight utilisation achieved by the (improved) hybrid algorithm across the problem sets considered in the experiments. A significant reduction in the computation time of the hybrid algorithm was also observed on the same problem sets. Observation of the results obtained from this initial improvement to the algorithm, revealed that the layouts that had 'larger' stacks packed earlier on in the packing process, tended to have a higher number of stacks packed overall; thus resulting in a higher container utilisation. 'Larger' in this context refers to the total surface area of the stack in contact with the container floor.

This observation led to the second improvement to the hybrid algorithm, which involved the replacement of the entire Packing GA with a much simpler 'Sort-and-Pack' algorithm. This new algorithm simply sorted the input stacks in descending order by size (of bottom stack surface area) before packing them with the same 'Cygon' packer employed initially. This change resulted in the modified algorithm achieving layouts with a higher container utilisation much faster than with the Packing GA. Overall, the gains obtained from this improvement are in the much faster computation times with which the layouts are produced. This significant increase in speed was due to the fact that the 'Sort-and-Pack' algorithm only processes a given set of stacks once, in comparison to the Packing GA which processes the same set of stacks for a significantly higher number of times.

Chapter 6

Optimising Container Layouts for Real-World Packing

6.1 Introduction

In the work presented so far, we have solved a version of the container loading problem (CLP) with a number of defined constraints. Our solution presents a selection of pallets that is known to fit entirely in the enclosing container as well as maximise the weight capacity of the same container. We are also able to show exactly how the selected pallets will be placed in the container to achieve the observed fit. This is similar to many of the solutions to the CLP observed in the literature where success is usually determined by the measure of the container's volume utility. Sometimes, as in our case, the measure of success is driven by the container's weight utility. Little or no attention is usually paid to the aesthetics of the container loading layouts produced, nor to the practicalities involved in the physical loading of such layouts. Physically loading in the real world might typically involve the use of machinery such as forklift trucks. This is of consequence because the trucks are only able to move in and out of a container in a particular way, and might place further restrictions on how pallets can be picked up, oriented and placed in a container. In our study, we observed that one of the smallest pallet sizes available to us could only be picked up in one orientation because the holes available on the other orientation of the pallet where too

close together to allow it to be picked up by the forklift truck. These sort of considerations really need to be in place when producing container layouts for the CLP otherwise we will be left with solutions that in theory should work, but are totally infeasible in practical cases where motion is constrained by the ability to be picked up and moved using forklift trucks.

In this chapter, we therefore present an attempt to tackle this issue of optimising a container layout for physical loading in the real world. To do this, an entropy measure was derived to help give an indication of how feasible or desirable a loading layout is to physically load. The desirability in this sense was a qualitative measure of how content a loader would be if he was given the layout to load. This often went hand in hand with the feasibility of the layout - i.e. how practical it would be to load using a forklift truck. If the layout was complex and unfeasible, it would be less desirable to load. There were also cases of layouts that would be feasible to load, but not as desirable because the layout might involve extra work for the loaders e.g. having to lift the pallet a first time to move it out of storage racking, placing it on the floor somewhere, and manoeuvering the forklift truck to the side of the pallet so as to pick it up from its second (alternate) orientation. The experiments performed involved experienced loaders identifying feasible layouts with some indication of which layouts they would prefer to load. Thus, they selected layouts that were the most practical to load, which minimised the amount of work they would have to do when using forklift trucks. In their criteria for desirability, they also often implicitly took into consideration layouts that provided a rigid structure within the container, so that when the container is in transit, the pallets will have none or very minimal motion within the container. This should help minimise the possible damage that could occur in transit. Tuning the measure to match these expectations meant the measure would help identify layouts that closely mimicked actual physical loading as carried out by a human. Indeed, the experiments that followed revealed a strong correlation between layouts that had a very low entropy measure and layouts that were simple to understand and reproduce; while layouts that had a very high entropy measure were those that would involve more work when loading them using forklift trucks.

6.2 Deriving an Entropy-based measure for Container Layouts

The derivation of entropy for use as a computational aesthetic measure is not a new idea. Detailed discussions covering the idea and a formulation of an entropy measure are presented in Cant et al. [2012]. In this section, we present a derivation of an entropy measure that is heavily influenced by the same discussions and provide an interpretation and application of the measure for 2D container loading layouts. As such, we borrow from and refer to a number of the mathematical formulations in those discussions.

Entropy as a measure is usually associated with thermodynamics in physics, where it is a well-defined quantity that represents the number of specific ways in which a thermodynamic system may be arranged. In this context, it is commonly understood as a measure of the disorder of the system in question. Formally stated, if a system is in a macrostate that has a number of possible microstates, then the entropy of the system is defined as the logarithmic measure of the number of microstates that can give rise to an observed macrostate:

$$S = k_b \ln(\Omega) \tag{6.1}$$

where k_b is the Boltzmann constant, and Ω is the number of microscopically distinct states that would give rise to the same, measured, macroscopic variables.

In our application, there are no obvious macroscopic variables, so we will next examine another definition for entropy associated with information theory. The definition is provided by Shannon [Shannon, 1948] and it expresses entropy as a measure of the uncertainty or unpredictability about a source of information. It is based on the equation:

$$H = -\sum_i p_i \ln(p_i) \tag{6.2}$$

where i indexes the possible states of the system and p_i is the probability of a particular state occurring. Hence Shannon entropy relates to the probability distribution that generates the states rather than to the individual states them-

selves.

Note that if all states are equally probable, and there are N possible states then the entropy simply becomes:

$$S = N \times \left(-\frac{1}{N} log \left(\frac{1}{N} \right) \right) = logN$$

reducing to the definition in (6.1) above. In this case the role of the thermodynamic variables is played by the probability distribution i.e. the macrostates are identified with the rules that generate the data rather than the data itself (see Cant and Langensiepen [2010]).

In calculating the entropy for a container layout, the states are identified with the rules used to generate the layout. These rules are then deduced and all the alternative arrangements that could be generated by the same rule are examined and each assigned a probability (i.e. the probability distribution used to generate the layout configuration). This enables the use of Shanon's formula in 6.2 to generate an entropy value. In practice however, there seems to be no easy way of assigning different probabilities to the different states, so equal probabilities are assigned to each. The task is then to count the number of ways in which an equivalent layout could be produced. Of course, the meaning of the word equivalent is to some extent a matter of human judgement. As a simple example of this, consider a sequence of playing cards. If we are not restricted to a single deck and select two cards in succession, the second card could be identical to the first, and there is only one way to do that. For example, the 4 of spades would be followed by another 4 of spades. The number of equivalent states is just 1 and the entropy is 0. If the second card has the same number and colour but a different suit, then there are now two options (i.e. 4 of spades followed by 4 of clubs, or a 4 of spades followed by another 4 of spades) hence the entropy will be log 2. If the colour is also different (e.g. 4 of spades followed by 4 of diamonds), then there are now four choices and the entropy will be log 4. If we follow the 4 of spades with the 9 of spades, there are 13 options (i.e. any of the 13 spades) and hence the entropy will be log 13. We can separate the entropy associated with the suit from the entropy associated with the number. If we follow the 4 of spades with the 9 of clubs then there are two suit choices and 13 number choices

so the number of options altogether is $2 \times 13 = 26$ and the entropy is log 26 or log 2 + log 13. In other words, the entropies associated with different aspects of the arrangement can simply be added. Using this principle, the entropy of a longer sequence of cards can be deduced as the sum of the entropies associated with the individual steps. Note that there are potential ambiguities associated with the deduction of these rules and the resulting entropies. For example, given the sequence 4 followed by 5 we could choose the number entropy to be log 13 or we could note the sequential nature and choose the number entropy to be log 2 (this would be on the basis of the two choices 4-4 or 4-5). Ultimately one should make sure that whatever convention is adopted is used consistently and reflects the requirements of the particular problem at hand. In the following sections, we will identify the conventions that we have adopted for the container loading problem.

Generally, there will exist a number of different sets of rules, and these rules relate each stack in the layout to every other stack in terms of their semantic relevance (i.e. selection entropy, defined in 6.2.2), geometric arrangement (i.e. rotational entropy, defined in 6.2.3) and distance (i.e. positional entropy, defined in 6.2.4), relative to each other. In practice, we proceed with the calculation of entropy for a layout by creating a connected graph that relates each stack in the layout to every other stack. Every edge in the graph is a link between two stacks that represents the rule that relates the two connected stacks. The edge weights hold the respective calculated values corresponding to the selection, rotational and positional entropies. We then compute a minimum spanning tree (MST) from the connected graph using only the positional entropy value as the edge weight to drive edge selection. We choose to use the positional entropy value to drive the choice of the rules selected based on the assumption that layouts are generated in sequence, one at a time, with stacks placed relatively close to each other, and that the entropy of each new stack is only calculated with reference to stacks that have already been calculated. This removes the ambiguity of choice faced when trying to determine whether to use only a specific entropy value, a combination of any two, or a combination of all three different calculated entropy values, to drive the choice of the rules selected. The MST represents our entropy tree and is guaranteed to connect all stacks in the graph together using

the shortest positional entropy value path along its edges. This has the effect of giving a higher consideration to nearest neighbour stacks when connecting stacks in the layout to each other The entropy of the entire layout is then calculated as a combined sum of the entropy values for all the selected rules that are assumed to have generated the layout. In conclusion, the overall entropy value of the layout is computed as the combined sum of the three respective entropy values associated with every edge in the resulting entropy tree.

6.2.1 Basic Definitions

In this thesis, we associate our entropy measure with the aesthetic look and feel of a 2D container layout and present it as a measure of the disorderliness of the layout. The higher the value of the entropy measure, the higher the perceived disorderliness of the layout. Layouts are typically presented as large rectangular boxes, which represent a container, that contain a number of smaller rectangular boxes, which represent the packed items.

In determining the aesthetic measure or feel for a 2D container layout, the types of items present in the layout, as well as the geometric arrangement and position of the items need to be taken into consideration. Following on from this, we identify the need to create relationships between items that indicate and measure their semantic relevance (i.e. the types of items present) and their orientation and position in the layout relative to each other. We account for each of these relationships in the subsections that follow.

The following is a summary of the general entropy formulation as presented in Cant et al. [2012]; we identify the simplified form and show how we use this form in subsequent calculations in the subsections that follow. Let T be the set of states of a given system. A rule R is defined as a boolean function on T such that:

$$W(R) = \{t \in T | R(t)\}$$

is the set of states that are observed to obey the rule. For differences that can't be observed or are not important, a concept of symmetry, Z, is introduced and defined as an equivalence relation on T such that:

$$[a] = \{t \in T | t \ Z \ a\}$$

and $[a]$ is the set of states indistinguishable from a. If Z is to be consistent with R, then for two states, a and b, we say:

$$a \ Z \ b \rightarrow R(a) = R(b).$$

The entropy S for the rule R given the symmetry Z is then formally defined as the logarithm of the cardinality of the quotient set of $W(R)$ by Z:

$$S(R, Z) = \ln(|W(R)/Z|).$$

The cardinality of the quotient set can be rewritten as $\Omega(R)$ thus we have a simplified form:

$$S(R) = \ln(\Omega(R)). \tag{6.3}$$

Further simplification of (6.3) gives us:

$$S = \ln(\Omega),$$

which is the same as the original physics definition introduced earlier in (6.1).

6.2.2 Selection Entropy

We formulate the selection entropy as a measure of the semantic relevance of items in a layout to each other. The semantic relevance, in the examples we use, is determined by the 'type' of an item. Practically, an item's type is determined by its length, width and possible orientations. In the example layout in Figure 6.1, we have three different types indicated by the different coloured boxes. For any two items, we calculate the selection entropy value of the rule that relates the items to each other as the logarithm of the cardinality of the set of states that are observed to obey the rule.

Determining these sets of states presents some ambiguity. In the example layout, the set of states that contains both items for a link between any two 'blue'

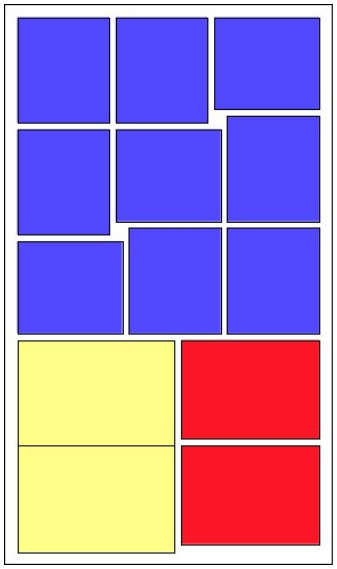


Figure 6.1: An example of a 2D container layout

items can be said to have a cardinality of 9, because there are 9 different blue items to choose from, or 1 because all the blue items are identical in type and ultimately there is only 1 type of blue item. Similarly, the cardinality of the set of states that links a blue item and a yellow item can be said to be 11 because we have a choice of 9+2 different items to choose from in order to select a blue and yellow item, or it can be said to be 2 because we are only selecting from 2 different item types. We can even go further to say that the cardinality of the set for the states that obey the rule for the blue-yellow link is 13, not 11, because, in the absence of any links between the item types, the multiset that contains both blue and yellow items is the universal set that contains all available 13 items.

Formally, this can be represented as:

for items $x \in A$ and $y \in B$, where A and B are sets that each contain a specific item type, and U is the universal set containing all items:

$$\textbf{if } A == B,$$

$$\textbf{then } S(x, y) = \ln(|A|) = \ln(|B|),$$

$$\textbf{else } S(x, y) = \ln(|AUB|) \textbf{ or } \ln(U),$$

where $|A|$ is the cardinality of A.

In our experiments, we calculate the selection entropy for the rule that relates similar items as the logarithm of the total number of items of the same type; and that for different items as the logarithm of the total number of all items present.

6.2.3 Rotational Entropy

The rotational entropy is a measure that is indicative of the geometric arrangement (orientation) of items relative to each other. For any two items, we calculate the rotational entropy by examining the orientation of both items. If the orientations are the same, we set the rotational entropy value to zero (i.e. the logarithm of 1); otherwise if the orientations are different, we set the value to the logarithm of 2. In arriving at this formulation, we make the assumption that items are packed in orthogonal orientations parallel to container walls. As such, only two orientations are taken into account: 0° and 90° (orientations 180° and 270° are not considered as separate orientations due to their symmetry to 0° and 90° respectively). Formally, the calculation is represented as:

for items x and y,

$$\textbf{if } x.orientation == y.orientation$$

$$\textbf{then } S(x, y) = \ln(1)$$

$$\textbf{else } S(x, y) = \ln(2)$$

This results in producing lower entropy for items that are packed in the same orientation, and a higher entropy for items that are packed in different orientations.

6.2.4 Positional Entropy

The positional entropy is a measure determined by the vertical and horizontal displacement between any two items. This displacement is measured from the centre of the items. For two stacks a and b both with coordinates on the Cartesian plane, where the point $(x1, y1)$ represents the top left corner, a_x and b_x represent the dimensions of both stacks in x, and a_y and b_y represent the dimensions of both stacks in y, we define the horizontal displacement as the difference of the position of their centres in the x direction, X_{disp} as:

$$X_{disp} = \left| \left(a_{x1} + \frac{a_x}{2} \right) - \left(b_{x1} + \frac{b_x}{2} \right) \right|$$

and define the vertical displacement as the difference of the position of their centres in the y direction, Y_{disp} as:

$$Y_{disp} = \left| \left(a_{y1} + \frac{a_y}{2} \right) - \left(b_{y1} + \frac{b_y}{2} \right) \right|$$

We then calculate the horizontal displacement entropy, S_x as:

$$S_x(a, b) = \ln \left(1 + \frac{2 \cdot X_{disp}}{a_x + b_x} \right)$$

if

$$X_{disp} \leq \frac{a_x + b_x}{2}$$

otherwise

$$S_x(a, b) = \ln \left(1 + \frac{2 \cdot X_{disp}}{b_x} - \frac{a_x}{b_x} \right)$$

Similarly, we calculate the vertical displacement entropy, S_y as:

$$S_y(a, b) = \ln \left(1 + \frac{2 \cdot Y_{disp}}{a_y + b_y} \right)$$

if

$$Y_{disp} \leq \frac{a_y + b_y}{2}$$

otherwise

$$S_y(a,b) = \ln\left(1 + \frac{2 \cdot Y_{disp}}{b_y} - \frac{a_y}{b_y}\right)$$

We therefore calculate the positional entropy, $S(a,b)$ as:

$$S(a,b) = S_x(a,b) + S_y(a,b)$$

6.3 An Entropy-driven Genetic Algorithm for the Packing Problem

Once an entropy value could be calculated for a layout, layouts composed of the same set of stacks could be compared relative to each other. The comparisons had to be between layouts composed of the same pallet load, as differences in the number and types of pallets in a layout have an impact on the derived entropy measure. This comparison was automated and built into the algorithm by the introduction of the derived entropy measure as a fitness function for a genetic algorithm (GA). The Packing GA from Chapter 4 was re-used for this exercise. This way, in the packing step of the hybrid algorithm, we are able to generate a lot of different layouts from the same set of stacks and use the entropy measure to drive the comparison and selection of the 'fittest' layouts i.e. those with the lowest entropy scores. These 'fittest' layouts are hypothetically the most practical and straightforward to load layouts using a forklift truck. The operation of this entropy-driven GA can be seen in Algorithm 6.1.

6.4 Experiments and Results

In the experiments, entropy values were computed for a number of layouts using the derivations presented earlier. Layouts were then shown in groups to experienced loaders so the loaders could rate the layouts in terms of their perception of what was more 'ordered' and 'desirable' to load. The correlation between the calculated entropy values and the loaders' perception of what layouts were most

Algorithm 6.1 Entropy Packing Genetic Algorithm

```
Input:
  PalletStacks
Output:
  StackLayout                                   ▷ selected layout of stacks

  InitialisePopulation(PalletStacks)
  EvaluatePopulation
  while max generation count not reached do
    create new population
    apply elitism
    while total population count not reached do
      Parent_1, Parent_2 ← RandomSelection
      Child_1, Child_2 ← Crossover(Parent_1, Parent_2)
      Mutate(Child_1)
      Mutate(Child_2)
      add Child_1 and Child_2 to new population
    end while
    EvaluatePopulation
  end while
  StackLayout ← fittest individual in most recent population
  return StackLayout
  end

  function InitialisePopulation
    for all individual ∈ population do      ▷ each individual is a list of stacks
      individual.fitness ← 0                          ▷ intialise fitness to 0
      Shuffle(individual)                    ▷ shuffle the position of stacks
      for all stack ∈ individual do
        RandomiseOrientation(stack)   ▷ set stack orientation to 0 or 1
      end for
      layout ← GenerateLayout(individual)
      if AllStacksFitInLayout(layout) then
        individual.fitness ← CalculateEntropy(layout)
      end if
    end for
  end function
```

```
function EvaluatePopulation
    for all individual ∈ population do
        layout ← GenerateLayout(individual)
        if AllStacksFitInLayout(layout) then
            individual.fitness ← CalculateEntropy(layout)
        else
            individual.fitness ← 0          ▷ penalty for stacks that don't fit
        end if
    end for
end function
```

ordered and best for loading, was recorded.

The following figures show groups of layouts and their respective calculated entropy values. These groups, each consisting of 2 to 3 layouts, were presented to the loaders to rate. No notion of a 'measure' was provided or mentioned to the loaders, so as not to influence their choices; they were simply presented with layouts 2 or 3 at a time and asked which of the layouts they would rather load - i.e. which was the better layout. Table 6.1 shows the relationship between the calculated entropy values of the layouts and their ratings of perceived orderliness by the loaders.

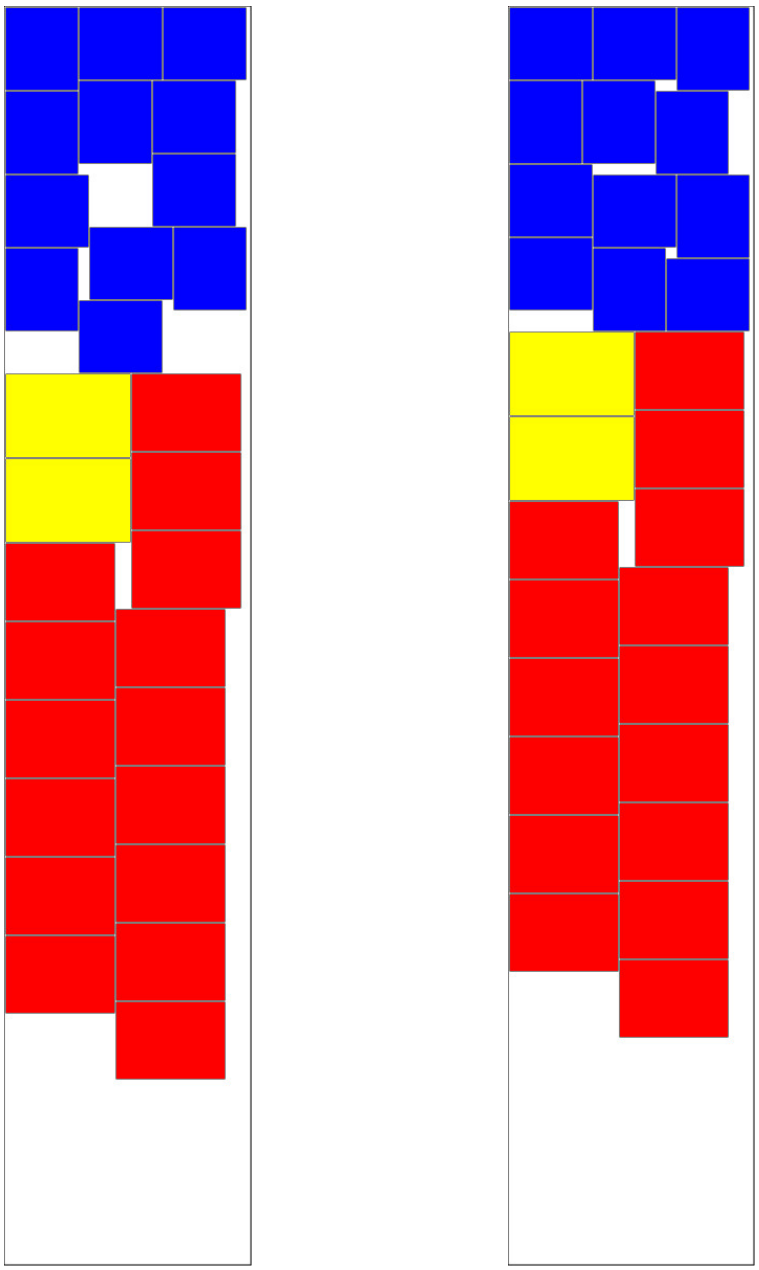
For groups 1, 3, and 4 (Figures 6.2, 6.4 and 6.5 respectively), the layouts presented were rated the same by the loaders as by the entropy measure. For the layouts in group 2 (Figure 6.3), the loaders rated layouts 3 and 4 as being more ordered than layout 5. This correlated with the calculated entropy values for the layouts. They however rated layout 3 as being more ordered than layout 4, which did not correlate with the ratings obtained using the entropy measure. This observation was puzzling and was the first of its kind where there was a mismatch between what the entropy values indicated and what the loaders said. Further conversation with the loaders revealed that in making their decision for this particular case, the loaders were using additional information not known to us, in the form of a loading constraint they needed to meet when loading containers. This constraint was not considered or included in the calculations made for the entropy measure. Layout 3 was therefore apparently selected as their preferred layout because it filled out the container more breadth-wise, which would then reduce any possible motion laterally within the container, thus reducing the

Table 6.1: Entropy vs Loader rating (most ordered first) for layouts

Group	Layout	Entropy value	Entropy rating	Loader rating
1	1	98.2	2, 1	2, 1
	2	97.95		
2	3	101.70	4, 3, 5	3, 4, 5
	4	97.73		
	5	111.29		
3	6	100.36	6, 7	6, 7
	7	107.34		
4	8	97.57	8, 9	8, 9
	9	98.27		

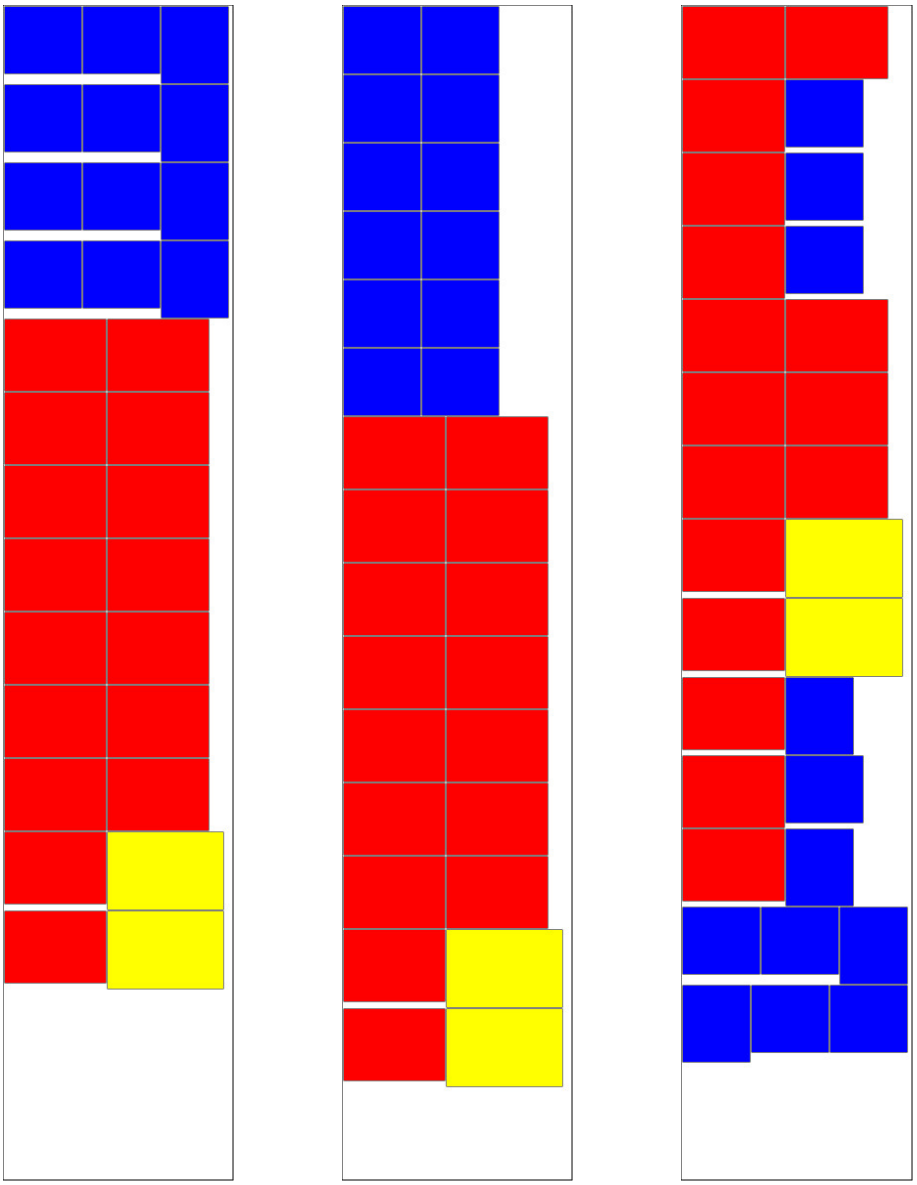
possibility of damage that could occur in transit. This constraint was the major influence and deciding factor in their selection of layout 3 as the more ordered layout.

In general, for all of the experiments, despite the occasional differences in loading preference or style of the different loaders, they all agreed on and provided the same ratings of what they thought as being more ordered. The correlation between their own concept of order and the entropy measure of the layouts was high. All the calculated entropy values for all of the layouts presented to the loaders, except for the group shown in Figure 6.3, correlated with and matched the loaders' rating and perception of orderliness. In such cases, as discussed earlier, we determined that other factors and constraints not present in the calculations for the entropy measure were involved in the decision making process when rating the layouts, and were the reason for the mismatch.



(a) Layout 1 Entropy: 98.92 (b) Layout 2 Entropy: 97.95

Figure 6.2: Entropy comparisons: Group 1 layouts

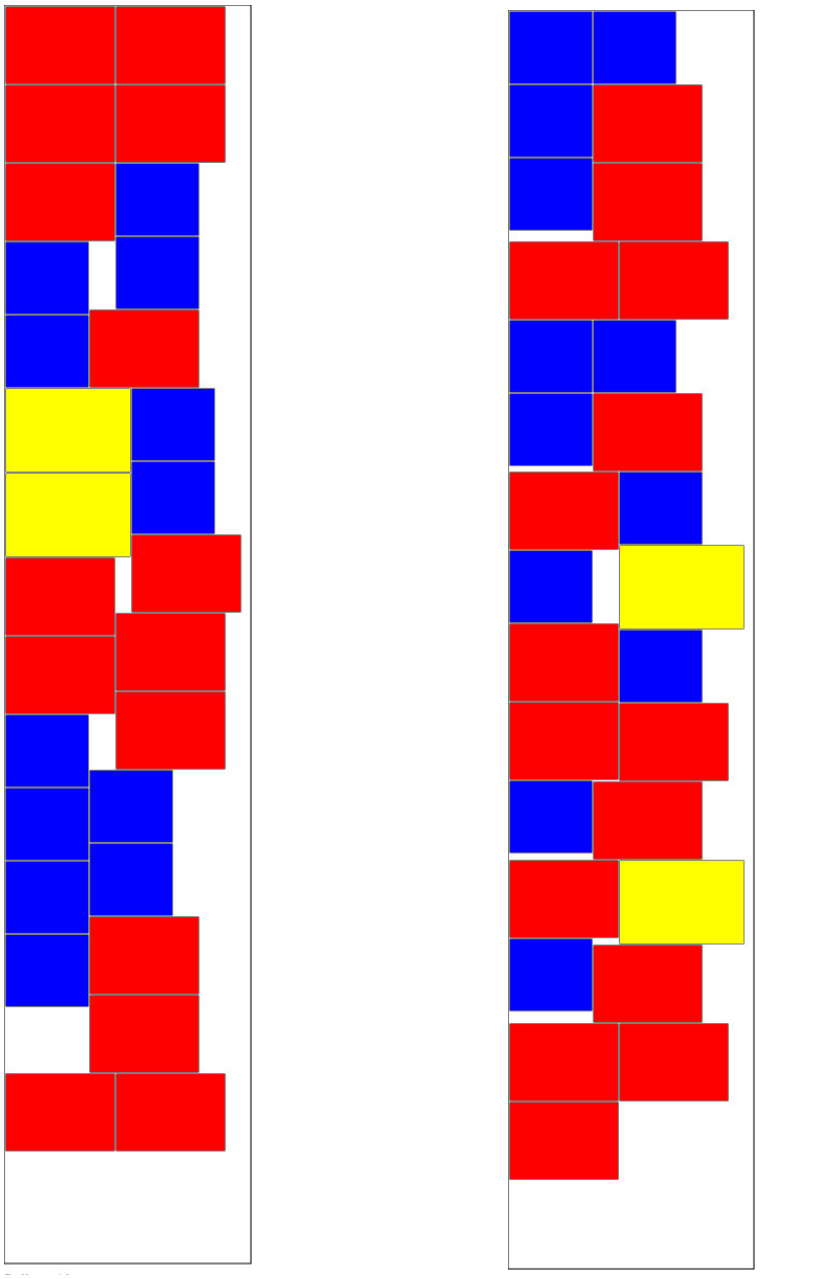


(a) Layout 3
Entropy: 101.70

(b) Layout 4
Entropy: 97.74

(c) Layout 5
Entropy: 111.29

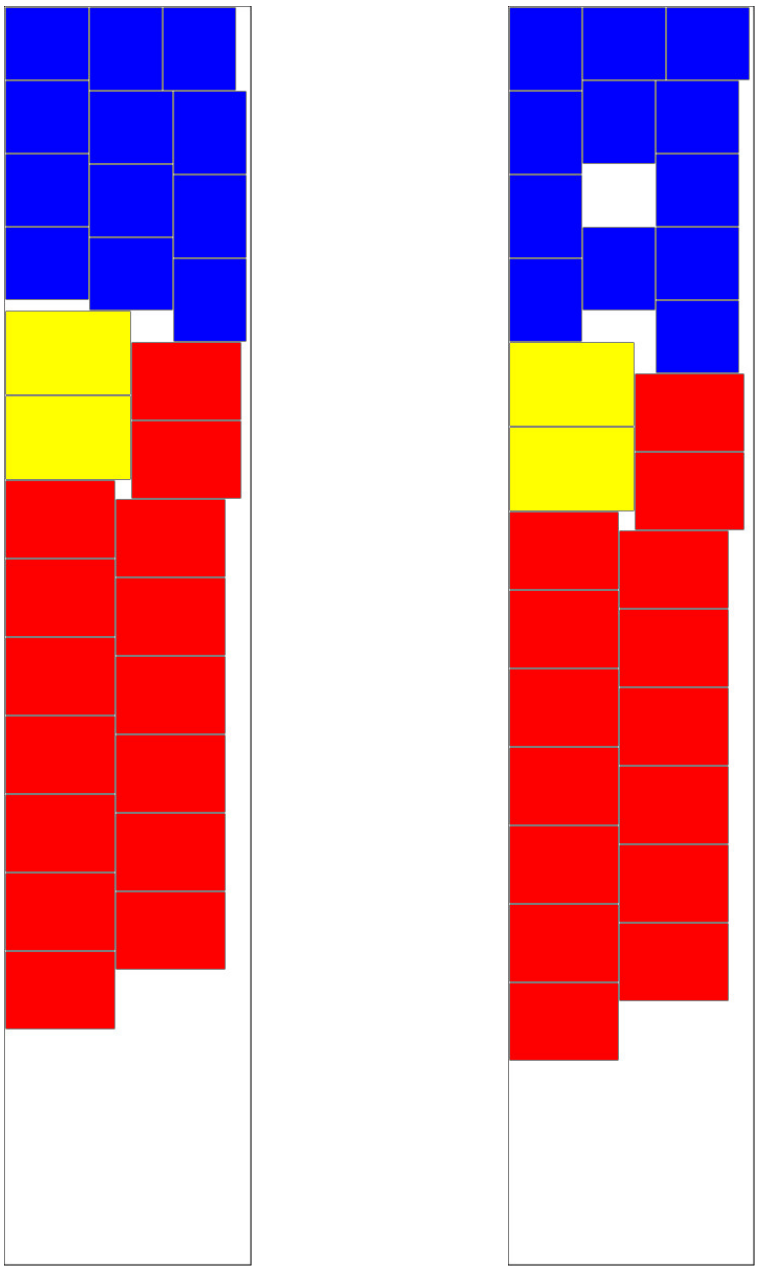
Figure 6.3: Entropy comparisons: Group 2 layouts



(a) Layout 6 Entropy: 100.36

(b) Layout 7 Entropy: 107.34

Figure 6.4: Entropy comparisons: Group 3 layouts



(a) Layout 8 Entropy: 97.57 (b) Layout 9 Entropy: 98.27

Figure 6.5: Entropy comparisons: Group 4 layouts

6.5 Conclusion

In this chapter, a method of calculating entropy for a 2D container layout as a measure of its disorderliness is derived. The entropy measure is calculated as the combined sum of different individual entropy values that each identify relationships of type, geometric placement, and position respectively, for all items in a loading layout. Feedback from expert loaders was used to assess the validity of the measure and to demonstrate that the measure can indeed be used to rate loading layouts in practice: in terms of their aesthetic look and feel or disorderliness, and their desirability to be loaded. The lower the overall entropy value of a layout is, the higher the aesthetic value or desirability of the layout, and vice versa. The measure is viable and consistently rate layouts reasonably well in terms of their disorderliness, obtaining the lowest entropy values when we place similar items (selection entropy) close together (positional entropy) in the same orientation (rotational entropy). As seen in the experiments, exceptions may occur where there is a mismatch in the correlation of the calculated entropy value and the loaders' own perception of orderliness for a layout. This occasional mismatch does not reduce the effectiveness of the measure; it only serves to point out that the measure in itself does not completely cover all of the factors and constraints that an experienced loader would consider when determining a feasible layout.

Chapter 7

An Entropy-Guided Monte-Carlo Method for Generating Optimal Container Loading Layouts

7.1 Introduction

In Chapter 4, we presented a framework for the algorithmic selection of groups of pallets from a larger collection of pallet groups, to be loaded into a container. In Chapter 5, we further improved the packing algorithm to obtain solutions that were more efficient in container space utilisation. In Chapter 6, we defined a measure of order and consistency for a layout based on entropy, and we showed that forklift drivers found low entropy solutions easier for them to understand and achieve.

In this chapter, we show how low-entropy solutions to the CLP can be efficiently produced by the new technique of using entropy to direct a Monte Carlo tree search (MCTS) process. In doing this, we take a holistic approach to the loading process in that we try to combine efficiency of space utilisation within the container with simplicity for the forklift drivers i.e. the layouts produced are easy to understand and implement in real-world loading using forklift trucks. This reduces the time that a container spends being loaded, and the number of containers required for a given single job. The proposed algorithm, i.e., the entropy-guided

Monte Carlo packing algorithm, was used in a series of experiments to assess its performance in achieving layouts that optimised space usage while being human loadable. These experiments, as well as a discussion of the observed results and a number of layouts generated by the algorithm, are presented.

7.2 Related Work

The work by Bischoff and Ratcliff [1995b] had a significant impact in the consideration of practical constraints that might impact the methods used for solving CLPs. Techanitisawad and Tangwiwatwong [2004] included container stability and stack priority within their solution and used an integrated heuristic approach based on genetic algorithms for container selection and loading. Peng et al. [2009] considered orientation and stability constraints but used a hybrid simulated annealing algorithm. Bortfeldt and Gehring [Bortfeldt and Gehring, 2001; Gehring and Bortfeldt, 1997] also considered orientation, stability, top placement, weight, and balance constraints, and again used a genetic algorithm approach. These examples have specifically included some of the real-world constraints to achieve solutions and generally have taken heuristic or hybridised approaches.

Taking the real-world problems further, Gendreau et al. [2006] generated solutions that took vehicle route constraints into account with the CLP. However, I have not been able to find work within the literature that addresses the issues of being able to achieve a packing within a container where there are the physical constraints of getting the packed boxes into the required positions using a single entry point, as well as the limited manoeuvrability associated with the packing of heavy pallets. These extra constraints affect the density at which items may be packed and the required simplicity of its layout. Although Iori and Riera-Ledesma [2015] considered loading based on a 'last in, first out' principle, the emphasis was on the decisions associated with the choice of vehicles and the travelling salesman route issue, rather than the container layout. More recently, Moura and Bortfeldt [2016] discussed how the process could be optimised for distribution to multiple customers with trucks packed in 2 layers, optimising numbers of trucks used; while Alonso et al. [2017] took a mathematical approach to optimising loading and unloading effort by minimising the number of trucks to be used.

In previous work [Cant et al., 2012; Cant and Langensiepen, 2010], a new interpretation of the concept of entropy was derived, taking influences from its use in the domains of physics and information into the domain of graphical scene layouts for computer games. A new measure was derived from this interpretation and applied to the container layouts generated by the algorithms presented earlier for the CLP in this thesis. The measure helped to provide a quantitative assessment of the ease of loading any particular layout for the forklift drivers. This was explored earlier in Chapter 6, where we found that the measure seemed to correlate with what the forklift drivers considered to be easy layouts to load.

Coulom [2007] combined tree based searching with Monte Carlo evaluation and applied it to the challenging board game Go. Its potential for wider impact was seen immediately so that by 2012 a survey by Browne et al. [2012] could cite more than 240 papers using MCTS in a range of areas and variants from computer Go through crossword puzzle generation to printer scheduling. Moura and Oliveira [2008] have used MCTS in their work on combining the travelling salesman problem with container loading, but they provided their own GRASP technique for the CLP aspects of the problem, using the MCTS to direct the combination of load selection and route. Most of these methods use the UCT: 'Upper Confidence Bound 1 applied to trees' (or a closely related) variant of the MCTS, introduced by Kocsis and Szepesvári [2006]. Moura and Bortfeldt also used a tree search algorithm for filling trucks in their work [Moura and Bortfeldt, 2016] mentioned earlier, but in this case, the tree search was not Monte Carlo; rather, it was a recursive process to ensure the pallets were stacked in an order suitable for delivery.

The implementation presented in this chapter is closer to that of Pure MCTS, with the distinguishing factor being the application of a derived measure (i.e. entropy) combined with elements of randomness using a weighting, as a method for influencing the choice of nodes during MCTS playouts; instead of a purely random choice for nodes. A number of experiments were performed using the UCT variant of the MCTS. Initial results suggested that the overhead of keeping the information about earlier passes combined with back-propagation resulted in more costs than gains in performance. Subsequent results indicated that for our problem instances, the UCT variant doesn't do any better than the purely

random variant.

7.3 Proposed Algorithm

The remaining problem requires the packing of a selected set of pallet stacks into a shipping container such that the packing achieves as high a density as possible, compatible with loading from a single entry point with forklift trucks, such that the layout produced is consistent enough to be easily assimilated by the loaders to make their job easier, and that it can be achieved within a reasonable time without significant computer resources. We have developed a new algorithm for this purpose that has two key aspects in order to achieve the goals.

7.3.1 Placement Method

In order to achieve feasibility of loading, the placement of each potential stack within the container is performed by placing the stack randomly in the spare space in the container, moving it towards the already filled area, then sliding it to abut the stack to its left so that it aligned as close as possible to the left bottom corner of the previous stack (as seen in the layouts presented subsequently in this chapter). If the stack cannot be fitted into the current 'strip' of stacks in this manner, a new strip is started. This 'Tetris-like' method ensures that the resultant packing layout can be achieved by the loaders using their forklift trucks.

7.3.2 Directed Choice

In order to achieve a loading layout that can be easily assessed and carried out by the loaders, the selection of the next stack to be placed uses a Monte Carlo tree search where the weighting is generated by a measure of the entropy of the layout. The actual calculation of entropy for a stack is calculated as discussed in Chapter 6 and the overall entropy for a layout that includes it is generated by summing it with the running entropy for the previously placed stacks.

Algorithm 7.1 Entropy Guided Monte Carlo Tree Search

```
Initialise all stacks to unplaced
while Some unplaced stacks remain do
    Initialise S_min
    for Each unplaced stack q do
        for Each orientation o of q do
            Calculate entropy S(q, o) relative to existing layout
            if S(q, o) < S_min then
                S_min = S(q, o)
            end if
        end for
    end for
    Select one unplaced stack and orientation randomly, using Algorithm 7.2
    Add chosen stack to layout in selected orientation
end while
Record layout, entropy, length
```

7.3.3 Algorithm description

In our process, as expressed in Algorithm 7.1, we try placing each unplaced stack in each possible orientation and calculate the entropy. During this process, we also track the minimum entropy S_{min}. We allow entropy to direct the branch chosen at each node in the tree i.e. the choice of the next stack in a given orientation to be included in the container layout. Equation 7.1 is used to compute a probability $P(q, o)$ which can be used to bias the choice made towards lower entropy configurations.

$$P(q, o) = \frac{1}{1 + \omega \times (S(q, o) - S_{min} - 1)} \tag{7.1}$$

where ω is a weighting parameter that determines how strongly entropy affects the outcome.

There are two special cases. If the weighting, ω, given to the entropy is 0, then all probabilities will be equal and the choice is completely random; whereas if the weighting is 1, (7.1) is singular for $S(q, o) = S_{min}$, resulting in the purely deterministic choice of the branch producing the lowest entropy solution thus far. Occasionally, there will be more than one configuration with the same lowest

Algorithm 7.2 Weighted Choice of Stack

if $\omega = 0$ **then**
  All P are equal (pure monte carlo)
  Select randomly
else if $\omega = 1$ **then**
  All lowest (degenerate) entropies have P equal (e.g. 1), all others are 0
  Find all *stacks* and corresponding *orientations* with lowest entropy
  Select randomly within this subset
else
  Calculate total probability for all allowed stacks:
  Set P_{total} to 0
  for all stack q and orientation o **do**
    Calculate $P = \frac{1}{1+\omega\times(S(q,o)-S_{min}-1)}$
    where S_{min} is the lowest entropy found in this set,
    and $S(q, o)$ is the entropy of the stack q in orientation o
    Add P to P_{total}
  end for
  Select stack q and orientation o randomly with distribution P
end if

entropy solution, so a random selection will still be required between these options only. For values between these two extremes, the choice of which stack to include is based on a weighted distribution based on the entropy calculations for all the potential next stacks (Algorithm 7.2).

As a sanity test of the need for both the placement and entropy aspects of the algorithm, the pure Monte Carlo tree search method ($\omega = 0$) was trialled without enforcing the Tetris-like placement method specified in subsection 7.3.1. This selected stacks randomly, packing them in a randomly selected orientation (for stacks with multiple packing orientations) and placed the stacks in a random choice of available placement points within the container. The method failed to achieve any successful layouts (i.e. ones wholly within the outline of the container) despite providing a relatively small number of stacks to place and running for 48 hours of processing.

My presented version of the MCTS needs only local values to decide its path (i.e. the entropy associated with adding a particular stack), and so has no need to perform the back-propagation stage used in implementations that improve their

estimates of global quantities during their branch exploration.

7.4 Experiments

The experiments were carried out on a farm of identical four-core AMD A8 desktop machines, with 8GB of RAM, running Windows 10 with no other user programs active. Each machine was running three instances of the experiment, each of which was explicitly bound to one of the cores. The software was written in C# and no explicit optimisation or parallelisation was applied.

In order to assess the consistency and robustness of the algorithm, we applied it to a range of situations. The container layouts were categorised in terms of the 'fill level'. This denoted the relative proportion of the floor area of the selected stacks compared to that of the container - in other words, a $2D$ liquid measure that takes no account of individual dimensions, only absolute floor area covered. At the UKDC, four different pallet dimensions are used, and so each stack would have one of those four as its base. For each of the different fill levels to be tested in the experiments (i.e. values ranging from 60% to 99% of container space), fifty sets of stacks were randomly generated using the 4 possible pallet sizes. The use of the notional fill level and the randomisation of the stacks used in each experiment meant that it was quite likely that for high fill levels it might be impossible to get all the stacks into the container once the actual dimensions were taken into account by the algorithm.

The algorithm was then applied to produce layouts for each of the random sets of stacks at each of the fill levels. At each iteration within the algorithm, if a better layout was achieved, it was recorded. A layout was considered better if it achieved an overall lower entropy or if it occupied less space along the length of the container (the 'length measure' referred to in the algorithm). In either of these cases, the values, the time taken and the number of iterations were recorded. In Monte Carlo tree search (MCTS) as used here, a stopping criterion is required. Initial experiments showed that, over a range of fill levels, the algorithm had already achieved its best results well within 600 seconds (see Figure 7.1). This was therefore used as the stopping criterion to end the experiments. The early termination of some lines shows that no further improvement in entropy was

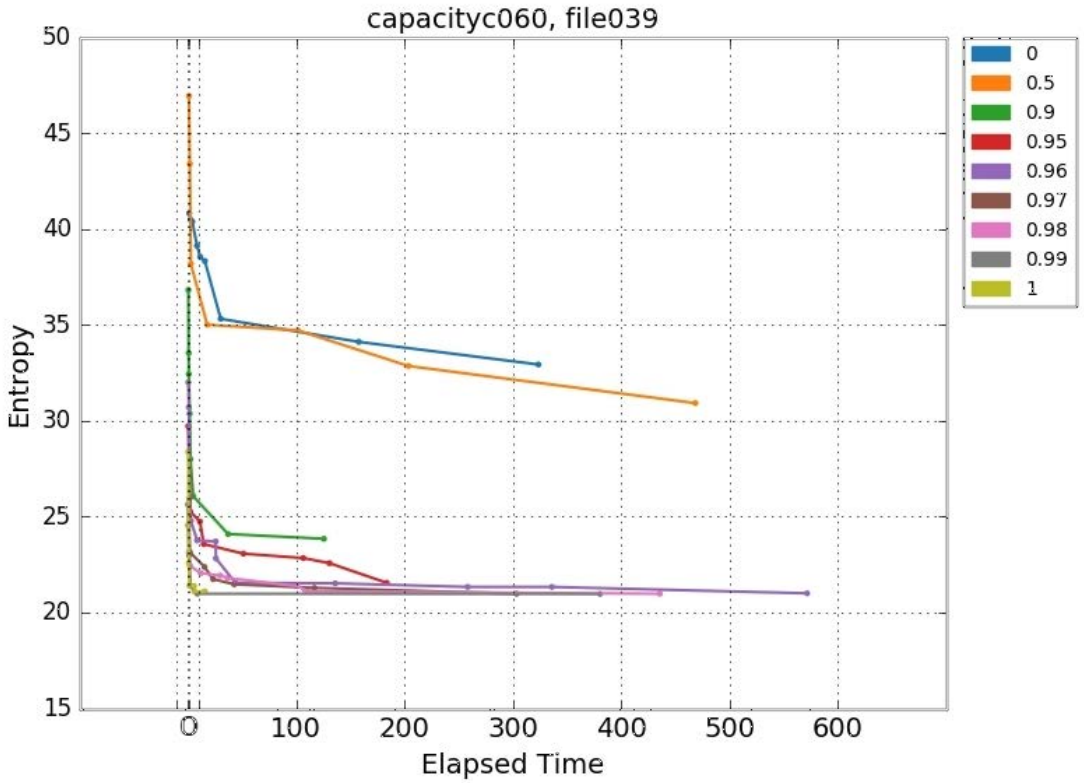


Figure 7.1: Entropy variation with time for a single set of stacks

achieved for the remainder of the run for that value of ω. The time limit is also appropriate for the problem. Given the time taken for the physical process of loading using a forklift truck there is clearly no pressing need for a result in a few seconds, but equally we cannot afford to wait several hours. A limit of ten minutes is therefore a good reflection of user needs.

In order to assess the efficacy of the entropy directed aspect of MCTS, the experiments were repeated for different values of the entropy weighting parameter ω introduced in (7.1). The value of ω was varied from 0 (for completeness) to 1 i.e. the selection of the next stack to be placed during container loading was varied from being completely random to being weighted towards a low-entropy choice. Initial experiments revealed that values very close to, but less than 1, produced good results so we emphasised that region in the full tests. To give an external comparison, the performance of a deterministic packer, the skyline algorithm [Wei et al., 2011], was also evaluated. This algorithm is deterministic

but highly dependent on the input ordering. In order to show the best results that this algorithm could achieve we presorted the input files to group all the similar sized stacks together, ordered from the largest. We chose this algorithm because its dependence on the input order would allow this simple pre-sorting process to force an ordered layout. This would provide a direct comparison to the entropy-driven method, not just in the ability to successfully fill the container but also in terms of the quality of the configuration that is produced.

Thus the overall set of experiments involved running the MCTS and the deterministic algorithm for each of 50 randomly chosen sets of stacks, for each of 13 different fill levels: i.e. 60%, 70%, 80%, 90%, 91%, 92%, 93%, 94%, 95%, 96%, 97%, 98% and 99%, and in the case of the MCTS, for an additional 9 different weightings of entropy directing the MCTS, i.e., $\omega = 0, 0.5, 0.9, 0.95, 0.96, 0.97, 0.98, 0.99$ and 1. In some of the following tables and graphs of results, not all of the entropy weightings may be shown, as it was found that when the fill level was high, MCTS with lower values of ω failed to find any configuration that packed all the stacks into the container within the stopping time, and so provided no results. In addition, for some of the higher fill levels considered, i.e., fill levels 95%, 96%, 97%, 98%, and 99%, no results were produced for either the deterministic algorithm or any of the values of ω considered for the MCTS.

7.5 Results

7.5.1 Overall Performance Comparisons

Table 7.1 shows the number of different random sets that were successfully processed for each algorithm variant at each fill level. All the algorithms found successful layouts for all fill levels up to 70%. At the 80% level, the variants of our algorithm with weaker entropy dependence, i.e. $\omega < 0.95$, began to fail on some datasets. It is worth noting that the skyline algorithm only succeeds at this fill level when the stacks are sorted in largest first order. We also attempted skyline with the stacks ordered smallest first, but its results were significantly worse, so these have not been shown. At the 90% level, there are 17 combinations for which none of the algorithms was successful. For this case, the entropy guided

Table 7.1: Overall success for 50 sets at each fill level

Fill Level	skyline	$\omega = 0$	0.5	0.9	0.95	0.96	0.97	0.98	0.99	1
60%	50	50	50	50	50	50	50	50	50	50
70%	50	50	50	50	50	50	50	50	50	50
80%	50	35	20	48	50	50	50	50	50	50
90%	17	0	0	0	5	7	13	22	30	17
91%	14	0	0	1	6	6	8	10	13	6
92%	1	0	0	0	2	1	5	5	8	1
93%	0	0	0	0	0	0	0	2	1	0
94%	0	0	0	0	0	0	0	0	1	0
95%	0	0	0	0	0	0	0	0	0	0
96%	0	0	0	0	0	0	0	0	0	0
97%	0	0	0	0	0	0	0	0	0	0
98%	0	0	0	0	0	0	0	0	0	0
99%	0	0	0	0	0	0	0	0	0	0

MCTS algorithm with $\omega = 0.99$ performed best. The version with $\omega = 0.98$ was next, whilst skyline and the $\omega = 1$ algorithm produced the same number of successful layouts. These were, however, not exactly the same set of layouts. The $\omega = 0.99$ algorithm was most successful, creating a valid layout in 30 out of 50 cases, including all the configurations for which skyline found a solution. For the 3 configurations for which $\omega = 0.99$ failed, $\omega = 0.97$ succeeded. In one of these cases, $\omega = 1$ also succeeded and in another $\omega = 0.96$ also succeeded.

At the 91% level, the skyline algorithm as well as all the entropy-guided MCTS algorithm with all the weightings except $\omega = 0.9$ and $\omega = 0.95$, processed fewer sets than at the previous level. The $\omega = 0.99$ algorithm was again the most successful, producing a valid layout for 13 of the 50 cases, followed by $\omega = 0.98$, which succeeded in 10 of the 50 cases. At levels higher than 91%, the total number of successful combinations processed drops significantly. For these, the most challenging cases, less than 5% of the total sets of stacks considered across all weightings for each fill level were able to produce any layouts successfully.

At 92%, the skyline algorithm and the versions of the entropy-guided MCTS algorithm with $\omega = 0.96$ and $\omega = 1$ managed to process only a single set of stacks. It should be noted that each processed a different set of stacks. The $\omega = 0.99$ algorithm again performed best, producing layouts for 8 of the 50 sets of stacks considered, followed by $\omega = 0.98$ and $\omega = 0.97$ which produced layouts for 5 sets of stacks. At 93%, only $\omega = 0.98$ and $\omega = 0.99$ are able to successfully produce layouts for 2 and 1 set of stacks respectively. Finally, at 94%, only $\omega = 0.99$ successfully processed a single set of stacks. For levels 95% up to 99%, no configuration of the skyline or entropy-guided MCTS algorithm was found that could successfully process any of the total sets of stacks considered.

7.5.2 Visual Comparisons

Figure 7.2 shows the layouts created by different algorithm versions for one of the 90% fill sets for which the skyline algorithm was successful. The leftmost image shows the output of the skyline algorithm and the remaining images, in order left to right, show the layouts created by our algorithm for $\omega = 0.97$, $\omega = 0.98$ and $\omega = 0.99$ respectively. The layout for $\omega = 1$ was identical to that for $\omega = 0.99$. When it works, the skyline algorithm produces a similar looking result to the high ω layouts. Although the skyline method with a sorted input list produced an apparently similar arrangement to our algorithm it is inflexible and could not find a solution for many of the high fill level sets. One of these is shown in Figure 7.3. The order from left to right is $\omega = 0.95$, $\omega = 0.97$, and $\omega = 0.98$. Here the layouts for $\omega = 0.99$ and $\omega = 1$ matched that for $\omega = 0.98$. The lower values of ω do produce less regular patterns, however, as can be seen from Table 7.1, this sometimes allows a layout to be found where the variant with a higher value of ω was unsuccessful. A specific example of this can be seen in Figure 7.4. This is the case where only $\omega = 0.96$ and $\omega = 0.97$ were successful. It is notable that the entropy value for $\omega = 0.96$ is 29.88 which is lower than the value for $\omega = 0.97$, which is 31.84. This is because all versions of the algorithm are searching for the lowest entropy, whereas the value of ω controls the amount of random variation within the search.

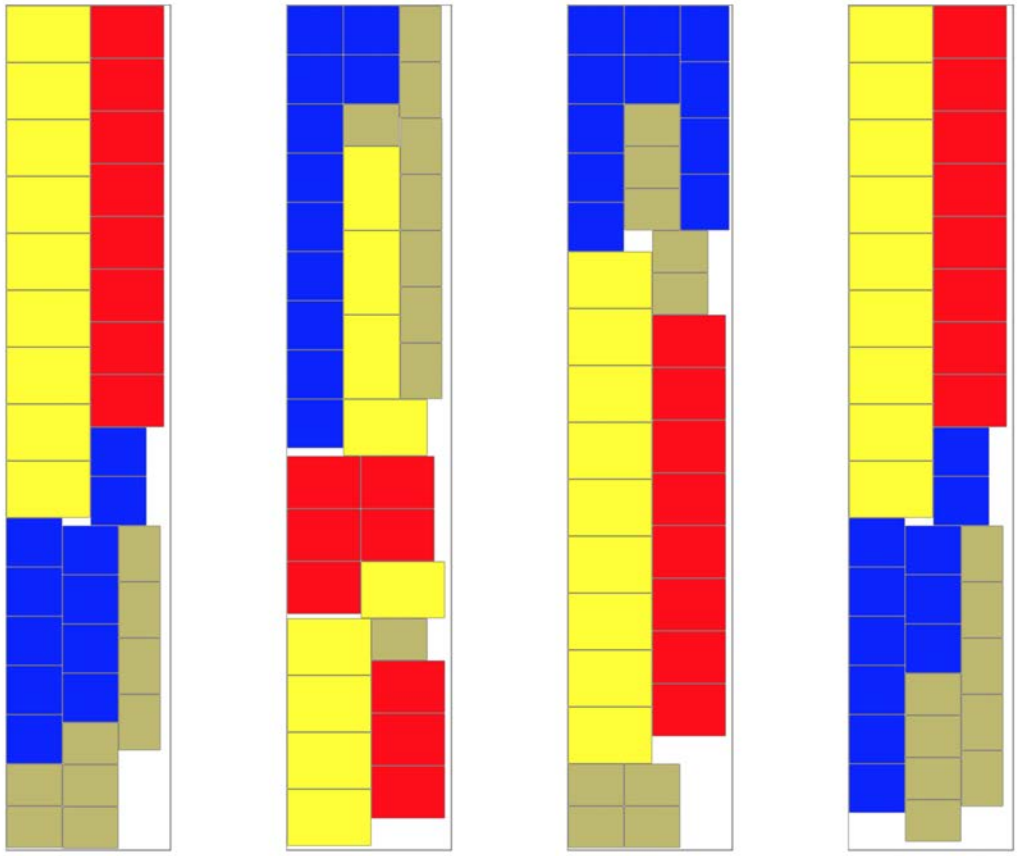


Figure 7.2: Comparison of layout methods at 90% fill, including (left to right): skyline algorithm, $\omega = 0.97$, $\omega = 0.98$, and $\omega = 0.99$.

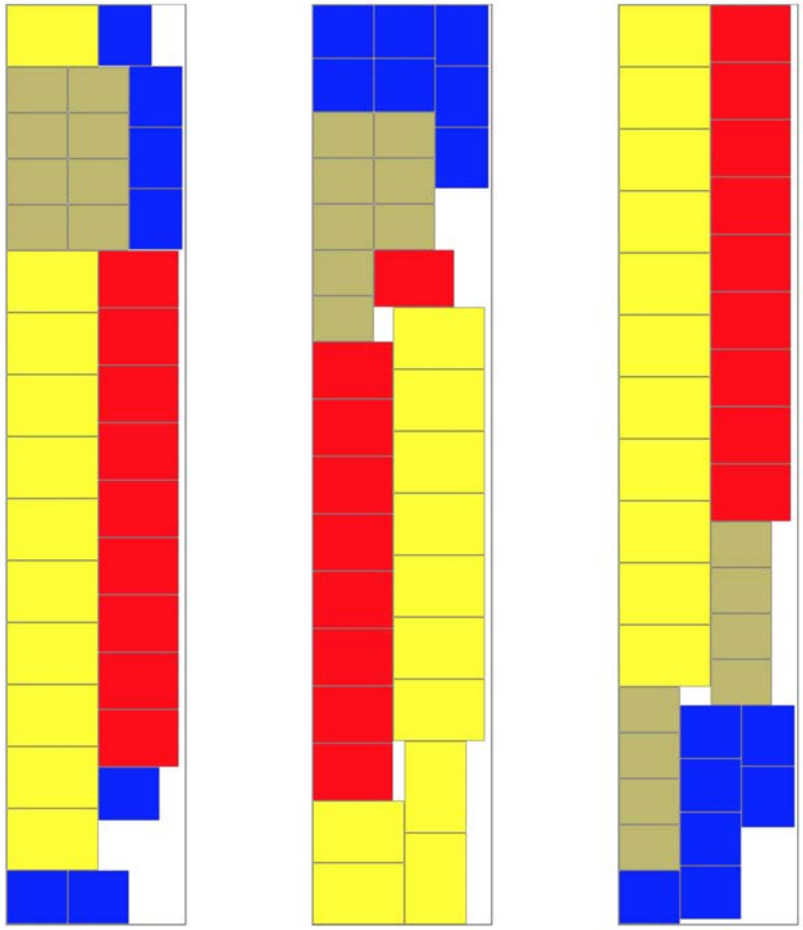


Figure 7.3: Comparison of layout methods 90% fill where skyline algorithm failed, (left to right): $\omega = 0.95$, $\omega = 0.97$ and $\omega = 0.98$.

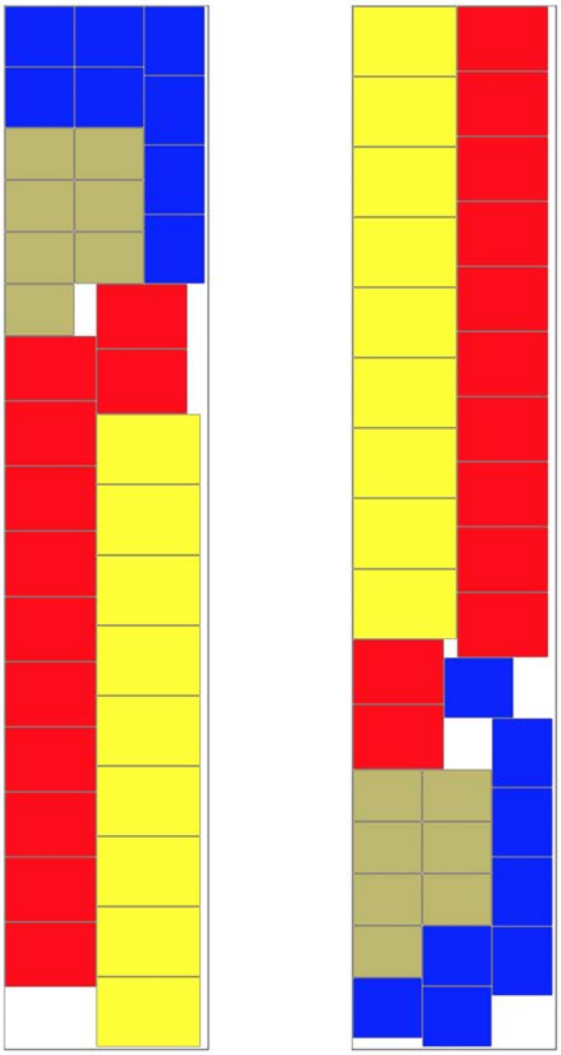


Figure 7.4: Comparison of layout methods 90% fill where skyline algorithm failed and only $\omega = 0.96$ (left image) and $\omega = 0.97$ (right image) succeeded.

Table 7.2: Which weighting generated most of the best entropy and length measures

Fill Level	weighting	$\omega = 0$	0.5	0.9	0.95	0.96	0.97	0.98	0.99	1
60%	Entropy	0	0	0	6	5	13	12	12	2
	Length	0	0	1	5	9	12	17	6	0
70%	Entropy	0	0	0	1	5	4	10	19	11
	Length	0	0	1	3	7	10	13	16	0
80%	Entropy	0	0	0	1	1	6	11	31	0
	Length	0	0	0	4	3	14	8	21	0
90%	Entropy	0	0	0	1	1	4	8	15	4
	Length	0	0	0	1	2	3	6	19	2
91%	Entropy	0	0	0	0	0	2	2	10	1
	Length	0	0	0	1	0	2	4	8	1
92%	Entropy	0	0	0	0	0	2	2	4	0
	Length	0	0	0	0	0	3	2	3	0
93%	Entropy	0	0	0	0	0	0	2	1	0
	Length	0	0	0	0	0	0	2	1	0
94%	Entropy	0	0	0	0	0	0	0	1	0
	Length	0	0	0	0	0	0	0	1	0

7.5.3 Layout Entropy

Table 7.2 shows how the best entropy and the best length achieved occurred at different values of ω for the individual sets of stacks. The best length achieved gives an indication of the tightness of the fill when the fill level was relatively low. So, for example, at a fill level of 70%, 1 of the set of 50 randomly chosen sets of stacks achieved its best entropy value when the weighting applied to the MCTS was 0.95, and 5 of the same set of stacks achieved their best entropy values when the weighting applied to the MCTS was 0.96. The table shows that most of the layouts achieved their best entropy values when the weighting was 0.98 or 0.99. A similar result is apparent for the best-achieved length usage. This confirms the previous conclusion that values of ω just below 1 performed best.

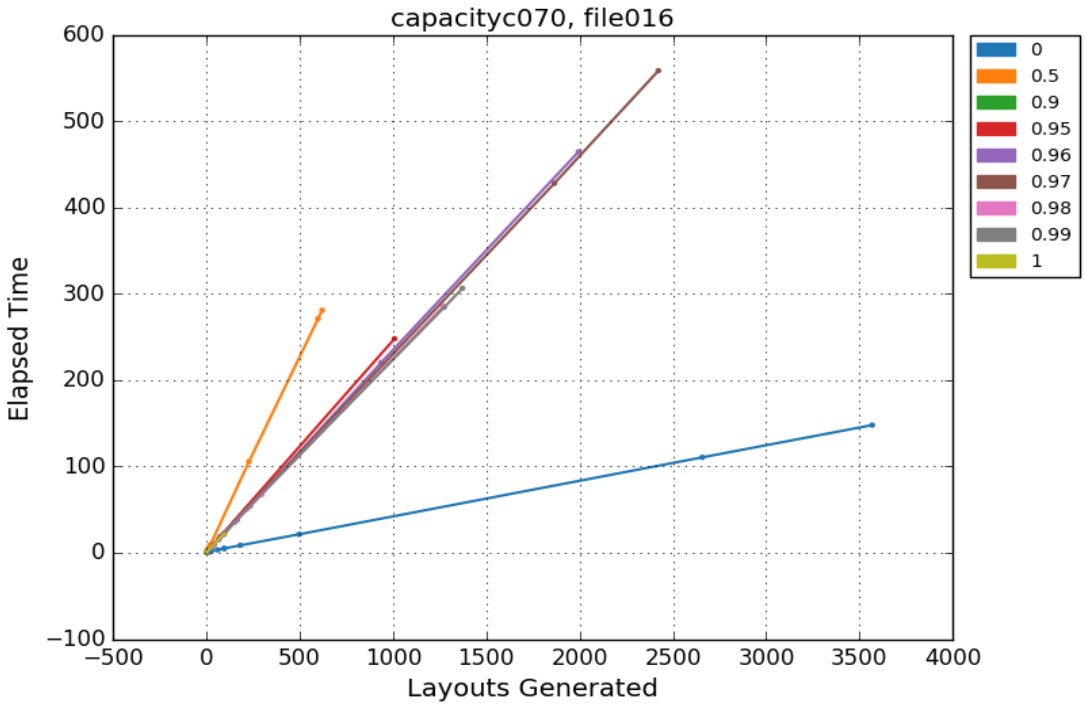


Figure 7.5: Times to generate individual layouts for a single set of stacks

7.6 Analysis

7.6.1 Time Behaviour

Figure 7.5 shows the generation of layouts over time during the course of an experiment for a single set of pallet stacks chosen for a fill level of 70%, and for a range of entropy weightings. Although the run for a weighting of 0 was faster, as it did not have to calculate the entropy, in terms of the overall timescale of the experiments this was not significant, and the container layouts produced, as will be seen later, were inferior.

7.6.2 Layout Progression

Figure 7.6 shows the typical progression of the layouts generated for a single set of pallet stacks that have been selected to cover 60% of the container floor area. The y axis indicates the best entropy level achieved while the x axis indicates the

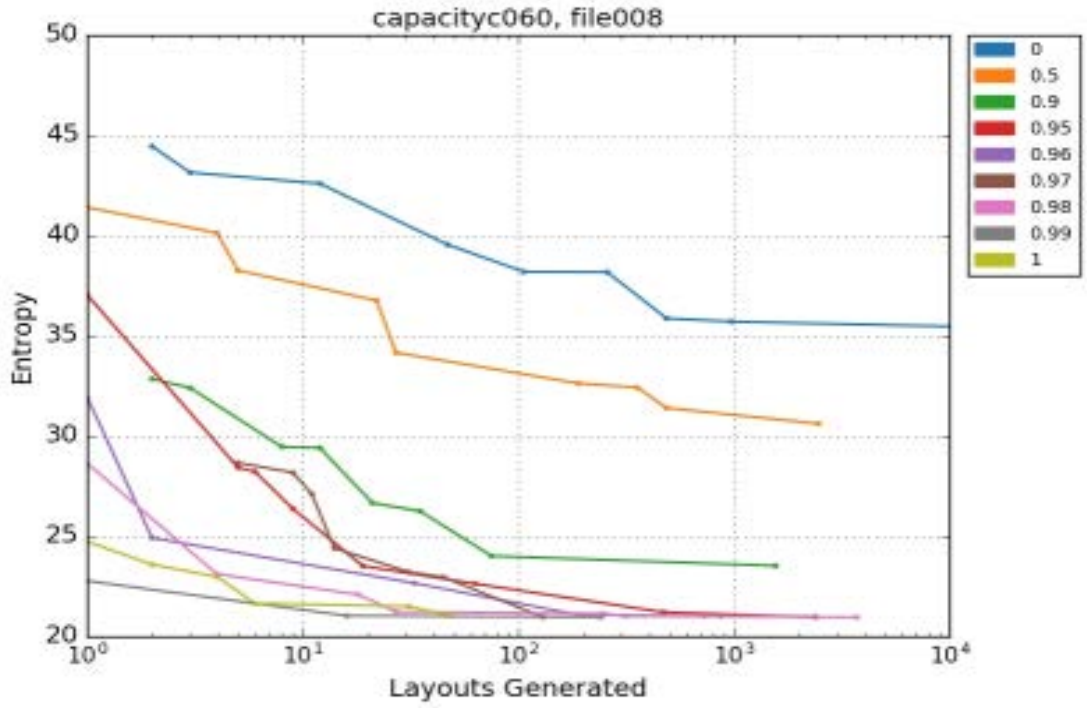


Figure 7.6: Best entropy layout generation for a single set of stacks, 60% fill

number of layouts generated and processed to produce that entropy level. Note that when the next stack to be chosen is heavily biassed towards low entropy choices, the overall entropy achieved for the layout is soon far below that of the weakly or unbiased choice. However, the purely deterministic version of the algorithm, with $\omega = 1$, is not always the best choice, in that sometimes the ability to randomly choose the next stack achieves the lowest overall entropy sooner (see Figure 7.7).

7.6.3 Further Discussion

Table 7.3 summarises the results of the experiments in terms of the best entropy achieved over the random selection of stacks for each of the notional fill levels, and for different weightings of entropy used to direct the MCTS. Lower entropy values indicate a more desirable layout. Note that for a fill level of 90%, no successful layouts were achieved when the entropy weighting (ω) was less than 0.95. In other words, the stacks could not be arranged such that they could all fit

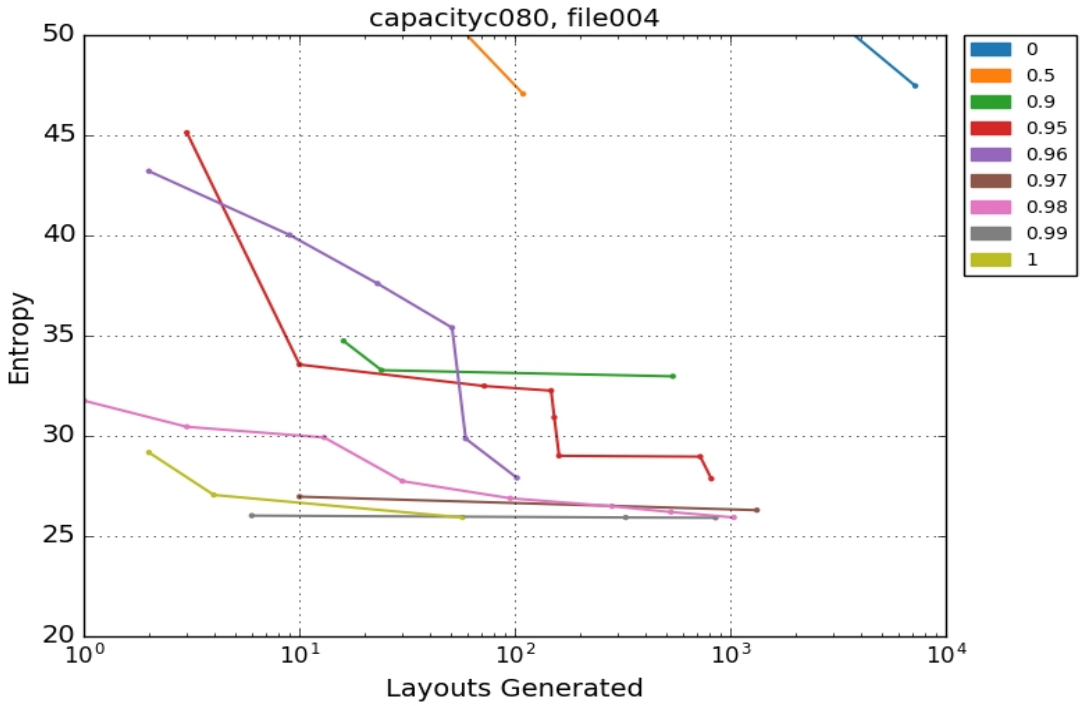


Figure 7.7: Best entropy layout generation for a single set of stacks, 80% fill

in the container, even though their total floor area was 90% of the container floor area. However, for higher weightings, the entropy direction enabled the MCTS to find ways to fit the selected stacks into the container. The mean and standard deviation show that there is considerable variation of the best entropy that can be achieved within the 50 randomly chosen sets of stacks.

Table 7.4 summarises the results in terms of the best length usage achieved for the container; that is, the value for the layout that occupied the lowest proportion of the length of the shipping container. These values were recorded during the experiments, though the entropy was always used to direct the MCTS. The length measure shown here is the length of the container in cm which has no stacks in it - the overall container length is 12.03 metres. Thus in this table, a higher value indicates more space left at the end of the container. Figure 7.8 shows the typical progression of length utilisation as the layouts are generated by the algorithm.

As the tables show that the variation in the best entropy and best length was quite large across the randomly chosen sets of stacks, we also looked at the way

Table 7.3: Best entropy values achieved for each fill level and weighting in MCTS

Fill Level		$\omega = 0$	0.5	0.9	0.95	0.96	0.97	0.98	0.99	$\omega = 1$
60%	Best Value	26.57	24.64	19.66	19.5	19.5	19.5	19.5	19.5	19.54
	Mean	38.89	35.98	28.29	26.01	25.48	24.64	23.96	23.4	22.51
	std	4.57	4.28	4.02	3.62	3.44	3.04	2.81	2.42	1.94
70%	Best Value	35.25	30.12	24.64	23.09	22.28	22.31	22.28	22.28	22.47
	Mean	46.27	42.5	33.25	30.17	29.41	28.62	27.84	26.67	25.87
	std	4.33	4.05	3.96	3.44	3.37	3.37	3.12	2.35	1.7
80%	Best Value	42.92	39.7	28.69	25.56	25.26	25.09	25.09	25.09	25.21
	Mean	50.7	45.68	36.63	33.75	33.12	31.87	31.24	30.39	29.54
	std	3.78	2.88	3.66	3.68	3.4	3.06	2.92	2.7	1.99
90%	Best Value	N/A	N/A	N/A	30.74	29.88	29.71	29.03	29.71	30.51
	Mean	N/A	N/A	N/A	34.57	33.15	33.09	33.23	32.47	32.1
	std	N/A	N/A	N/A	3.31	1.73	2.68	2.32	1.44	1.19
91%	Best Value	N/A	N/A	32.99	31.57	31.80	30.61	29.62	29.62	29.62
	Mean	N/A	N/A	32.99	33.42	35.14	32.76	32.27	31.60	31.42
	std	N/A	N/A	0.0	1.19	2.81	1.87	1.60	1.37	1.37
92%	Best Value	N/A	N/A	N/A	31.14	31.29	29.87	29.74	28.93	30.89
	Mean	N/A	N/A	N/A	32.11	31.29	32.52	31.71	32.90	30.89
	std	N/A	N/A	N/A	0.96	0.00	1.70	1.19	2.53	0.00
93%	Best Value	N/A	N/A	N/A	N/A	N/A	N/A	32.60	35.45	N/A
	Mean	N/A	N/A	N/A	N/A	N/A	N/A	33.09	35.45	N/A
	std	N/A	N/A	N/A	N/A	N/A	N/A	0.50	0.00	N/A
94%	Best Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	31.66	N/A
	Mean	N/A	N/A	N/A	N/A	N/A	N/A	N/A	31.66	N/A
	std	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.00	N/A

Table 7.4: Best length usage achieved for each fill level and weighting in MCTS

Fill Level		$\omega = 0$	0.5	0.9	0.95	0.96	0.97	0.98	0.99	$\omega = 1$
60%	Best Value	348	338	397	387	396	403	405	402	383
	Mean	116.54	100.80	133.99	149.67	156.27	169.07	182.44	189.58	142.96
	std	97.97	83.36	107.02	114.56	116.72	119.54	118.18	118.43	98.86
70%	Best Value	183	125	211	241	246	254	273	267	189
	Mean	42.75	38.40	61.17	69.73	76.30	77.11	84.52	85.19	59.41
	std	34.25	27.60	48.72	53.11	57.12	59.43	60.56	60.21	37.20
80%	Best Value	71	50	89	124	116	120	138	148	80
	Mean	19.09	13.10	28.54	29.09	31.13	33.74	31.92	34.41	28.83
	std	16.05	11.42	21.81	22.93	24.04	24.28	24.87	25.69	19.13
90%	Best Value	N/A	N/A	N/A	22	24	31	34	54	25
	Mean	N/A	N/A	N/A	8.00	9.00	7.90	8.49	12.26	8.77
	std	N/A	N/A	N/A	8.46	7.28	7.96	7.47	10.62	7.12
91%	Best Value	N/A	N/A	9	23	18	33	46	43	33
	Mean	N/A	N/A	9.00	10.67	6.67	10.22	14.00	18.92	13.67
	std	N/A	N/A	0.00	6.85	6.47	8.68	10.91	11.28	9.85
92%	Best Value	N/A	N/A	N/A	13	12	36	34	36	N/A
	Mean	N/A	N/A	N/A	12.50	12.00	17.17	15.17	14.73	N/A
	std	N/A	N/A	N/A	0.50	0.00	10.99	11.02	10.54	N/A
93%	Best Value	N/A	N/A	N/A	N/A	N/A	N/A	3	2	N/A
	Mean	N/A	N/A	N/A	N/A	N/A	N/A	3.00	2.00	N/A
	std	N/A	N/A	N/A	N/A	N/A	N/A	0.00	0.00	N/A
94%	Best Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	5	N/A
	Mean	N/A	N/A	N/A	N/A	N/A	N/A	N/A	5.00	N/A
	std	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.00	N/A

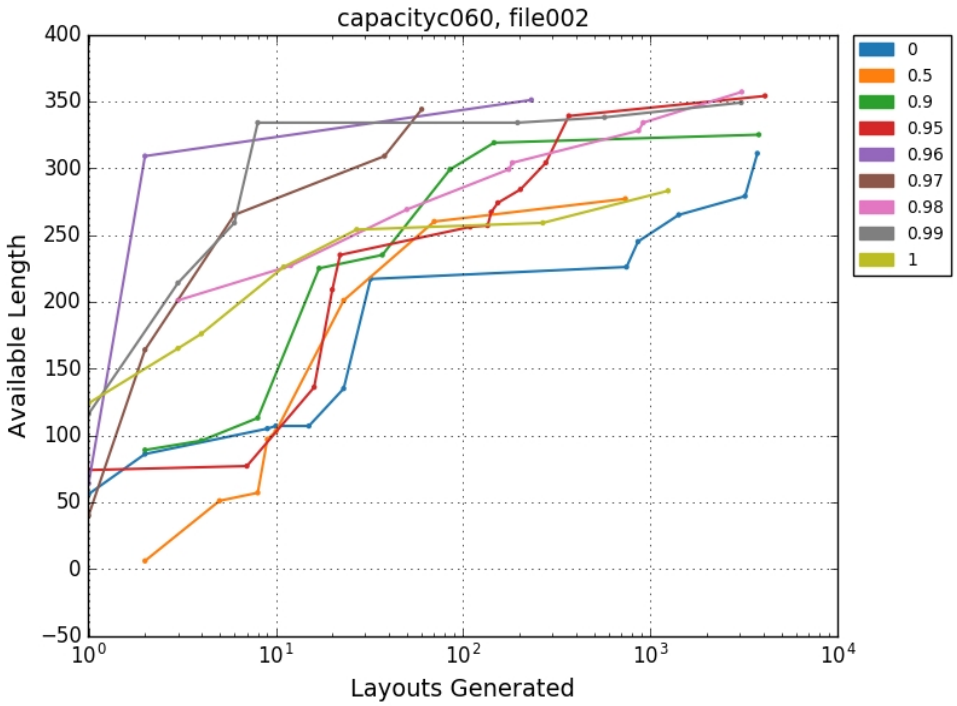


Figure 7.8: Best length utilisation layout generation for a single set of stacks, 60% fill

the different weightings contributed to the best values on a set by set basis.

For further comparison of the effect of the different weightings, consider the three example layouts shown in Figures 7.9, 7.10 and 7.11. Figure 7.9 shows a layout generated at the start of the process for a particular set of stacks using an entropy weighting of 0 i.e. the MCTS was undirected. Note that even though the stacks do fit within the container, they are very disorganised (with an entropy of 50.6). After a further 11 857 layouts had been generated for the same set of stacks (and still using $\omega = 0$), the best entropy recorded had gone down to 32.8, but the layout was still complex, as shown in Figure 7.10. These two may be compared with the layout shown in Figure 7.11, which was generated from the same set of stacks, but with a value of 0.98 for ω. It can be clearly seen that the third layout is highly ordered and compactly fitted into the container, with the entropy value down to 22.1 after processing only 51 layout attempts.

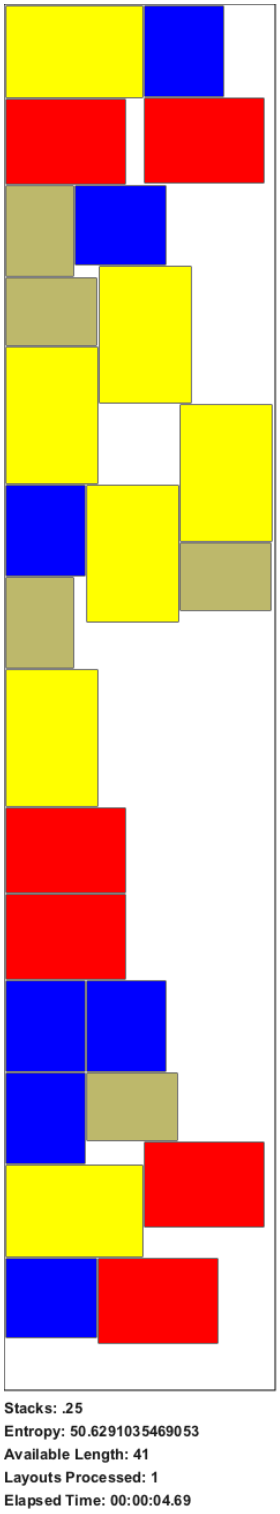


Figure 7.9: Layout after 1 cycle, 60% fill, entropy weight 0

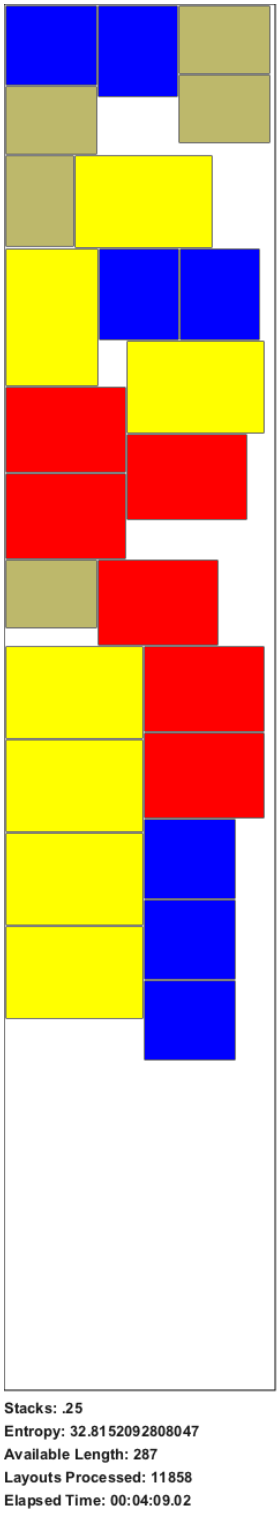


Figure 7.10: Layout after 11858 cycles, 60% fill, entropy weight 0

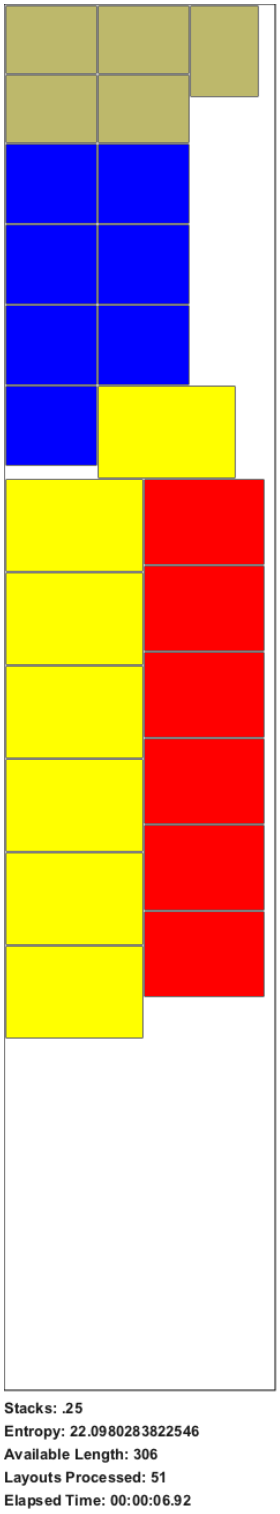


Figure 7.11: Layout after 51 cycles, 60% fill, entropy weight 0.98

7.7 Conclusion

In this chapter, a new entropy-based approach to solve the problem of generating feasible layouts for the single container loading problem was presented. The approach uses a Tetris-like placement method that ensures that generated layouts can be easily implemented thus allowing for the safe and easy loading of palletised goods by warehouse operatives using forklift trucks to load from a single entry point. The choice of what stack to load next and in what orientation to load it in is driven by a Monte Carlo tree search process and a weighting that can be used to bias the choice towards lower entropy layouts i.e. layouts that are easier to understand and implement (see Chapter 6). The generality of the approach makes it suitable for dealing with container loading configurations where the types of pallets used are not known beforehand.

The series of experiments used to evaluate the entropy-driven Monte Carlo tree search algorithm, as well as an analysis of the results obtained, were also presented. The results show that the layouts produced with a weighting value closer to 1, i.e. strongly biased towards low-entropy choices, were more ordered than those where the weighting value was closer to 0. The weighting values 0.98 and 0.99 in particular, accounted for a majority of the layouts with the best entropy values, particularly in the situations where the packing density was high. Together, they accounted for 70% of the total layouts produced when testing the packing of stacks with a 90% fill capacity, 46% for the 91% fill capacity, 59% for the 92% fill capacity, and 100% for the 93% fill capacity.

While it was faster to generate layouts using a weighting of 0 because no entropy calculation is performed, the difference was not particularly significant when considering the overall timescale of the experiments. As alluded to earlier, the layouts produced using this weighting, $\omega = 0$, were less inferior to those produced with the weighting values closer to 1. At the other extreme, using a weighting value of 1 did not perform as well as expected. The observed results indicate that this is due to a lack of the random variation present in the other weighting values (where $0 \leq \omega < 1$). This ability to occasionally randomly select stacks and their orientations during layout generation often resulted in the algorithm reaching an overall lower entropy sooner than with a purely deterministic

'lowest-entropy' selection mechanism, i.e., where $\omega = 1$.

Overall, the results show that the algorithm is viable and can be used to produce good 'low-entropy' layouts in a very reasonable time. While the layouts generated might seem obvious to the reader, this is the very thing I hope to have achieved: to have produced a generalised approach implemented on a computer that produces container layouts similar to those produced by expert human operatives with their knowledge of the limitations of the environment. Experts at generating layouts may not always be available and can make mistakes, so an algorithm that can achieve equivalent outcomes in a short time is desirable.

Chapter 8

Conclusion

8.1 Context

The Container Loading Problem (CLP) is an active research area with numerous real-world applications, particularly in the container transportation and distribution industries [Dereli and Sena Das, 2010]. Research in container loading, however, is still in its infancy with respect to the inclusion and satisfaction of several practically-relevant constraints simultaneously; especially when compared to the body of work available where very few or no practical constraints are considered. This is in contrast with real-world applications of the CLP where typically all constraints considered must be satisfied in order for provided solutions to be deemed feasible. In this regard, the current literature is still lacking.

This thesis examines a version of the CLP motivated by a real-world problem experienced in the UK distribution centre (UKDC) of an engineering company. In solving the problem, a number of standard and non-standard approaches for solving the CLP are employed and several constraints, i.e. orientation, weight limit, stability, stacking, complete shipment and pattern complexity, are also simultaneously dealt with. The overall approach presented is a hybrid heuristic that resembles the existing process used to manually solve the problem in the UKDC. This approach selects, stacks and packs items in different stages, ensuring that all required constraints are satisfied at each corresponding stage. The packing algorithm employed packs in a manner that results in packing patterns that are easy to understand and reproduce by human loaders. Overall, the de-

vised hybrid algorithm has resulted in a consistent and significant increase in observed container weight utility for loading problems and runs in a fraction of the time required for a human solution to be provided. It also produces loading patterns that make the loading process simple and easy to follow; a feature that is of great benefit to less experienced loaders.

8.2 Summary of Key Contributions

This thesis contributes to the understanding and application of a number of Computational Intelligence techniques to Container Loading Problems. The contributions are especially relevant to real-world applications of container loading in warehouse environments, particularly for cases where the items to be loaded into containers are heavy and palletised, and need to be moved around using forklift trucks. Following is a summary of the key contributions presented:

- An algorithmic framework, that takes the approach of decomposing the CLP into sub-problems in order to solve it as a whole, is presented. This framework allows for algorithmic hybridisation wherein problems to be solved can be decomposed into sub-problems with each solved using any number of exchangeable algorithms, as long as any imposed constraints are satisfied. The initial set of algorithms introduced in this framework are: (i) a genetic algorithm for the selection sub-problem, (ii) a problem-specific greedy algorithm for the stacking sub-problem, and (iii) a genetic algorithm integrated with a rectangle packing algorithm, subsequently replaced by a Sort-and-Pack Cygon algorithm, and then by the derived Entropy-guided Monte Carlo tree search algorithm. These algorithms were evaluated using real-world historical data and simultaneously took into account all relevant practical constraints.

- Taking influence from physics and information theory, an approach based on a derived entropy measure is introduced for the identification of feasible container layouts. This measure provides an indication of which layouts are practical and easy to load, particularly when using forklift trucks.

- A novel method of directing a Monte-Carlo tree search process using entropy, during the process of generating layouts, is presented. The method makes use of Tetris-like placement, using an entropy-weighted value to bias the choice of what item to select and how exactly to place it, during a packing operation. This enables it to produce layouts with high-density packing, comparable to those produced by expert human loaders, that can be easily understood and reproduced by human loaders using forklift trucks. The layouts are produced in very reasonable time and have the additional advantage of being easily generalised to include other practical constraints. As experts may not always be available, this is a very desirable outcome.

- The variant of the CLP dealt with in this thesis is characterised by its weakly heterogeneous palletised goods, which are heavy and need to be moved around using forklift trucks. These characteristics impose additional constraints to those typically considered in the literature. The resulting combination of constraints can indeed be said to make the considered problem unique. A novel approach for solving this specific CLP variant optimally, that simultaneously satisfies all identified constraints, in order to provide a feasible real-world solution, is presented.

- Data that extends the existing container loading benchmark data is provided. The existing data typically only covers weakly or strongly heterogeneous problem instances that deal with relatively few practical constraints at a time. The presented data is representative of real-world problem instances that consider a larger number of constraints at a time. These additional constraints reflect a wide spectrum of practical applications that have not yet been dealt with extensively in literature.

- An approach that integrates the above contributions, i.e., the complicated algorithms and data, into an engaging system that presents results in a manner that can be easily understood and interpreted by humans, is also presented. Continued use of this system brought about some unintended consequences, including its use as: (i) a tool to verify and check if human-crafted layouts were feasible; (ii) an environment for training warehouse

operatives with little or no loading experience, thus minimising possible damage to goods in real-world training; (iii) a means to discover new loading patterns that have not been used before in practice. These use-cases, in addition to the system's original intended purpose, had a positive effect on the adoption of the container loading system that served as an abstraction for the algorithms presented in this thesis.

While these contributions were a direct result of a specific case study in practice, the generality of the approaches presented makes them suitable for dealing with container loading problems with configurations different from the one studied.

8.3 Future Work

The following points are suggestions for further work that can be carried out to extend the research presented in this thesis.

8.3.1 Solving the Multiple Container Loading Problem

The algorithm presented in this thesis was designed to solve the Single Container Loading Problem (SCLP). No attempt was made in particular to solve the related Multiple Container Loading Problem (MCLP). That said, the algorithm can be modified and extended to solve the MCLP using an approach known as the 'sequential' approach in MCLP literature [Eley, 2002; Lim and Zhang, 2005]. The approach repeatedly applies the algorithm to a given set of pallets, solving the SCLP each time and filling containers sequentially one after the other. The pallets that are already packed into containers are removed from the original set of pallets. Hence subsequent applications of the algorithm make use of a smaller set of pallets. This continues until all the pallets have been packed. The caveat of this approach, however, is that there will almost always be one container that has not been fully utilised. In practice, particularly in the use case at the UKDC, this is not necessarily a problem. Left-over pallets are often held back until there are more pallets available to pack so that a new container can be fully utilised. If the pallets need to be sent out urgently though, a smaller container is used or a courier that can handle the load is booked. If the left-over load was held

back until there are more pallets available, the held back pallets will typically be marked as having a higher shipment priority so that when a new container is made available, they will be packed before any of the newly added pallets.

8.3.2 Dealing with Loading Priorities

When loading priorities are implemented, certain items must be loaded before others. This could occur as a result of a deadline placed on the delivery of certain items. In this example, items with a nearer delivery deadline will be loaded and shipped out before items with a farther deadline. Handling loading priorities during loading can be dealt with using an approach that also makes use of the repeated application of the algorithm presented in this thesis. This time, instead of applying the algorithm to all of the available pallets, the algorithm is first applied only to pallets with the highest loading priority, and then applied to pallets with lower loading priorities. Pallets with lower loading priorities are only considered when all of the highest priority pallets have been loaded and there is still space in the container. This process is repeated until all the pallets are loaded in order of loading priority, or until the container is filled up at some point. At the point when the container is filled up, it is guaranteed that none of the pallets loaded so far will have a lower priority than any of the pallets left behind.

8.3.3 Keeping groups of related items together in close proximity

In real-world container loading, there are times when groups of items must be kept and packed together in close proximity within a container. This requirement is an example of the 'relative' positioning constraint. We see examples of this in situations where items must be delivered to multiple locations, i.e., multi-drop loading. In such situations, the items meant for delivery to the same location are typically kept together when loaded into a container. This makes it easy to find and unload items destined for the same location. It also saves time and minimises the possibility of damage that could occur when the items to be unloaded at a specific location are scattered across the container such that a number of unrelated items have to first be unloaded and loaded back in order to reach them.

While this constraint is not explicitly handled by the presented algorithm, it can be dealt with by attempting to pack related item groups one group at a time using repeated applications of the algorithm. For the initial application of the algorithm, the entire container is considered as the packing area. After the first group of related items have been packed, the length of the container space used is subtracted from the original container length and the resulting container space left is used as the new packing area for subsequent packing. This process repeats for each subsequent application of the algorithm, and the container space available (i.e. the container length) is reduced at each turn until either all the items have been packed or there is no more space available for packing. Breaking the container down into sections/rows this way at each step ensures that items belonging to different groups do not get mixed up with each other. Using this approach, there is an expected loss in packing space efficiency typical with packing methods, e.g. the 'Shelf Next Fit' packing algorithm, that prematurely close up rows/shelves in the container during packing. In practice, however, keeping the related item groups together during packing is seen as being more important and will often take preference over fitting in more items. This is also reflected in the literature [Christensen and Rousøe, 2009; de Queiroz and Miyazawa, 2013; Junqueira et al., 2012a], where this constraint, the multi-drop instance of the positioning constraint, is mostly treated as a hard constraint.

8.3.4 Extending the application of Gamification

The current gamification setup can be extended to involve the setting up of a scoring system and the implementation of a high scores table for the interactive simulation environment. This should leverage the natural human desire for competition to increase user engagement with the system. Initial experiments already revealed the existence of a friendly competition amongst loaders, with individual loaders often wanting to know how other loaders perform when laying out particular container loads. Loaders trying to best each others' scores should retain an increased level of engagement while sustaining the friendly competition.

As gamification is an ongoing process that should be constantly evolved over time to improve the nature of the interaction with users [Zichermann, 2011], a

system/framework could also be put in place to enable the continuous capture and analysis of data such as: (i) how easy it is to use the system; (ii) how effective the learning experience is; (iii) how much faster an inexperienced loader learns using the gamified system compared to the traditional means; (iv) how inexperienced loaders' performance in the gamified system compares to that of experienced loaders; (v) how much performance obtained in the gamified system reflects actual real-world performance; and (vi) how much correlation there is between loading performance in the gamified system and loading performance in practice. This should help refine the user engagement process and ensure that the system has a direct impact on the users, ultimately resulting in an increase in the performance of the loaders in their day-to-day loading activities.

8.3.5 Improving and extending the entropy measure

In the current method for calculating the overall entropy measure for a container layout (see Chapter 6), the edge weights of the graph representing all the connected items in the layout are what determines the shape of the resulting entropy tree. In this implementation, the edge weights are calculated as the combined sum of the selection, rotation and positional entropies. There is room for exploring different implementations that could result in a different shaped entropy tree e.g. the use of the individual entropy values or a combined sum of any two of the different entropy values as the edge weight. These different implementations will result in different calculated edge weights which will, in turn, result in different overall entropy values. Further experiments would need to be performed to determine if any of these methods are acceptable and produce better results than the current implementation.

Other thoughts for extending the measure include: (i) significantly reducing the overhead involved in calculating the entropy relationships between items in the entropy graph by only connecting items to their nearest neighbours rather than to all the items present, as items that are farther away add a higher distance entropy value to the edge weight resulting in a low probability of being selected during the entropy tree generation; (ii) exploring and giving further thought to the possibility of the inclusion of weight distribution considerations in the

calculation of the entropy measure, which might involve some sort of weight ratio calculation for an item relative to the weight of its nearest neighbours; and (iii) tweaking the measure to account for and reflect the length of layouts, as the current implementation sometimes results in layouts with different lengths having the same entropy measure. As with the initial suggestions above, these will also need to be tested extensively to determine their suitability for use as a feasible measure in practice, while keeping in line with the original intended purpose for the measure.

8.3.6 Extending the entropy-driven Monte Carlo search to address additional constraints

As the novel approach proposed for guiding Monte Carlo search via the entropy criterion lends itself to generalisation, further constraints that are pertinent to real-life situations can be included. Other practical issues such as the weight distribution of goods laterally and longitudinally across the floor of a container and legally enforced axle weight limits, can be addressed. Directed Monte Carlo search provides a means to weight the chosen pallets by multiple criteria, so that the resultant layouts achieve the optimal combination of characteristics. Generating guidance by the inclusion of the expert users' preferences or importance attached to these criteria will allow this. This inclusion of 'soft criteria' is an area that Bortfeldt and Wäscher [2013] consider has not been sufficiently explored. A further area that could be studied is the generation of 'stable' layouts that have very minimal or no lateral motion during container transportation in order to prevent (or reduce) potential damage to goods during transit.

Appendix A: Applying Gamification principles to the Container Loading Problem

A.1 Introduction

Gamification is a phenomenon that has in the last few years garnered a lot of attention with numerous applications particularly focusing on productivity and health fitness. It is defined as the use of game design elements in non-game contexts [Deterding et al., 2011a] and is mostly introduced into a system to increase user experience and user engagement [Deterding et al., 2011b], or to act as the means of actual user engagement where there is none. The increase in experience and engagement is considered to be the result of the effects obtained when leveraging peoples natural desire for learning and accomplishment.

In this chapter, we discuss the application of the principles of gamification to the container loading system used to assist warehouse operatives during container loading in the UKDC [see Section 3.1]. We discuss the effects gamification has on the adoption of the container loading system, and show a systematic build-up of trust and familiarity over time of the system by the operatives. This increased user engagement with the system which lead to an increase in system adoption.

We then propose a fully gamified system as an abstraction that provides an interactive environment for the engagement of warehouse operatives with the underlying complicated algorithms that solve the container loading problem. The container loading system used in this context refers to the hybrid algorithm described in Chapter 4, along with all the elements provided to make the algorithm accessible to the warehouse operatives so that they are able to interact with it.

A.2 Background

The introduction of information technology systems into the workplace to increase business performance is not a new idea and has its pros and cons. Recent trends show the application of gamification in this context as a means of increasing and retaining user engagement with the introduced information technology systems. In solving the problem faced by the UKDC, we introduced a computerised system and applied gamification elements to it. Our intention was to increase (and retain) user engagement with our introduced system, as well as to increase the overall system adoption.

The UKDC's problem (described in detail in chapter 3) can be summarised as that of optimally selecting and loading groups of palletised goods onto containers. To solve this problem optimally, the UKDC have invested in research towards a computerised loading optimisation system in a bid to:

- increase overall loading speed;
- reduce the cost of hiring containers by optimally maximising the capacity of every loaded container to the reduce overall number of containers used for loading;
- reduce damage to goods that might occur because of non-optimal packing in the container, therefore reducing costs that might arise from replacing damaged goods, or customer fines for the receipt of damaged goods;
- provide greater customer satisfaction by speedily processing and loading customer orders for safe and prompt delivery, and;
- increase warehouse throughput: the more goods that are loaded and sent out from the warehouse, the higher the warehouses capacity to process new customer orders with the existing space, which could lead to more business for the company;

which should have the overall effect of significantly improving business performance and raising the competitive edge of the UKDC while providing greater customer satisfaction.

Appendix A: Applying Gamification principles to the Container Loading Problem

As mentioned briefly earlier, we devised a container loading system for the UKDC that comprises the devised hybrid algorithm described in chapter 4 and a number of UI (user interface) elements that support and allow for interaction with the hybrid algorithm. These UI elements start off as simple and plain functional elements, that subsequently get upgraded to more engaging elements in order to encourage and increase interaction with the system. This progression and change in the UI elements, an example of which is a change from a purely textual output from the algorithm to graphical colour-coded aesthetically pleasing container layouts, is described in the subsequent sections of this chapter.

The initial container loading system produced its output as plain textual data (see Figure A.1), with numbers tersely showing item dimensions, weight, group membership, and coordinate point locations. The output from the system was difficult for the operatives to interpret and understand. Hence, the reception of this initial system by the warehouse operatives was negative. There was also resistance to the idea that a computer system could produce 'optimised' loading solutions; the popular belief was that a computer could not deal with the complexities involved in satisfying all necessary practical constraints while producing the container layouts. The system, if it worked, was also seen as something that would take over the more fulfilling aspects of the jobs of the warehouse operatives who currently manually work out the selection of pallets for loading and plan the layouts for the selected pallets in preparation for loading.

Results obtained from initial experiments (see Section 4.5) showed solutions that consistently achieved 100% container weight utilisation. These results, compared to the average of 85% utilisation obtained manually across historical loading data, only helped to fuel the already uneasy feelings towards the system.

What these tests did not show at the time was the flexibility that could be obtained from the solutions provided by the loading system and how these solutions could greatly complement a warehouse operative's experience; it was only later, after the addition of gamification principles to the system, that these factors became apparent.

```
Best Solution:
---------------
Selected Groups: 0002, 0004, 0005, 0015, 0029
Total Weight: 25948kg
Summary (54 items): E-TYPE (12), S-TYPE (22), N-TYPE (20)
GROUP0002/00001, W: 294kg, LBH: 80/70/74, STK:0
GROUP0002/00002, W: 592kg, LBH: 105/75/71, STK: 1
GROUP0002/00003, W: 391kg, LBH: 80/70/92, STK: 1
GROUP0002/00004, W: 279kg, LBH: 80/70/72, STK:0
GROUP0002/00005, W: 401kg, LBH: 120/81/76, STK:0
GROUP0002/00006, W: 495kg, LBH: 105/75/69, STK: 1
GROUP0004/00001, W: 292kg, LBH: 80/70/58, STK:0
```

Figure A.1: Example text output from initial loading system

A.3 Related Work

Information technology systems have long since been introduced into the workplace to bring about an increase in business performance [Black and Lynch, 2001; Brynjolfsson and Hitt, 2000]. Studies show however that such introduction does not always guarantee a positive result [Debrabander and Edstrom, 1977; Majchrzak and Klein, 1987]. Recent trends show the increased introduction of elements from game design into business computing systems in order to increase user engagement and improve or guarantee the adoption of the system in question [Alcivar and Abad, 2016]. This phenomenon, of introducing gaming elements in a non-gaming context in order to increase engagement, is generally referred to as gamification.

As gamification research is still in its infancy, several varied definitions exist for it in literature: Deterding, Nacke, Dixon and Khaled in [Deterding et al., 2011a, p. 9] define it as "the use of game design elements in non-game contexts"; Sy, Zichermann and Cunningham in [Sy et al., 2011, p. ix] define it as "using game-thinking and game mechanics to solve problems and engage audiences"; Huotari and Hamari in [Huotari and Hamari, 2012, p. 19] define it as "a process of enhancing a service with affordances for gameful experiences in order to support user's overall value creation"; and Werbach and Hunter in [Werbach and Hunter, 2012, p. 26] define it as "the use of game elements and game-design techniques in non-game contexts". While the existing definitions might be inconsistent, a

standard is emerging that emphasizes the use of "game elements" in "non-gaming contexts". To this end, we identify with the definition of gamification as a process of incorporating game elements, for a specific purpose, into a system in order to maximise a user's experience and increase engagement with the system. The important point in this definition is the presence of a purpose; the game elements incorporated into a system must have a specific purpose if an improvement in user engagement and motivation is expected [Alcivar and Abad, 2016].

A highly cited example of the successful application of gamification is Foursquare, a location-based service that allows its users to check in at various locations using mobile devices. It used badges as a game element to leverage the desire of people to be connected and saw an increase in the user engagement of their service. Li et al. [2012] gamified a tutorial system to help new users learn AutoCAD. They employed gamification elements such as scoring: to provide feedback on performance, game levels: to provide a means of progression, missions: to provide a challenge, and rewards: to motivate users. They recorded an increase in engagement, enjoyment and performance among their users. McDaniel et al. [2012] introduced gamification through the use of badges, as a sign of achievement, into a learning management system to motivate students towards certain behaviours desired by teaching staff. They observed that feelings of connectedness and competition drove students to engage with the system and reported an increase in engagement. de Marcos et al. [2015] studied the effects of gamification on learning performance in an undergraduate course. Their results suggest a significant positive impact of gamification on learning performance.

In this appendix, we show our attempts at the incremental introduction of game elements, each to satisfy a pre-determined goal, to a decision support system for the sole purpose of increasing user engagement and changing the user's perspective towards the system. Indeed, we can say that this process of introducing game elements into systems to improve engagement and change behaviour is a common theme across all applications of gamification, as it is an integral part of its definition. Whether or not an increase in performance or engagement is achieved is another matter, but the main design goal of the application of gamification must be to cause such an increase.

A.4 Gamification Approach and Experiments

Based on the observed initial attitude towards the loading system, we realised early the need for a way to initiate and maintain user engagement with the system in order to increase its adoption. If the system adoption remained low, the system would be unable to make any impact that could cause any measurable effect on user or business performance.

The main goal was therefore to ensure an increase in the user engagement of the loading system. We identified from the literature that the application of gamification principles was a good fit for this goal, and we set about identifying areas in the underlying system that could benefit from such principles. Table A.1 shows the gamification sub-goals we set and the eight strategies we identified for tackling them. In the rest of this section, we discuss the implementation of some of these strategies and outline some of the observations made when the warehouse operatives were exposed to the resulting gamified system. The remainder of our observed results is discussed in the section A.5.

A.4.1 Conventions for visual container layout representation

Our first steps involved building a visualisation for the text data output of the loading system (Strategy 1). We also set up naming and colour-coding conventions (see Table A.2) to identify the different types of pallets available for loading. The naming convention used is based on established names familiar to the warehouse operatives, and the colours used are easily identifiable primary colours. The visual representation is provided as a container layout that shows the exact placement of colour-coded palletised goods within a container (see Figure A.2). In subsequent interactions with the loading system, all loading operation results were presented using this visual representation. Our observations of these interactions revealed that our conceived visual representation, while a step in the right direction, came across as rigid and final to the operatives. This observation informed the need for a more flexible interactive interface and became the basis for the identification and implementation of Strategy 2 and Strategy 7.

Table A.1: Gamification strategies and goals identified for the system

Problem	Goal	Strategy
System adoption is low and system output is dull, non-engaging and difficult to interpret	A. Engage users visually with an intuitive interface	1. Present an interface with intuitive loading representation that is easy for users to use and understand
	B. Retain user engagement and make system fun	2. Ensure the interface is simple and can make loading tasks fun 3. Provide loading 'challenges' that can be rewarded with special badges or trophies
	C. Encourage user learning, improvement and knowledge sharing	4. Implement a scoring system to leverage user competiveness which makes users want to do better than others at loading tasks 5. Provide repeatable tasks, which can be used in conjunction with score feedback to reinforce learning 6. Record completed user tasks that other users can easily access and learn from 7. Provide users with a way to interact with the results from the loading system in order to allow modifications that result in new solutions 8. Provide an interface that allows one to quickly and easily check if a particular load layout will fit in a container

Table A.2: Defined convention for layout representation

Pallet Type	Dimension Ratio	Colour
STD	12 x 8	Yellow
NSK	8 x 7	Blue
EURO	10.5 x 8	Red
EURO2	8 x 6	Green

A.4.2 An interface for interactive simulation

In other to provide an interface that would be fun and interesting (Strategy 2), we decided to build a simulation interface that would incorporate the same visual representation conventions we had previously defined (see Figure A.3). We made this interface accessible on a tablet because of its ubiquity and mobility; the idea being that the warehouse operatives would find it very familiar and easy to operate. We then presented the simulation interface in a manner that vaguely resembles the game 'Tetris'. Altogether, this puts a familiar interface in front of the complicated algorithms running behind the scenes in the loading system. As part of the interface design, and in keeping with our defined conventions, the interactive blocks used to represent palletised goods in the simulation were sized to scale and colour-coded appropriately. The end result was an interactive interface that allowed for easy modification of loading layouts in a simulated container (Strategy 7). We observed in subsequent user interaction a natural extension to the use of this interface that was not part of its intended design, being the use of the simulation interface to check if manual loading plans not generated by the loading system were feasible and could fit completely in the simulated container. This helped loaders check and reinforce their own loading knowledge. As part of our continuous evaluation of the system, this observed interaction helped further inform the gamification goals and became the basis for the identification and implementation of Strategy 8.

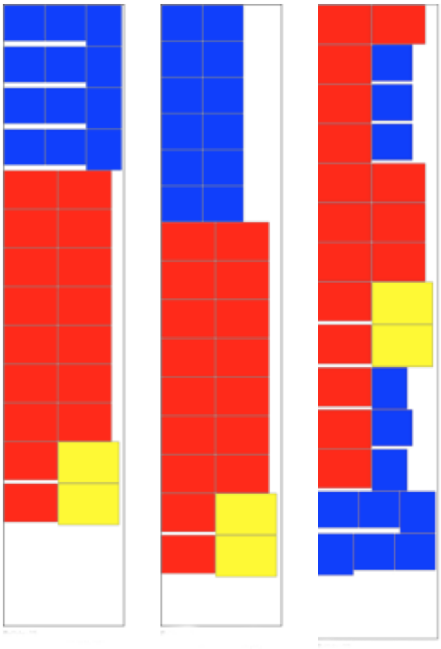


Figure A.2: Visual representations for loading system output

A.5 Results and Discussion

Our continuous observation of user interaction with the loading system throughout the entire gamification process was very informative. In fact, our observations of certain parts of the process directly informed further actions applied to other parts of the process.

As the conventions we introduced for the visual representation of the loading system's output were easy to understand and relate to, they were easily adopted by the operatives and internalised; this brought about an increased engagement in the loading system. This adoption provided a common vocabulary for the loaders to use to represent loading terms and made it easier for them to understand and relate to the output of the loading system. It also brought about easier communication between us, the designers of the system, and the loaders. Loading problems became easier to discuss as there were no longer any barriers to the understanding, or the description, of the problem in question as both parties to the conversation know what every term means and what each colour-coded figure

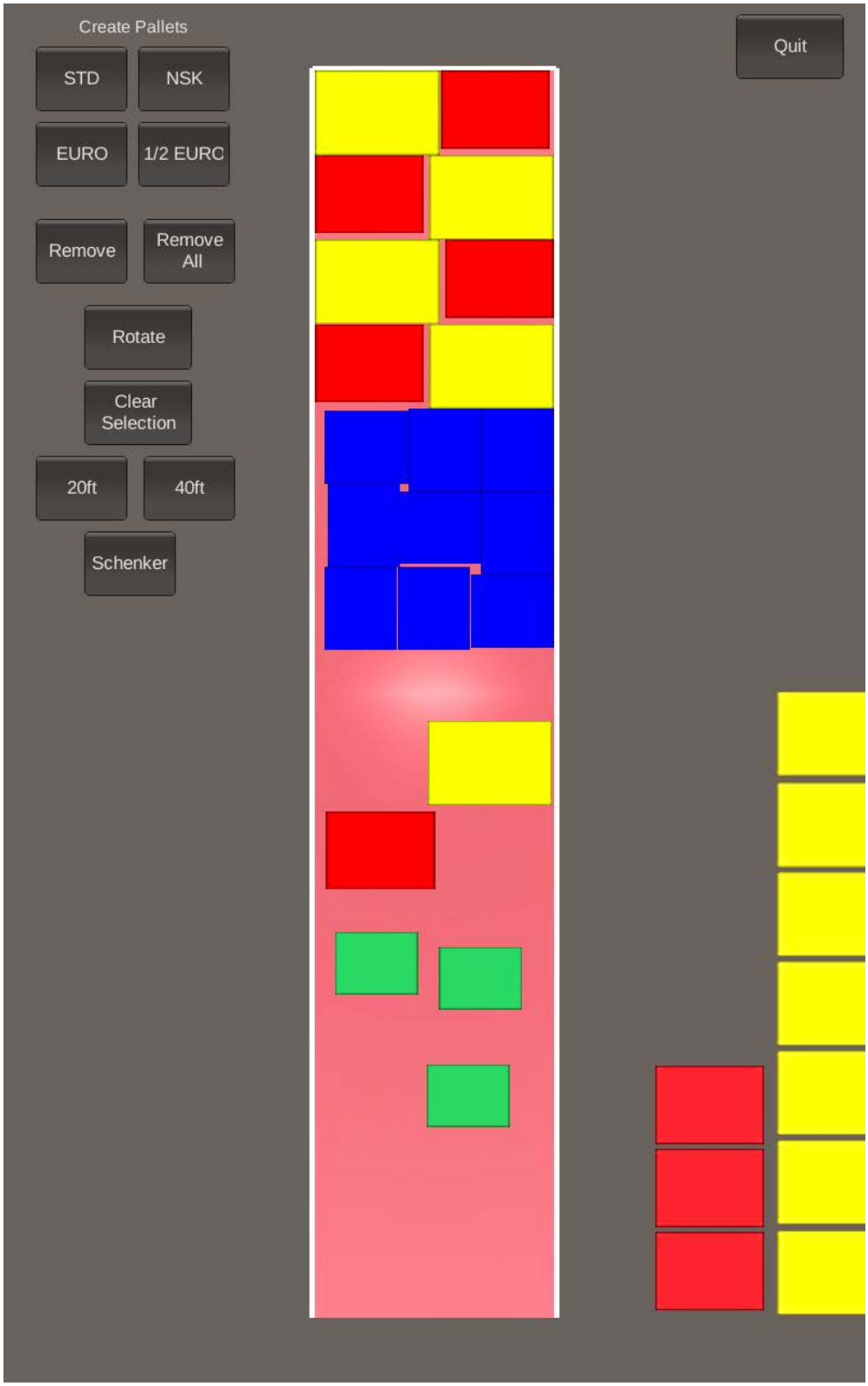


Figure A.3: Interactive simulation interface for the loading system

represents. Our visual representation convention has now been internalised so much that it is used in the day-to-day discussion of general loading activities, not necessarily related to the loading system, in the warehouse (see Figure A.4).

In our initial gamified representation of the loading system output, users were presented with loading layouts as seen in Figure A.2. The users often commented on how the system output was feasible but not how they would have loaded it themselves. This sentiment was expressed several times by different loaders. We observed that in the majority of the times this comment was made, the changes the users would have made to the generated layout were minor, and if these minor changes could be made, the user's satisfaction would increase. Using this feedback, we further gamified the system to produce an interactive simulation interface. Using the interactive interface, loading plans were no longer set in stone, and loaders were free to modify the results of loading operations to better suit their preferred loading style, while still ensuring that the resulting new layout is feasible. This feature alone caused a significant increase in user engagement with the system.

A.5.1 Gamified system use cases

As a result of this increased engagement, additional use cases of the gamified system were identified to include some functionality that was not an intended part of the original system design.

A.5.1.1 Loading Feasibility Checker

The system can be used to check if a load can fit completely into the simulated container. As the simulation is built to scale, if the load fits in the simulation, it will most likely fit in the real world. The users used this functionality often to check the feasibility of planned loads in the simulation before proceeding with actual physical loading in the real world. This helped to catch any potential issues that could occur before actual physical loading is performed, saving time that would otherwise have been spent trying to rectify the issue. In turn, this saved costs possibly incurred through loading damage. We remind the reader that the real-world loading operations involve using forklift trucks to move around

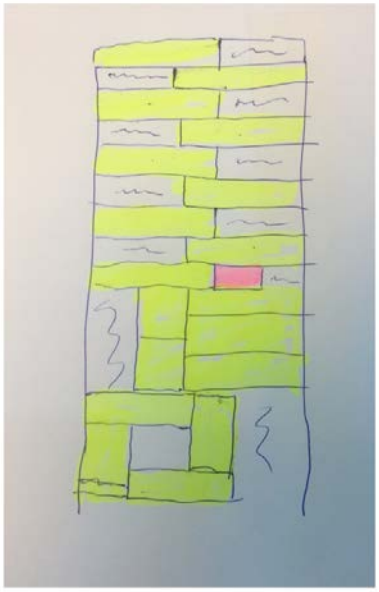


Figure A.4: An operative uses our colour scheme when sketching a layout

heavy goods; it is easier, faster and safer to plan out such activities first in the simulation and then loading, rather than directly proceeding with physical loading and trying to rectify any issues that develop as they manifest. This particular complementary behaviour of the system has proven to be very useful to the operatives.

A.5.1.2 Knowledge Discovery Tool

The system has sometimes generated and presented loading layout patterns that the loaders have never experienced or implemented before. A common comment received from the users regarding this behaviour is "I would never have thought to do it that way". Some of these interesting loading layouts allow the loaders to pack more goods onto the container than they previously thought possible; others introduce entirely new ways of packing loads efficiently. The loaders have adopted these new patterns and started to apply them practically to their loading operations in the real-world (see Figure A.5 and Figure A.6).

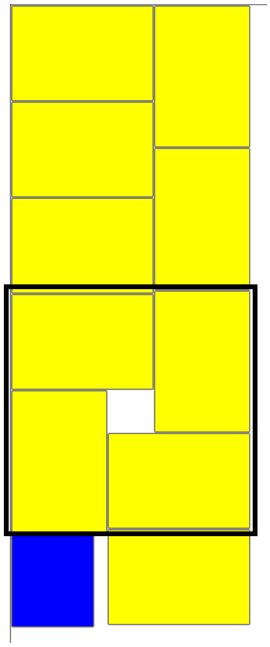


Figure A.5: Loading system representation of an interlocking arrangement of boxes

A.5.1.3 Training Aid

The system can be used as a training aid for teaching new or inexperienced loaders about loading and how to perform loading activities. We observed that this category of users found it easier to follow explanations of loading activities that were communicated to them visually. Learning is made easier if the user can see, instead of imagine, what exactly a load should look like, and what steps to take to complete a loading activity. The simulation interface provides such a visual communication interface that can help make loading activities more tangible. It also provides immediate feedback, scenario testing, and the opportunity to make mistakes with no real-world impact, which can be invaluable to a learner.

In the long run, we observed that continuous interaction with the gamified system: checking the feasibility of loading plans, explaining loading concepts, experimenting with alternative plans for the same load, gradually brought about trust in the system. The users had over time come to rely on the output of the system and

Figure A.6: A loaders real-world representation of a loading plan using the same interlocking arrangement

on its capability to help check the feasibility of their own work. The system was now seen in a different perspective as an assistive technology brought in to complement their own effort and to help them perform their job more effectively. We acknowledge that making the system easily accessible and interactive played an important role in engaging users; creating a simulation interface to present the cryptic output of the complicated algorithms as easily accessible interactive layouts helped to significantly increase system adoption.

Overall, the application of gamification principles and the manner of our approach has had a very positive effect on the use of the underlying loading system to which we applied the principles. The gamified system has increased, and continues to retain, user engagement and has provided a fun and engaging environment for performing serious loading tasks and activities.

A.6 Conclusion

The majority of the studies on gamification tend to generally indicate a positive effect on the system that is gamified; this is however highly dependent on the context in which the gamification is applied and on the users of the gamified system [Groh, 2012; Hamari et al., 2014]. We have taken specific gamification principles and applied them to the industrial context of a warehouse environment, with warehouse operatives as the users of the system. Our preliminary investigations revealed that the introduction of the gamification principles had a very positive effect on the adoption of the underlying container loading system. Prior to the introduction of gamification principles, the adoption of the system was poor with warehouse operatives being wary of a system they saw as a potential replacement for themselves and their work. Gamifying the system helped changed this perception over time by presenting the system in a less threatening manner as an engaging environment where serious work and learning intersect with fun. This gamified user interface on top of the system helped break down perceived barriers that had previously been set up, and helped the loaders see the system as it was intended, as an assistive system to help complement their loading operations and thus increase their overall performance.

Appendix B: Verified Hybrid Algorithm solutions

The data presented in the following table (Table B.1) represents solutions obtained from the hybrid algorithm presented in Chapter 4. Each row in the table represents a solution found by the algorithm i.e. the total number of pallet types selected that the algorithm found to fit in the container. There are 50 of such solutions presented in the table. Each of the solutions was found to obtain a weight utilisation of 100%. These results were validated and confirmed by experienced warehouse operatives.

Table B.1: Summary of 50 solutions confirmed to have 100% utilisation

Solutions	**Number of Pallet Types**				**Total Pallets**
	EURO2	**STD**	**NSK**	**EURO**	
1	1	48	22	3	74
2	1	42	11	11	65
3	0	49	14	5	68
4	0	36	20	9	65
5	0	42	22	3	67
6	1	39	20	3	63
7	2	36	25	3	66
8	1	25	32	2	60
9	1	28	28	3	60
10	1	23	31	3	58
11	0	23	33	6	62

12	1	22	24	11	58
13	0	28	31	6	65
14	0	30	25	12	67
15	2	40	24	6	72
16	2	41	26	2	71
17	0	48	18	2	68
18	0	39	16	13	68
19	0	36	20	9	65
20	3	27	22	7	59
21	3	25	26	6	60
22	0	24	28	7	59
23	2	16	17	16	51
24	3	19	27	7	56
25	4	21	29	6	60
26	3	16	32	6	57
27	3	24	28	5	60
28	5	19	27	0	51
29	5	21	32	3	61
30	5	21	35	1	62
31	3	49	16	3	71
32	4	32	14	13	63
33	5	34	20	5	64
34	3	40	22	2	67
35	3	19	22	14	58
36	2	32	21	6	61
37	3	40	15	6	64
38	0	32	20	6	58
39	2	21	18	12	53
40	1	22	25	7	55
41	3	35	22	2	62
42	1	20	13	19	53
43	1	27	26	3	57

44	3	22	22	10	57
45	4	23	28	3	58
46	5	20	24	3	52
47	3	25	27	3	58
48	2	25	28	3	58
49	1	42	8	13	64
50	5	20	33	1	59

Appendix C: Hybrid Algorithm Problem Sets

The following tables present the data for the 15 problem sets used to evaluate the hybrid algorithm in the experiments presented in Chapter 4.

Table C.1: Problem Set #1

Pallet number	Weight	Length	Breadth	Height
JOB0001/00001	381	120	81	78
JOB0001/00002	487	105	75	61
JOB0001/00003	597	105	75	73
JOB0001/00004	380	105	75	56
JOB0001/00005	497	105	75	69
JOB0001/00006	279	80	70	81
JOB0001/00007	306	80	70	72
JOB0001/00008	655	120	81	69
JOB0001/00009	561	105	75	73
JOB0001/00010	810	105	75	72
JOB0001/00011	718	105	75	73
JOB0001/00012	325	105	75	56
JOB0001/00013	479	105	75	61
JOB0001/00014	228	80	70	72
JOB0001/00015	433	80	70	79
JOB0002/00001	523	120	81	62

Continued on next page

Table C.1 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0002/00002	699	120	81	74
JOB0002/00003	608	120	81	76
JOB0002/00004	551	120	81	76
JOB0002/00005	648	120	81	88
JOB0002/00006	662	120	81	77
JOB0002/00007	729	120	81	76
JOB0002/00008	503	120	81	76
JOB0002/00009	513	120	81	76
JOB0002/00010	852	120	81	78
JOB0002/00011	353	80	70	71
JOB0002/00012	815	120	81	94
JOB0003/00001	176	80	70	45
JOB0003/00002	362	80	70	74
JOB0003/00003	358	80	70	74
JOB0003/00004	296	80	70	74
JOB0003/00005	614	80	70	74
JOB0003/00006	467	120	81	60
JOB0003/00007	227	80	70	58
JOB0004/00001	52	80	60	30
JOB0005/00001	577	105	75	87
JOB0005/00002	577	105	75	87
JOB0005/00003	46	80	60	36
JOB0006/00001	66	80	60	28
JOB0007/00001	551	120	81	72
JOB0007/00002	564	120	81	73
JOB0007/00003	508	80	70	75
JOB0007/00004	459	80	70	75
JOB0007/00005	282	80	70	75
JOB0007/00006	351	80	70	75
JOB0007/00007	378	80	70	75

Continued on next page

Table C.1 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0007/00008	393	80	70	84
JOB0007/00009	412	80	70	96
JOB0007/00010	447	80	70	75
JOB0007/00011	498	80	70	75
JOB0007/00012	349	80	70	86
JOB0007/00013	393	80	70	81
JOB0007/00014	329	80	70	80
JOB0007/00015	563	120	81	70
JOB0007/00016	618	105	75	70
JOB0008/00001	400	105	75	63
JOB0008/00002	204	80	70	49
JOB0008/00003	163	80	70	66
JOB0009/00001	360	80	70	77
JOB0009/00002	241	80	70	75
JOB0009/00003	363	80	70	73
JOB0009/00004	372	80	70	73
JOB0009/00005	337	80	70	73
JOB0009/00006	549	105	75	71
JOB0009/00007	406	80	70	72
JOB0009/00008	331	80	70	72
JOB0009/00009	543	105	75	72
JOB0009/00010	342	80	70	76
JOB0009/00011	464	80	70	73
JOB0009/00012	303	80	70	76
JOB0009/00013	757	105	75	83
JOB0009/00014	387	80	70	72
JOB0009/00015	340	80	70	72
JOB0009/00016	298	80	70	72
JOB0009/00017	735	105	75	84
JOB0010/00001	294	80	70	74

Continued on next page

Table C.1 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0010/00002	592	105	75	71
JOB0010/00003	391	80	70	92
JOB0010/00004	279	80	70	72
JOB0010/00005	401	120	81	76
JOB0010/00006	495	105	75	69
JOB0011/00001	308	80	70	81
JOB0011/00002	296	80	70	81
JOB0011/00003	418	80	70	73
JOB0011/00004	522	105	75	72
JOB0011/00005	693	105	75	86
JOB0011/00006	336	80	70	79
JOB0011/00007	633	105	75	75
JOB0011/00008	590	105	75	79
JOB0011/00009	600	105	75	71
JOB0011/00010	600	105	75	71
JOB0011/00011	599	105	75	71
JOB0011/00012	345	80	70	63
JOB0011/00013	503	105	75	61
JOB0011/00014	600	105	75	71
JOB0011/00015	534	105	75	72
JOB0011/00016	740	105	75	71
JOB0011/00017	435	80	70	73
JOB0012/00001	292	80	70	58
JOB0012/00002	700	120	81	60
JOB0012/00003	676	120	81	76
JOB0012/00004	816	120	81	76
JOB0012/00005	503	120	81	60
JOB0012/00006	601	80	70	92
JOB0012/00007	700	120	81	76
JOB0012/00008	660	120	81	76

Continued on next page

Table C.1 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0012/00009	661	120	81	92
JOB0012/00010	292	80	70	73
JOB0012/00011	407	80	70	69
JOB0012/00012	619	120	81	76
JOB0012/00013	459	120	81	61
JOB0013/00001	268	80	70	64
JOB0013/00002	515	105	75	72
JOB0013/00003	552	105	75	73
JOB0013/00004	555	105	75	72
JOB0013/00005	346	105	75	47
JOB0013/00006	264	105	75	69
JOB0013/00007	416	105	75	58
JOB0013/00008	403	105	75	58
JOB0013/00009	261	80	70	76
JOB0013/00010	371	80	70	95
JOB0013/00011	156	80	70	60
JOB0013/00012	549	120	81	76
JOB0014/00001	776	105	75	81
JOB0014/00002	332	105	75	56
JOB0014/00003	418	80	70	99
JOB0014/00004	553	105	75	65
JOB0014/00005	352	80	70	73
JOB0014/00006	450	105	75	56
JOB0014/00007	261	80	70	64
JOB0014/00008	500	105	75	56
JOB0014/00009	924	105	75	85
JOB0014/00010	440	105	75	56
JOB0014/00011	653	105	75	70
JOB0014/00012	322	80	70	74
JOB0014/00013	393	80	70	91

Continued on next page

Table C.1 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0014/00014	310	80	70	66
JOB0015/00001	316	80	70	74
JOB0015/00002	331	80	70	74
JOB0015/00003	308	80	70	74
JOB0015/00004	281	80	70	71
JOB0015/00005	296	80	70	73
JOB0015/00006	441	80	70	89
JOB0015/00007	347	80	70	74
JOB0015/00008	317	80	70	74
JOB0015/00009	319	80	70	74
JOB0015/00010	154	80	70	51
JOB0015/00011	400	80	70	74
JOB0015/00012	258	80	70	59
JOB0015/00013	366	80	70	74
JOB0015/00014	184	80	70	67
JOB0015/00015	345	80	70	74
JOB0015/00016	343	80	70	74
JOB0015/00017	349	80	70	74
JOB0015/00018	368	80	70	73
JOB0015/00019	276	80	70	73
JOB0015/00020	276	80	70	73
JOB0015/00021	74	80	70	39
JOB0016/00001	309	80	70	79
JOB0016/00002	282	80	70	66
JOB0016/00003	306	80	70	78
JOB0016/00004	499	105	75	59
JOB0016/00005	617	105	75	71
JOB0016/00006	418	105	75	59
JOB0016/00007	238	80	70	79
JOB0016/00008	420	105	75	59

Continued on next page

Table C.1 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0016/00009	619	105	75	71
JOB0016/00010	628	105	75	74
JOB0016/00011	617	105	75	71
JOB0016/00012	479	105	75	59
JOB0016/00013	605	105	75	71
JOB0016/00014	279	105	75	71
JOB0016/00015	196	80	70	49
JOB0016/00016	783	105	75	71
JOB0016/00017	645	105	75	73
JOB0017/00001	737	105	75	82
JOB0017/00002	736	105	75	82
JOB0017/00003	737	105	75	82
JOB0017/00004	409	80	70	84
JOB0017/00005	355	80	70	83
JOB0017/00006	402	80	70	83
JOB0017/00007	640	105	75	81
JOB0017/00008	656	105	75	81
JOB0017/00009	316	105	75	72
JOB0017/00010	733	105	75	82
JOB0017/00011	732	105	75	82
JOB0017/00012	212	80	70	77
JOB0017/00013	231	80	70	73
JOB0017/00014	584	105	75	83
JOB0017/00015	416	105	75	72
JOB0018/00001	616	105	75	72
JOB0018/00002	600	105	75	72
JOB0018/00003	125	80	60	46
JOB0018/00004	361	80	70	84
JOB0018/00005	472	80	70	92
JOB0018/00006	352	80	70	82

Continued on next page

Table C.1 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0018/00007	583	105	75	71
JOB0018/00008	680	105	75	73
JOB0018/00009	370	80	70	83
JOB0019/00001	335	80	70	83
JOB0019/00002	444	80	70	83
JOB0019/00003	307	80	70	73
JOB0019/00004	825	120	81	82
JOB0019/00005	315	80	70	81
JOB0019/00006	286	80	70	76
JOB0019/00007	250	80	70	72
JOB0019/00008	389	120	81	83
JOB0019/00009	683	105	75	82
JOB0019/00010	484	105	75	63
JOB0019/00011	230	105	75	41
JOB0019/00012	441	105	75	71
JOB0019/00013	696	105	75	82
JOB0019/00014	75	80	70	43
JOB0020/00001	193	80	70	60
JOB0020/00002	363	105	75	61
JOB0020/00003	394	80	70	82
JOB0020/00004	254	80	70	62
JOB0020/00005	484	105	75	62
JOB0020/00006	594	105	75	77
JOB0020/00007	212	80	70	60
JOB0020/00008	189	80	70	65
JOB0020/00009	446	105	75	56
JOB0020/00010	539	105	75	65
JOB0020/00011	613	105	75	75
JOB0020/00012	608	105	75	72
JOB0020/00013	598	105	75	72

Continued on next page

Table C.1 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0020/00014	208	80	60	66
JOB0020/00015	555	105	75	70
JOB0020/00016	559	105	75	72
JOB0021/00001	599	120	81	74
JOB0021/00002	709	105	75	72
JOB0021/00003	708	105	75	72
JOB0021/00004	341	80	70	89
JOB0021/00005	620	105	75	72
JOB0021/00006	641	105	75	72
JOB0021/00007	563	120	81	67
JOB0021/00008	663	105	75	71
JOB0021/00009	639	105	75	72
JOB0021/00010	522	105	75	71
JOB0021/00011	329	80	70	73
JOB0021/00012	227	80	70	76
JOB0021/00013	291	80	70	76
JOB0021/00014	291	80	70	51
JOB0021/00015	348	80	70	78
JOB0021/00016	506	80	70	76
JOB0021/00017	586	105	75	71
JOB0021/00018	328	80	70	82
JOB0021/00019	518	105	75	71
JOB0021/00020	494	80	70	76
JOB0021/00021	353	80	70	73
JOB0021/00022	234	80	70	76
JOB0021/00023	260	80	70	73
JOB0021/00024	234	80	70	73
JOB0021/00025	331	80	70	73
JOB0021/00026	378	80	70	76
JOB0021/00027	354	80	70	87

Continued on next page

Table C.1 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0022/00001	279	80	70	85
JOB0022/00002	290	80	70	85
JOB0022/00003	325	80	70	84
JOB0022/00004	237	80	70	74
JOB0022/00005	355	80	70	88
JOB0022/00006	350	80	70	96
JOB0022/00007	348	80	70	78
JOB0022/00008	396	80	70	84
JOB0022/00009	331	80	70	79
JOB0022/00010	358	105	75	67
JOB0022/00011	317	80	70	74
JOB0022/00012	398	80	70	85
JOB0022/00013	394	80	70	84
JOB0022/00014	398	105	75	61
JOB0022/00015	399	80	70	84
JOB0022/00016	362	120	81	78
JOB0022/00017	528	105	75	91
JOB0023/00001	679	120	81	90
JOB0023/00002	783	105	75	86
JOB0023/00003	188	80	70	62
JOB0023/00004	742	105	75	71
JOB0023/00005	268	80	70	58
JOB0023/00006	334	80	70	79
JOB0023/00007	656	105	75	85
JOB0023/00008	308	80	70	83
JOB0023/00009	725	105	75	84
JOB0023/00010	726	105	75	84
JOB0023/00011	399	80	70	88
JOB0023/00012	348	80	70	72
JOB0023/00013	348	80	70	72

Continued on next page

Table C.1 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0023/00014	176	80	70	44
JOB0024/00001	435	80	70	92
JOB0024/00002	614	105	75	76
JOB0024/00003	263	80	70	57
JOB0024/00004	317	80	70	78
JOB0024/00005	642	105	75	77
JOB0024/00006	306	80	70	79
JOB0024/00007	502	80	70	92
JOB0025/00001	324	80	70	79
JOB0025/00002	443	80	70	90
JOB0025/00003	433	80	70	82
JOB0025/00004	323	80	70	59
JOB0025/00005	795	120	81	75
JOB0025/00006	573	105	75	82
JOB0025/00007	666	120	81	79
JOB0025/00008	602	105	75	76
JOB0025/00009	620	105	75	68
JOB0025/00010	586	105	75	71
JOB0026/00001	477	80	70	87
JOB0026/00002	341	105	75	47
JOB0026/00003	601	105	75	70
JOB0026/00004	578	105	75	74
JOB0026/00005	442	80	70	94
JOB0026/00006	848	105	75	89
JOB0026/00007	268	80	70	71
JOB0026/00008	336	80	70	76
JOB0026/00009	232	80	70	72
JOB0027/00001	670	105	75	75
JOB0027/00002	281	105	75	66
JOB0027/00003	368	105	75	83

Continued on next page

Table C.1 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0027/00004	534	105	75	71
JOB0027/00005	455	105	75	71
JOB0027/00006	371	80	70	78
JOB0027/00007	280	105	75	66
JOB0027/00008	535	105	75	71
JOB0027/00009	596	105	75	71
JOB0027/00010	595	105	75	71
JOB0027/00011	534	105	75	71
JOB0027/00012	504	105	75	76
JOB0027/00013	536	105	75	71
JOB0027/00014	386	105	75	87
JOB0027/00015	364	80	70	79
JOB0027/00016	201	80	70	46
JOB0028/00001	281	80	70	92
JOB0028/00002	400	80	70	67
JOB0028/00003	778	105	75	71
JOB0028/00004	209	80	70	62
JOB0028/00005	359	80	70	83
JOB0028/00006	339	80	70	84
JOB0028/00007	460	80	70	94
JOB0028/00008	403	105	75	87
JOB0028/00009	311	105	75	71
JOB0029/00001	169	80	70	59
JOB0029/00002	306	80	70	76
JOB0029/00003	875	120	81	76
JOB0030/00001	359	80	70	75
JOB0030/00002	486	80	70	88
JOB0030/00003	630	105	75	80
JOB0030/00004	516	105	75	70
JOB0030/00005	651	105	75	74

Continued on next page

Table C.1 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0030/00006	464	105	75	74
JOB0030/00007	490	105	75	60
JOB0030/00008	379	105	75	57
JOB0030/00009	291	80	70	73
JOB0031/00001	237	80	70	61
JOB0031/00002	352	80	70	72
JOB0031/00003	443	105	75	64
JOB0031/00004	702	105	75	83
JOB0031/00005	476	80	70	96
JOB0031/00006	335	80	70	73
JOB0031/00007	581	105	75	79
JOB0031/00008	391	105	75	65
JOB0031/00009	607	105	75	64
JOB0032/00001	238	80	70	77
JOB0032/00002	412	105	75	69
JOB0032/00003	613	105	75	79
JOB0032/00004	217	80	70	64
JOB0032/00005	198	80	70	59
JOB0032/00006	389	80	70	76
JOB0032/00007	309	80	70	79
JOB0032/00008	358	80	70	66
JOB0032/00009	341	80	70	75
JOB0033/00001	14	80	60	31
JOB0034/00001	36	80	60	29
JOB0035/00001	220	120	81	38
JOB0036/00001	370	80	70	59
JOB0036/00002	880	105	75	92
JOB0036/00003	536	105	75	71
JOB0036/00004	580	120	81	72
JOB0036/00005	812	105	75	70

Continued on next page

Table C.1 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0036/00006	304	80	70	76
JOB0036/00007	800	105	75	70
JOB0036/00008	798	105	75	70
JOB0036/00009	348	80	70	64
JOB0036/00010	288	80	70	63
JOB0037/00001	325	80	70	77
JOB0037/00002	458	80	70	87
JOB0037/00003	937	105	75	85
JOB0037/00004	340	80	70	88
JOB0037/00005	280	105	75	71
JOB0037/00006	229	105	75	51
JOB0037/00007	983	105	75	85
JOB0037/00008	513	105	75	57
JOB0037/00009	709	105	75	70

Table C.2: Problem Set #2

Pallet number	Weight	Length	Breadth	Height
JOB0001/00001	354	80	70	55
JOB0001/00002	589	120	81	63
JOB0001/00003	589	120	81	63
JOB0001/00004	586	120	81	63
JOB0001/00005	589	120	81	63
JOB0001/00006	590	120	81	63
JOB0001/00007	591	120	81	63
JOB0001/00008	241	80	70	61
JOB0001/00009	312	80	70	75
JOB0001/00010	410	80	70	75
JOB0001/00011	306	80	70	75
JOB0001/00012	898	120	81	75
JOB0001/00013	198	80	70	58
JOB0001/00014	201	80	70	59
JOB0001/00015	923	120	81	75
JOB0002/00001	351	120	81	64
JOB0002/00002	378	80	70	80
JOB0002/00003	398	80	70	75
JOB0002/00004	459	80	70	75
JOB0002/00005	468	80	70	75
JOB0003/00001	371	105	75	68
JOB0003/00002	643	105	75	72
JOB0003/00003	322	80	70	87
JOB0003/00004	327	105	75	75
JOB0003/00005	600	105	75	72
JOB0003/00006	328	80	70	84
JOB0003/00007	212	80	70	77
JOB0003/00008	269	105	75	74

Continued on next page

Table C.2 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0003/00009	310	80	70	87
JOB0003/00010	330	105	75	75
JOB0003/00011	369	80	70	78
JOB0003/00012	578	105	75	73
JOB0003/00013	566	105	75	72
JOB0003/00014	225	80	70	56
JOB0003/00015	331	80	70	73
JOB0003/00016	172	80	70	72
JOB0003/00017	179	80	70	69
JOB0004/00001	212	80	70	62
JOB0004/00002	304	80	70	73
JOB0004/00003	282	80	70	76
JOB0004/00004	310	80	70	75
JOB0004/00005	306	80	70	88
JOB0004/00006	136	80	70	60
JOB0004/00007	522	105	75	69
JOB0004/00008	303	80	70	74
JOB0005/00001	503	105	75	72
JOB0006/00001	179	80	70	79
JOB0006/00002	452	80	70	87
JOB0007/00001	384	105	75	57
JOB0007/00002	563	105	75	67
JOB0008/00001	558	120	81	61
JOB0008/00002	341	80	70	74
JOB0008/00003	296	80	70	74
JOB0009/00001	63	80	60	35
JOB0010/00001	23	80	60	50
JOB0011/00001	128	80	70	50
JOB0011/00002	248	80	70	64
JOB0011/00003	279	80	70	81

Continued on next page

Table C.2 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0011/00004	276	80	70	79
JOB0011/00005	253	80	70	64
JOB0011/00006	504	105	75	72
JOB0011/00007	392	120	81	50
JOB0011/00008	560	120	81	83
JOB0012/00001	173	80	70	46
JOB0012/00002	328	80	70	95
JOB0012/00003	374	80	70	85
JOB0012/00004	455	80	70	98
JOB0012/00005	249	80	70	82
JOB0012/00006	364	80	70	89
JOB0012/00007	174	105	75	71
JOB0012/00008	223	105	75	71
JOB0012/00009	194	120	81	64
JOB0012/00010	520	105	75	72
JOB0012/00011	366	80	70	95
JOB0013/00001	312	80	70	77
JOB0013/00002	524	120	81	70
JOB0013/00003	582	105	75	64
JOB0013/00004	766	105	75	79
JOB0013/00005	756	105	75	79
JOB0013/00006	590	120	81	77
JOB0013/00007	677	105	75	71
JOB0013/00008	742	105	75	71
JOB0013/00009	164	80	70	64
JOB0013/00010	228	105	75	64
JOB0013/00011	300	105	75	75
JOB0013/00012	380	105	75	84
JOB0013/00013	268	105	75	71
JOB0013/00014	268	105	75	71

Continued on next page

Table C.2 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0013/00015	545	120	81	88
JOB0013/00016	698	105	75	83
JOB0014/00001	242	80	70	60
JOB0014/00002	662	120	81	79
JOB0014/00003	731	120	81	82
JOB0014/00004	396	80	70	93
JOB0014/00005	397	80	70	76
JOB0014/00006	733	105	75	70
JOB0014/00007	506	120	81	80
JOB0014/00008	273	80	70	87
JOB0015/00001	597	120	81	89
JOB0015/00002	386	80	70	92
JOB0015/00003	858	120	81	90
JOB0015/00004	248	80	70	52
JOB0015/00005	369	80	70	73
JOB0015/00006	349	80	70	64
JOB0015/00007	332	80	70	73
JOB0015/00008	333	80	70	73
JOB0015/00009	562	105	75	70
JOB0015/00010	600	105	75	70
JOB0015/00011	661	105	75	70
JOB0015/00012	505	105	75	70
JOB0015/00013	680	105	75	71
JOB0015/00014	666	105	75	70
JOB0016/00001	94	80	70	29
JOB0017/00001	20	80	60	27
JOB0018/00001	77	80	70	30
JOB0019/00001	443	105	75	64
JOB0020/00001	166	80	70	78
JOB0020/00002	388	80	70	84

Continued on next page

Table C.2 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0020/00003	417	80	70	92
JOB0020/00004	464	80	70	96
JOB0020/00005	299	80	70	63
JOB0020/00006	543	105	75	75
JOB0020/00007	350	80	70	90
JOB0020/00008	296	80	70	89
JOB0021/00001	294	80	70	82
JOB0021/00002	450	105	75	80
JOB0021/00003	169	80	70	63
JOB0021/00004	431	120	81	61
JOB0021/00005	580	105	75	70
JOB0021/00006	609	105	75	74
JOB0021/00007	241	80	70	65
JOB0021/00008	601	105	75	74
JOB0021/00009	428	80	70	82
JOB0021/00010	222	80	70	65
JOB0021/00011	327	80	70	74
JOB0021/00012	222	80	70	75
JOB0021/00013	760	105	75	89
JOB0021/00014	549	105	75	72
JOB0021/00015	474	105	75	72
JOB0022/00001	298	80	70	76
JOB0022/00002	342	105	75	71
JOB0022/00003	294	80	70	76
JOB0022/00004	296	80	70	74
JOB0022/00005	494	80	70	83
JOB0022/00006	392	80	70	75
JOB0022/00007	407	80	70	92
JOB0022/00008	394	80	70	77
JOB0022/00009	477	105	75	71

Continued on next page

Table C.2 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0022/00010	346	80	70	73
JOB0022/00011	186	80	70	75
JOB0022/00012	354	80	70	73
JOB0022/00013	411	80	70	74
JOB0022/00014	222	80	70	44
JOB0022/00015	115	80	60	47
JOB0023/00001	365	80	70	75
JOB0023/00002	625	105	75	60
JOB0023/00003	612	105	75	82
JOB0023/00004	404	105	75	57
JOB0023/00005	348	80	70	91
JOB0023/00006	627	105	75	61
JOB0023/00007	459	105	75	86
JOB0023/00008	357	105	75	71
JOB0023/00009	366	120	81	65
JOB0023/00010	244	80	70	65
JOB0023/00011	356	80	70	92
JOB0023/00012	445	80	70	97
JOB0023/00013	503	105	75	65
JOB0024/00001	551	105	75	72
JOB0024/00002	262	80	70	70
JOB0024/00003	605	105	75	71
JOB0024/00004	369	105	75	56
JOB0024/00005	641	105	75	71
JOB0024/00006	451	105	75	60
JOB0024/00007	374	105	75	60
JOB0024/00008	377	80	70	73
JOB0024/00009	407	80	70	73
JOB0024/00010	358	80	70	86
JOB0024/00011	624	120	81	85

Continued on next page

Table C.2 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0024/00012	766	105	75	72
JOB0025/00001	61	80	70	58
JOB0025/00002	500	80	70	83
JOB0025/00003	318	80	70	73
JOB0025/00004	803	105	75	81
JOB0025/00005	357	105	75	62
JOB0025/00006	478	105	75	65
JOB0025/00007	594	105	75	79
JOB0025/00008	421	80	70	80
JOB0025/00009	320	80	70	89
JOB0025/00010	432	80	70	73
JOB0025/00011	204	80	70	56
JOB0025/00012	173	80	70	49
JOB0026/00001	296	80	70	80
JOB0026/00002	341	80	70	72
JOB0026/00003	280	80	70	72
JOB0026/00004	479	80	70	91
JOB0026/00005	577	105	75	72
JOB0026/00006	116	105	75	56
JOB0026/00007	346	80	70	82
JOB0026/00008	500	105	75	72
JOB0026/00009	429	80	70	81
JOB0026/00010	337	80	70	72
JOB0026/00011	347	80	70	72
JOB0026/00012	360	80	70	70
JOB0026/00013	330	80	70	75
JOB0026/00014	431	80	70	73
JOB0027/00001	487	80	70	76
JOB0027/00002	492	80	70	84
JOB0027/00003	383	80	70	74

Continued on next page

Table C.2 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0027/00004	249	80	70	64
JOB0027/00005	357	80	70	79
JOB0027/00006	323	80	70	81
JOB0027/00007	293	80	70	61
JOB0027/00008	265	80	70	74
JOB0027/00009	515	105	75	72
JOB0027/00010	261	105	75	76
JOB0027/00011	629	105	75	72
JOB0027/00012	486	105	75	73
JOB0027/00013	269	80	70	72
JOB0027/00014	167	80	70	76
JOB0027/00015	244	80	70	63
JOB0028/00001	535	105	75	71
JOB0028/00002	252	80	70	64
JOB0028/00003	300	80	70	73
JOB0028/00004	244	80	70	73
JOB0028/00005	400	80	70	72
JOB0028/00006	379	80	70	82
JOB0028/00007	322	80	70	75
JOB0028/00008	370	80	70	61
JOB0028/00009	345	80	70	73
JOB0028/00010	440	80	70	79
JOB0028/00011	440	80	70	81
JOB0028/00012	405	105	75	63
JOB0029/00001	547	105	75	80
JOB0029/00002	273	80	70	65
JOB0029/00003	377	105	75	62
JOB0029/00004	279	80	70	61
JOB0029/00005	278	105	75	63
JOB0029/00006	513	105	75	75

Continued on next page

Table C.2 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0029/00007	382	105	75	74
JOB0029/00008	175	80	70	64
JOB0029/00009	407	80	70	87
JOB0029/00010	332	80	70	82
JOB0030/00001	703	120	81	63
JOB0030/00002	524	120	81	63
JOB0030/00003	703	120	81	63
JOB0030/00004	770	120	81	79
JOB0031/00001	241	105	75	71
JOB0031/00002	230	105	75	51
JOB0031/00003	254	105	75	72
JOB0031/00004	256	105	75	72
JOB0031/00005	167	80	70	81
JOB0031/00006	253	105	75	72
JOB0031/00007	199	80	70	51
JOB0031/00008	131	105	75	36
JOB0031/00009	369	105	75	64
JOB0032/00001	355	80	70	74
JOB0032/00002	176	80	60	53
JOB0032/00003	382	80	70	74
JOB0032/00004	569	120	81	77
JOB0032/00005	282	80	70	60
JOB0032/00006	380	80	70	73
JOB0032/00007	418	80	70	74
JOB0032/00008	320	80	70	74
JOB0032/00009	248	80	70	59
JOB0032/00010	852	120	81	69
JOB0032/00011	530	120	81	60
JOB0032/00012	618	120	81	69
JOB0032/00013	806	120	81	69

Continued on next page

Table C.2 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0032/00014	351	120	81	53
JOB0032/00015	767	120	81	83
JOB0033/00001	428	80	70	90
JOB0033/00002	511	120	81	63
JOB0033/00003	410	120	81	55
JOB0033/00004	511	120	81	63
JOB0033/00005	297	80	70	84
JOB0034/00001	269	80	70	67
JOB0034/00002	396	80	70	74
JOB0034/00003	208	80	70	74
JOB0034/00004	256	80	70	74
JOB0034/00005	171	80	70	69
JOB0034/00006	320	80	70	62
JOB0034/00007	632	105	75	71
JOB0034/00008	311	80	70	78
JOB0034/00009	180	80	70	54
JOB0034/00010	271	80	70	64
JOB0034/00011	235	80	70	66
JOB0034/00012	261	80	70	57
JOB0035/00001	391	80	70	80
JOB0035/00002	429	80	70	73
JOB0035/00003	366	80	70	73
JOB0035/00004	317	80	70	81
JOB0035/00005	410	80	70	98
JOB0035/00006	222	80	70	80
JOB0035/00007	500	80	70	98
JOB0035/00008	234	80	70	66
JOB0035/00009	717	105	75	71
JOB0035/00010	389	80	70	73
JOB0035/00011	322	80	70	91

Continued on next page

Table C.2 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0035/00012	421	80	70	85

Table C.3: Problem Set #3

Pallet number	Weight	Length	Breadth	Height
JOB0001/00001	360	80	70	77
JOB0001/00002	241	80	70	75
JOB0001/00003	363	80	70	73
JOB0001/00004	372	80	70	73
JOB0001/00005	337	80	70	73
JOB0001/00006	549	105	75	71
JOB0001/00007	406	80	70	72
JOB0001/00008	331	80	70	72
JOB0001/00009	543	105	75	72
JOB0001/00010	342	80	70	76
JOB0001/00011	464	80	70	73
JOB0001/00012	303	80	70	76
JOB0001/00013	757	105	75	83
JOB0001/00014	387	80	70	72
JOB0001/00015	340	80	70	72
JOB0001/00016	298	80	70	72
JOB0001/00017	735	105	75	84
JOB0002/00001	294	80	70	74
JOB0002/00002	592	105	75	71
JOB0002/00003	391	80	70	92
JOB0002/00004	279	80	70	72
JOB0002/00005	401	120	81	76
JOB0002/00006	495	105	75	69
JOB0003/00001	308	80	70	81
JOB0003/00002	296	80	70	81
JOB0003/00003	418	80	70	73
JOB0003/00004	522	105	75	72
JOB0003/00005	693	105	75	86

Continued on next page

Table C.3 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0003/00006	336	80	70	79
JOB0003/00007	633	105	75	75
JOB0003/00008	590	105	75	79
JOB0003/00009	600	105	75	71
JOB0003/00010	600	105	75	71
JOB0003/00011	599	105	75	71
JOB0003/00012	345	80	70	63
JOB0003/00013	503	105	75	61
JOB0003/00014	600	105	75	71
JOB0003/00015	534	105	75	72
JOB0003/00016	740	105	75	71
JOB0003/00017	435	80	70	73
JOB0004/00001	292	80	70	58
JOB0004/00002	700	120	81	60
JOB0004/00003	676	120	81	76
JOB0004/00004	816	120	81	76
JOB0004/00005	503	120	81	60
JOB0004/00006	601	80	70	92
JOB0004/00007	700	120	81	76
JOB0004/00008	660	120	81	76
JOB0004/00009	661	120	81	92
JOB0004/00010	292	80	70	73
JOB0004/00011	407	80	70	69
JOB0004/00012	619	120	81	76
JOB0004/00013	459	120	81	61
JOB0005/00001	268	80	70	64
JOB0005/00002	515	105	75	72
JOB0005/00003	552	105	75	73
JOB0005/00004	555	105	75	72
JOB0005/00005	346	105	75	47

Continued on next page

Table C.3 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0005/00006	264	105	75	69
JOB0005/00007	416	105	75	58
JOB0005/00008	403	105	75	58
JOB0005/00009	261	80	70	76
JOB0005/00010	371	80	70	95
JOB0005/00011	156	80	70	60
JOB0005/00012	549	120	81	76
JOB0006/00001	776	105	75	81
JOB0006/00002	332	105	75	56
JOB0006/00003	418	80	70	99
JOB0006/00004	553	105	75	65
JOB0006/00005	352	80	70	73
JOB0006/00006	450	105	75	56
JOB0006/00007	261	80	70	64
JOB0006/00008	500	105	75	56
JOB0006/00009	924	105	75	85
JOB0006/00010	440	105	75	56
JOB0006/00011	653	105	75	70
JOB0006/00012	322	80	70	74
JOB0006/00013	393	80	70	91
JOB0006/00014	310	80	70	66
JOB0007/00001	316	80	70	74
JOB0007/00002	331	80	70	74
JOB0007/00003	308	80	70	74
JOB0007/00004	281	80	70	71
JOB0007/00005	296	80	70	73
JOB0007/00006	441	80	70	89
JOB0007/00007	347	80	70	74
JOB0007/00008	317	80	70	74
JOB0007/00009	319	80	70	74

Continued on next page

Table C.3 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0007/00010	154	80	70	51
JOB0007/00011	400	80	70	74
JOB0007/00012	258	80	70	59
JOB0007/00013	366	80	70	74
JOB0007/00014	184	80	70	67
JOB0007/00015	345	80	70	74
JOB0007/00016	343	80	70	74
JOB0007/00017	349	80	70	74
JOB0007/00018	368	80	70	73
JOB0007/00019	276	80	70	73
JOB0007/00020	276	80	70	73
JOB0007/00021	74	80	70	39
JOB0008/00001	309	80	70	79
JOB0008/00002	282	80	70	66
JOB0008/00003	306	80	70	78
JOB0008/00004	499	105	75	59
JOB0008/00005	617	105	75	71
JOB0008/00006	418	105	75	59
JOB0008/00007	238	80	70	79
JOB0008/00008	420	105	75	59
JOB0008/00009	619	105	75	71
JOB0008/00010	628	105	75	74
JOB0008/00011	617	105	75	71
JOB0008/00012	479	105	75	59
JOB0008/00013	605	105	75	71
JOB0008/00014	279	105	75	71
JOB0008/00015	196	80	70	49
JOB0008/00016	783	105	75	71
JOB0008/00017	645	105	75	73
JOB0008/00001	737	105	75	82

Continued on next page

Table C.3 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0008/00002	736	105	75	82
JOB0008/00003	737	105	75	82
JOB0008/00004	409	80	70	84
JOB0008/00005	355	80	70	83
JOB0008/00006	402	80	70	83
JOB0008/00007	640	105	75	81
JOB0008/00008	656	105	75	81
JOB0008/00009	316	105	75	72
JOB0008/00010	733	105	75	82
JOB0008/00011	732	105	75	82
JOB0008/00012	212	80	70	77
JOB0008/00013	231	80	70	73
JOB0008/00014	584	105	75	83
JOB0008/00015	416	105	75	72
JOB0009/00001	616	105	75	72
JOB0009/00002	600	105	75	72
JOB0009/00003	125	80	60	46
JOB0009/00004	361	80	70	84
JOB0009/00005	472	80	70	92
JOB0009/00006	352	80	70	82
JOB0009/00007	583	105	75	71
JOB0009/00008	680	105	75	73
JOB0009/00009	370	80	70	83
JOB0010/00001	335	80	70	83
JOB0010/00002	444	80	70	83
JOB0010/00003	307	80	70	73
JOB0010/00004	825	120	81	82
JOB0010/00005	315	80	70	81
JOB0010/00006	286	80	70	76
JOB0010/00007	250	80	70	72

Continued on next page

Table C.3 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0010/00008	389	120	81	83
JOB0010/00009	683	105	75	82
JOB0010/00010	484	105	75	63
JOB0010/00011	230	105	75	41
JOB0010/00012	441	105	75	71
JOB0010/00013	696	105	75	82
JOB0010/00014	75	80	70	43
JOB0011/00001	193	80	70	60
JOB0011/00002	363	105	75	61
JOB0011/00003	394	80	70	82
JOB0011/00004	254	80	70	62
JOB0011/00005	484	105	75	62
JOB0011/00006	594	105	75	77
JOB0011/00007	212	80	70	60
JOB0011/00008	189	80	70	65
JOB0011/00009	446	105	75	56
JOB0011/00010	539	105	75	65
JOB0011/00011	613	105	75	75
JOB0011/00012	608	105	75	72
JOB0011/00013	598	105	75	72
JOB0011/00014	208	80	60	66
JOB0011/00015	555	105	75	70
JOB0011/00016	559	105	75	72
JOB0012/00001	599	120	81	74
JOB0012/00002	709	105	75	72
JOB0012/00003	708	105	75	72
JOB0012/00004	341	80	70	89
JOB0012/00005	620	105	75	72
JOB0012/00006	641	105	75	72
JOB0012/00007	563	120	81	67

Continued on next page

Table C.3 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0012/00008	663	105	75	71
JOB0012/00009	639	105	75	72
JOB0012/00010	522	105	75	71
JOB0012/00011	329	80	70	73
JOB0012/00012	227	80	70	76
JOB0012/00013	291	80	70	76
JOB0012/00014	291	80	70	51
JOB0012/00015	348	80	70	78
JOB0012/00016	506	80	70	76
JOB0012/00017	586	105	75	71
JOB0012/00018	328	80	70	82
JOB0012/00019	518	105	75	71
JOB0012/00020	494	80	70	76
JOB0012/00021	353	80	70	73
JOB0012/00022	234	80	70	76
JOB0012/00023	260	80	70	73
JOB0012/00024	234	80	70	73
JOB0012/00025	331	80	70	73
JOB0012/00026	378	80	70	76
JOB0012/00027	354	80	70	87
JOB0013/00001	279	80	70	85
JOB0013/00002	290	80	70	85
JOB0013/00003	325	80	70	84
JOB0013/00004	237	80	70	74
JOB0013/00005	355	80	70	88
JOB0013/00006	350	80	70	96
JOB0013/00007	348	80	70	78
JOB0013/00008	396	80	70	84
JOB0013/00009	331	80	70	79
JOB0013/00010	358	105	75	67

Continued on next page

Table C.3 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0013/00011	317	80	70	74
JOB0013/00012	398	80	70	85
JOB0013/00013	394	80	70	84
JOB0013/00014	398	105	75	61
JOB0013/00015	399	80	70	84
JOB0013/00016	362	120	81	78
JOB0013/00017	528	105	75	91
JOB0014/00001	679	120	81	90
JOB0014/00002	783	105	75	86
JOB0014/00003	188	80	70	62
JOB0014/00004	742	105	75	71
JOB0014/00005	268	80	70	58
JOB0014/00006	334	80	70	79
JOB0014/00007	656	105	75	85
JOB0014/00008	308	80	70	83
JOB0014/00009	725	105	75	84
JOB0014/00010	726	105	75	84
JOB0014/00011	399	80	70	88
JOB0014/00012	348	80	70	72
JOB0014/00013	348	80	70	72
JOB0014/00014	176	80	70	44
JOB0015/00001	435	80	70	92
JOB0015/00002	614	105	75	76
JOB0015/00003	263	80	70	57
JOB0015/00004	317	80	70	78
JOB0015/00005	642	105	75	77
JOB0015/00006	306	80	70	79
JOB0015/00007	502	80	70	92
JOB0016/00001	324	80	70	79
JOB0016/00002	443	80	70	90

Continued on next page

Table C.3 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0016/00003	433	80	70	82
JOB0016/00004	323	80	70	59
JOB0016/00005	795	120	81	75
JOB0016/00006	573	105	75	82
JOB0016/00007	666	120	81	79
JOB0016/00008	602	105	75	76
JOB0016/00009	620	105	75	68
JOB0016/00010	586	105	75	71
JOB0017/00001	477	80	70	87
JOB0017/00002	341	105	75	47
JOB0017/00003	601	105	75	70
JOB0017/00004	578	105	75	74
JOB0017/00005	442	80	70	94
JOB0017/00006	848	105	75	89
JOB0017/00007	268	80	70	71
JOB0017/00008	336	80	70	76
JOB0017/00009	232	80	70	72
JOB0018/00001	670	105	75	75
JOB0018/00002	281	105	75	66
JOB0018/00003	368	105	75	83
JOB0018/00004	534	105	75	71
JOB0018/00005	455	105	75	71
JOB0018/00006	371	80	70	78
JOB0018/00007	280	105	75	66
JOB0018/00008	535	105	75	71
JOB0018/00009	596	105	75	71
JOB0018/00010	595	105	75	71
JOB0018/00011	534	105	75	71
JOB0018/00012	504	105	75	76
JOB0018/00013	536	105	75	71

Continued on next page

Table C.3 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0018/00014	386	105	75	87
JOB0018/00015	364	80	70	79
JOB0018/00016	201	80	70	46
JOB0019/00001	281	80	70	92
JOB0019/00002	400	80	70	67
JOB0019/00003	778	105	75	71
JOB0019/00004	209	80	70	62
JOB0019/00005	359	80	70	83
JOB0019/00006	339	80	70	84
JOB0019/00007	460	80	70	94
JOB0019/00008	403	105	75	87
JOB0019/00009	311	105	75	71
JOB0020/00001	169	80	70	59
JOB0020/00002	306	80	70	76
JOB0020/00003	875	120	81	76
JOB0021/00001	359	80	70	75
JOB0021/00002	486	80	70	88
JOB0021/00003	630	105	75	80
JOB0021/00004	516	105	75	70
JOB0021/00005	651	105	75	74
JOB0021/00006	464	105	75	74
JOB0021/00007	490	105	75	60
JOB0021/00008	379	105	75	57
JOB0021/00009	291	80	70	73
JOB0022/00001	237	80	70	61
JOB0022/00002	352	80	70	72
JOB0022/00003	443	105	75	64
JOB0022/00004	702	105	75	83
JOB0022/00005	476	80	70	96
JOB0022/00006	335	80	70	73

Continued on next page

Table C.3 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0022/00007	581	105	75	79
JOB0022/00008	391	105	75	65
JOB0022/00009	607	105	75	64
JOB0023/00001	238	80	70	77
JOB0023/00002	412	105	75	69
JOB0023/00003	613	105	75	79
JOB0023/00004	217	80	70	64
JOB0023/00005	198	80	70	59
JOB0023/00006	389	80	70	76
JOB0023/00007	309	80	70	79
JOB0023/00008	358	80	70	66
JOB0023/00009	341	80	70	75
JOB0024/00001	14	80	60	31
JOB0025/00001	36	80	60	29
JOB0026/00001	220	120	81	38
JOB0027/00001	370	80	70	59
JOB0027/00002	880	105	75	92
JOB0027/00003	536	105	75	71
JOB0027/00004	580	120	81	72
JOB0027/00005	812	105	75	70
JOB0027/00006	304	80	70	76
JOB0027/00007	800	105	75	70
JOB0027/00008	798	105	75	70
JOB0027/00009	348	80	70	64
JOB0027/00010	288	80	70	63
JOB0028/00001	325	80	70	77
JOB0028/00002	458	80	70	87
JOB0028/00003	937	105	75	85
JOB0028/00004	340	80	70	88
JOB0028/00005	280	105	75	71

Continued on next page

Table C.3 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0028/00006	229	105	75	51
JOB0028/00007	983	105	75	85
JOB0028/00008	513	105	75	57
JOB0028/00009	709	105	75	70

Table C.4: Problem Set #4

Pallet number	Weight	Length	Breadth	Height
JOB0001/00001	268	80	70	64
JOB0001/00002	515	105	75	72
JOB0001/00003	552	105	75	73
JOB0001/00004	555	105	75	72
JOB0001/00005	346	105	75	47
JOB0001/00006	264	105	75	69
JOB0001/00007	416	105	75	58
JOB0001/00008	403	105	75	58
JOB0001/00009	261	80	70	76
JOB0001/00010	371	80	70	95
JOB0001/00011	156	80	70	60
JOB0001/00012	549	120	81	76
JOB0002/00001	776	105	75	81
JOB0002/00002	332	105	75	56
JOB0002/00003	418	80	70	99
JOB0002/00004	553	105	75	65
JOB0002/00005	352	80	70	73
JOB0002/00006	450	105	75	56
JOB0002/00007	261	80	70	64
JOB0002/00008	500	105	75	56
JOB0002/00009	924	105	75	85
JOB0002/00010	440	105	75	56
JOB0002/00011	653	105	75	70
JOB0002/00012	322	80	70	74
JOB0002/00013	393	80	70	91
JOB0002/00014	310	80	70	66
JOB0003/00001	316	80	70	74
JOB0003/00002	331	80	70	74

Continued on next page

Table C.4 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0003/00003	308	80	70	74
JOB0003/00004	281	80	70	71
JOB0003/00005	296	80	70	73
JOB0003/00006	441	80	70	89
JOB0003/00007	347	80	70	74
JOB0003/00008	317	80	70	74
JOB0003/00009	319	80	70	74
JOB0003/00010	154	80	70	51
JOB0003/00011	400	80	70	74
JOB0003/00012	258	80	70	59
JOB0003/00013	366	80	70	74
JOB0003/00014	184	80	70	67
JOB0003/00015	345	80	70	74
JOB0003/00016	343	80	70	74
JOB0003/00017	349	80	70	74
JOB0003/00018	368	80	70	73
JOB0003/00019	276	80	70	73
JOB0003/00020	276	80	70	73
JOB0003/00021	74	80	70	39
JOB0004/00001	309	80	70	79
JOB0004/00002	282	80	70	66
JOB0004/00003	306	80	70	78
JOB0004/00004	499	105	75	59
JOB0004/00005	617	105	75	71
JOB0004/00006	418	105	75	59
JOB0004/00007	238	80	70	79
JOB0004/00008	420	105	75	59
JOB0004/00009	619	105	75	71
JOB0004/00010	628	105	75	74
JOB0004/00011	617	105	75	71

Continued on next page

Table C.4 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0004/00012	479	105	75	59
JOB0004/00013	605	105	75	71
JOB0004/00014	279	105	75	71
JOB0004/00015	196	80	70	49
JOB0004/00016	783	105	75	71
JOB0004/00017	645	105	75	73
JOB0005/00001	737	105	75	82
JOB0005/00002	736	105	75	82
JOB0005/00003	737	105	75	82
JOB0005/00004	409	80	70	84
JOB0005/00005	355	80	70	83
JOB0005/00006	402	80	70	83
JOB0005/00007	640	105	75	81
JOB0005/00008	656	105	75	81
JOB0005/00009	316	105	75	72
JOB0005/00010	733	105	75	82
JOB0005/00011	732	105	75	82
JOB0005/00012	212	80	70	77
JOB0005/00013	231	80	70	73
JOB0005/00014	584	105	75	83
JOB0005/00015	416	105	75	72
JOB0006/00001	616	105	75	72
JOB0006/00002	600	105	75	72
JOB0006/00003	125	80	60	46
JOB0006/00004	361	80	70	84
JOB0006/00005	472	80	70	92
JOB0006/00006	352	80	70	82
JOB0006/00007	583	105	75	71
JOB0006/00008	680	105	75	73
JOB0006/00009	370	80	70	83

Continued on next page

Table C.4 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0007/00001	335	80	70	83
JOB0007/00002	444	80	70	83
JOB0007/00003	307	80	70	73
JOB0007/00004	825	120	81	82
JOB0007/00005	315	80	70	81
JOB0007/00006	286	80	70	76
JOB0007/00007	250	80	70	72
JOB0007/00008	389	120	81	83
JOB0007/00009	683	105	75	82
JOB0007/00010	484	105	75	63
JOB0007/00011	230	105	75	41
JOB0007/00012	441	105	75	71
JOB0007/00013	696	105	75	82
JOB0007/00014	75	80	70	43
JOB0008/00001	193	80	70	60
JOB0008/00002	363	105	75	61
JOB0008/00003	394	80	70	82
JOB0008/00004	254	80	70	62
JOB0008/00005	484	105	75	62
JOB0008/00006	594	105	75	77
JOB0008/00007	212	80	70	60
JOB0008/00008	189	80	70	65
JOB0008/00009	446	105	75	56
JOB0008/00010	539	105	75	65
JOB0008/00011	613	105	75	75
JOB0008/00012	608	105	75	72
JOB0008/00013	598	105	75	72
JOB0008/00014	208	80	60	66
JOB0008/00015	555	105	75	70
JOB0008/00016	559	105	75	72

Continued on next page

Table C.4 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0009/00001	599	120	81	74
JOB0009/00002	709	105	75	72
JOB0009/00003	708	105	75	72
JOB0009/00004	341	80	70	89
JOB0009/00005	620	105	75	72
JOB0009/00006	641	105	75	72
JOB0009/00007	563	120	81	67
JOB0009/00008	663	105	75	71
JOB0009/00009	639	105	75	72
JOB0009/00010	522	105	75	71
JOB0009/00011	329	80	70	73
JOB0009/00012	227	80	70	76
JOB0009/00013	291	80	70	76
JOB0009/00014	291	80	70	51
JOB0009/00015	348	80	70	78
JOB0009/00016	506	80	70	76
JOB0009/00017	586	105	75	71
JOB0009/00018	328	80	70	82
JOB0009/00019	518	105	75	71
JOB0009/00020	494	80	70	76
JOB0009/00021	353	80	70	73
JOB0009/00022	234	80	70	76
JOB0009/00023	260	80	70	73
JOB0009/00024	234	80	70	73
JOB0009/00025	331	80	70	73
JOB0009/00026	378	80	70	76
JOB0009/00027	354	80	70	87
JOB0010/00001	279	80	70	85
JOB0010/00002	290	80	70	85
JOB0010/00003	325	80	70	84

Continued on next page

Table C.4 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0010/00004	237	80	70	74
JOB0010/00005	355	80	70	88
JOB0010/00006	350	80	70	96
JOB0010/00007	348	80	70	78
JOB0010/00008	396	80	70	84
JOB0010/00009	331	80	70	79
JOB0010/00010	358	105	75	67
JOB0010/00011	317	80	70	74
JOB0010/00012	398	80	70	85
JOB0010/00013	394	80	70	84
JOB0010/00014	398	105	75	61
JOB0010/00015	399	80	70	84
JOB0010/00016	362	120	81	78
JOB0010/00017	528	105	75	91
JOB0011/00001	679	120	81	90
JOB0011/00002	783	105	75	86
JOB0011/00003	188	80	70	62
JOB0011/00004	742	105	75	71
JOB0011/00005	268	80	70	58
JOB0011/00006	334	80	70	79
JOB0011/00007	656	105	75	85
JOB0011/00008	308	80	70	83
JOB0011/00009	725	105	75	84
JOB0011/00010	726	105	75	84
JOB0011/00011	399	80	70	88
JOB0011/00012	348	80	70	72
JOB0011/00013	348	80	70	72
JOB0011/00014	176	80	70	44
JOB0012/00001	435	80	70	92
JOB0012/00002	614	105	75	76

Continued on next page

Table C.4 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0012/00003	263	80	70	57
JOB0012/00004	317	80	70	78
JOB0012/00005	642	105	75	77
JOB0012/00006	306	80	70	79
JOB0012/00007	502	80	70	92
JOB0013/00001	324	80	70	79
JOB0013/00002	443	80	70	90
JOB0013/00003	433	80	70	82
JOB0013/00004	323	80	70	59
JOB0013/00005	795	120	81	75
JOB0013/00006	573	105	75	82
JOB0013/00007	666	120	81	79
JOB0013/00008	602	105	75	76
JOB0013/00009	620	105	75	68
JOB0013/00010	586	105	75	71
JOB0014/00001	477	80	70	87
JOB0014/00002	341	105	75	47
JOB0014/00003	601	105	75	70
JOB0014/00004	578	105	75	74
JOB0014/00005	442	80	70	94
JOB0014/00006	848	105	75	89
JOB0014/00007	268	80	70	71
JOB0014/00008	336	80	70	76
JOB0014/00009	232	80	70	72
JOB0015/00001	670	105	75	75
JOB0015/00002	281	105	75	66
JOB0015/00003	368	105	75	83
JOB0015/00004	534	105	75	71
JOB0015/00005	455	105	75	71
JOB0015/00006	371	80	70	78

Continued on next page

Table C.4 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0015/00007	280	105	75	66
JOB0015/00008	535	105	75	71
JOB0015/00009	596	105	75	71
JOB0015/00010	595	105	75	71
JOB0015/00011	534	105	75	71
JOB0015/00012	504	105	75	76
JOB0015/00013	536	105	75	71
JOB0015/00014	386	105	75	87
JOB0015/00015	364	80	70	79
JOB0015/00016	201	80	70	46
JOB0016/00001	281	80	70	92
JOB0016/00002	400	80	70	67
JOB0016/00003	778	105	75	71
JOB0016/00004	209	80	70	62
JOB0016/00005	359	80	70	83
JOB0016/00006	339	80	70	84
JOB0016/00007	460	80	70	94
JOB0016/00008	403	105	75	87
JOB0016/00009	311	105	75	71
JOB0017/00001	169	80	70	59
JOB0017/00002	306	80	70	76
JOB0017/00003	875	120	81	76
JOB0018/00001	359	80	70	75
JOB0018/00002	486	80	70	88
JOB0018/00003	630	105	75	80
JOB0018/00004	516	105	75	70
JOB0018/00005	651	105	75	74
JOB0018/00006	464	105	75	74
JOB0018/00007	490	105	75	60
JOB0018/00008	379	105	75	57

Continued on next page

Table C.4 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0018/00009	291	80	70	73
JOB0019/00001	237	80	70	61
JOB0019/00002	352	80	70	72
JOB0019/00003	443	105	75	64
JOB0019/00004	702	105	75	83
JOB0019/00005	476	80	70	96
JOB0019/00006	335	80	70	73
JOB0019/00007	581	105	75	79
JOB0019/00008	391	105	75	65
JOB0019/00009	607	105	75	64
JOB0020/00001	238	80	70	77
JOB0020/00002	412	105	75	69
JOB0020/00003	613	105	75	79
JOB0020/00004	217	80	70	64
JOB0020/00005	198	80	70	59
JOB0020/00006	389	80	70	76
JOB0020/00007	309	80	70	79
JOB0020/00008	358	80	70	66
JOB0020/00009	341	80	70	75
JOB0021/00001	14	80	60	31
JOB0022/00001	36	80	60	29
JOB0023/00001	220	120	81	38
JOB0024/00001	370	80	70	59
JOB0024/00002	880	105	75	92
JOB0024/00003	536	105	75	71
JOB0024/00004	580	120	81	72
JOB0024/00005	812	105	75	70
JOB0024/00006	304	80	70	76
JOB0024/00007	800	105	75	70
JOB0024/00008	798	105	75	70

Continued on next page

Table C.4 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0024/00009	348	80	70	64
JOB0024/00010	288	80	70	63
JOB0025/00001	325	80	70	77
JOB0025/00002	458	80	70	87
JOB0025/00003	937	105	75	85
JOB0025/00004	340	80	70	88
JOB0025/00005	280	105	75	71
JOB0025/00006	229	105	75	51
JOB0025/00007	983	105	75	85
JOB0025/00008	513	105	75	57
JOB0025/00009	709	105	75	70

Table C.5: Problem Set #5

Pallet number	Weight	Length	Breadth	Height
JOB0001/00001	403	80	70	95
JOB0001/00001	403	80	70	95
JOB0001/00002	243	80	70	57
JOB0001/00003	389	105	75	84
JOB0001/00004	391	105	75	84
JOB0001/00005	293	80	70	73
JOB0001/00006	340	80	70	73
JOB0001/00007	243	105	75	48
JOB0001/00008	308	80	70	74
JOB0001/00009	282	80	70	73
JOB0001/00010	329	80	70	73
JOB0001/00011	312	80	70	76
JOB0001/00012	389	105	75	62
JOB0001/00013	217	80	70	43
JOB0001/00014	248	80	70	62
JOB0001/00015	530	105	75	74
JOB0001/00016	194	80	70	76
JOB0001/00017	199	80	70	76
JOB0001/00018	199	80	70	57
JOB0001/00019	273	80	70	77
JOB0001/00020	322	80	70	73
JOB0001/00021	412	80	70	83
JOB0001/00022	248	80	70	73
JOB0002/00001	460	80	70	65
JOB0002/00002	203	105	75	95
JOB0002/00003	303	105	75	76
JOB0002/00004	555	105	75	57
JOB0002/00005	271	80	70	78

Continued on next page

Table C.5 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0002/00006	323	80	70	77
JOB0002/00007	236	80	70	66
JOB0002/00008	301	80	70	72
JOB0002/00009	323	105	75	74
JOB0002/00010	477	80	70	91
JOB0002/00011	321	105	75	73
JOB0002/00012	188	120	81	67
JOB0002/00013	239	105	75	69
JOB0002/00014	347	105	75	93
JOB0002/00015	347	105	75	93
JOB0002/00016	479	105	75	71
JOB0003/00001	364	105	75	72
JOB0003/00002	635	120	81	74
JOB0003/00003	599	105	75	72
JOB0003/00004	729	120	81	74
JOB0003/00005	464	120	81	69
JOB0003/00006	247	105	75	58
JOB0003/00007	267	80	70	62
JOB0003/00008	408	105	75	61
JOB0003/00009	414	80	70	74
JOB0003/00010	287	80	70	77
JOB0003/00011	131	80	70	52
JOB0003/00012	222	80	70	66
JOB0003/00013	359	80	70	58
JOB0003/00014	422	80	70	87
JOB0003/00015	294	80	70	74
JOB0003/00016	743	105	75	72
JOB0003/00017	395	105	75	69
JOB0003/00018	322	80	70	66
JOB0003/00019	339	80	70	69

Continued on next page

Table C.5 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0004/00001	330	105	75	71
JOB0004/00002	538	120	81	95
JOB0004/00003	704	120	81	90
JOB0004/00004	606	120	81	73
JOB0004/00005	346	105	75	76
JOB0004/00006	700	105	75	86
JOB0004/00007	340	80	70	95
JOB0004/00008	346	80	70	85
JOB0004/00009	577	105	75	71
JOB0004/00010	413	105	75	52
JOB0004/00011	690	105	75	71
JOB0004/00012	288	80	70	79
JOB0005/00001	217	80	70	58
JOB0005/00002	352	80	70	82
JOB0005/00003	380	105	75	46
JOB0005/00004	386	80	70	70
JOB0005/00005	243	80	70	59
JOB0005/00006	353	80	70	80
JOB0005/00007	478	80	70	93
JOB0005/00008	237	80	70	79
JOB0006/00001	236	80	70	57
JOB0006/00002	258	105	75	73
JOB0006/00003	548	120	81	75
JOB0006/00004	186	80	70	39
JOB0006/00005	408	80	70	66
JOB0006/00006	353	80	70	75
JOB0006/00007	313	80	70	75
JOB0006/00008	508	105	75	89
JOB0006/00009	293	80	70	73
JOB0006/00010	583	105	75	72

Continued on next page

Table C.5 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0007/00001	226	80	70	57
JOB0007/00002	684	120	81	77
JOB0007/00003	260	80	70	58
JOB0007/00004	316	80	70	71
JOB0007/00005	534	120	81	75
JOB0008/00001	679	105	75	83
JOB0008/00002	267	105	75	56
JOB0008/00003	310	105	75	74
JOB0008/00004	363	80	70	89
JOB0008/00005	60	80	60	40
JOB0008/00006	807	105	75	74
JOB0008/00007	329	80	70	85
JOB0008/00008	318	105	75	74
JOB0008/00009	375	120	81	39
JOB0008/00010	825	105	75	74
JOB0008/00011	637	105	75	74
JOB0008/00012	348	80	70	82
JOB0008/00013	398	80	70	93
JOB0009/00001	660	120	81	74
JOB0009/00002	416	120	81	60
JOB0009/00003	442	80	70	82
JOB0010/00001	64	80	70	42
JOB0010/00002	380	105	75	72
JOB0010/00003	329	105	75	72
JOB0010/00004	275	80	70	70
JOB0010/00005	675	105	75	71
JOB0010/00006	516	120	81	59
JOB0010/00007	105	80	70	49
JOB0010/00008	748	105	75	86
JOB0010/00009	745	105	75	70

Continued on next page

Table C.5 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0010/00010	303	80	70	76
JOB0010/00011	410	105	75	55
JOB0010/00012	465	105	75	69
JOB0010/00013	128	80	70	67
JOB0010/00014	482	105	75	70
JOB0011/00001	44	80	60	31
JOB0012/00001	14	80	60	30
JOB0013/00001	394	80	70	82
JOB0013/00002	385	105	75	62
JOB0013/00003	803	105	75	81
JOB0013/00004	197	80	70	73
JOB0013/00005	212	80	70	57
JOB0013/00006	347	80	70	93
JOB0013/00007	305	80	70	88
JOB0013/00008	554	105	75	65
JOB0013/00009	410	105	75	89
JOB0013/00010	314	105	75	57
JOB0013/00011	325	80	70	89
JOB0013/00012	311	80	70	77
JOB0014/00001	104	80	70	28
JOB0015/00001	349	80	70	75
JOB0015/00002	361	105	75	47
JOB0015/00003	880	120	81	82
JOB0015/00004	710	105	75	71
JOB0015/00005	461	105	75	62
JOB0015/00006	533	105	75	71
JOB0015/00007	781	120	81	84
JOB0015/00008	435	105	75	86
JOB0015/00009	226	80	70	56
JOB0015/00010	240	80	70	80

Continued on next page

Table C.5 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0015/00011	257	80	70	69
JOB0016/00001	224	80	70	55
JOB0016/00002	580	105	75	66
JOB0016/00003	642	105	75	69
JOB0016/00004	639	105	75	82
JOB0016/00005	392	105	75	56
JOB0016/00006	236	105	75	73
JOB0016/00007	479	80	70	83
JOB0016/00008	167	80	70	85
JOB0016/00009	190	105	75	87
JOB0016/00010	493	105	75	66
JOB0016/00011	232	105	75	50
JOB0016/00012	331	105	75	74
JOB0016/00013	334	105	75	74
JOB0016/00014	463	105	75	72
JOB0016/00015	367	80	70	96
JOB0016/00016	187	80	70	58
JOB0016/00017	149	80	70	53
JOB0017/00001	504	105	75	63
JOB0017/00002	253	80	70	62
JOB0017/00003	273	80	70	92
JOB0017/00004	327	105	75	58
JOB0017/00005	744	105	75	70
JOB0017/00006	310	105	75	75
JOB0017/00007	953	105	75	83
JOB0017/00008	412	105	75	65
JOB0017/00009	659	105	75	80
JOB0017/00010	531	105	75	71
JOB0017/00011	368	105	75	70
JOB0017/00012	701	105	75	84

Continued on next page

Table C.5 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0017/00013	369	105	75	70
JOB0017/00014	321	80	70	82
JOB0017/00015	205	80	70	58
JOB0017/00016	675	120	81	72
JOB0017/00017	280	105	75	57
JOB0018/00001	228	80	70	45
JOB0019/00001	646	120	81	71
JOB0019/00002	674	120	81	67
JOB0019/00003	656	105	75	70
JOB0019/00004	786	105	75	74
JOB0019/00005	444	80	70	76
JOB0019/00006	524	105	75	74
JOB0020/00001	277	105	75	41
JOB0020/00002	326	105	75	54
JOB0020/00003	582	105	75	81
JOB0020/00004	399	105	75	62
JOB0020/00005	558	105	75	73
JOB0020/00006	750	105	75	88
JOB0020/00007	367	80	70	91
JOB0020/00008	240	80	70	72
JOB0020/00009	416	80	70	84
JOB0020/00010	384	80	70	84
JOB0020/00011	408	105	75	66
JOB0020/00012	447	120	81	62
JOB0021/00001	104	80	70	76
JOB0021/00002	104	80	70	76
JOB0021/00003	565	105	75	72
JOB0021/00004	105	80	70	74
JOB0021/00005	206	80	70	69
JOB0021/00006	112	80	70	74

Continued on next page

Table C.5 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0021/00007	303	80	70	76
JOB0021/00008	161	80	70	51
JOB0021/00009	202	80	70	76
JOB0021/00010	105	80	70	76
JOB0021/00011	111	80	70	74
JOB0021/00012	353	80	70	76
JOB0021/00013	232	80	70	76
JOB0021/00014	112	80	70	74
JOB0021/00015	111	80	70	74
JOB0021/00016	155	80	70	91
JOB0021/00017	201	80	70	76
JOB0021/00018	106	80	70	74
JOB0021/00019	112	80	70	74
JOB0021/00020	111	80	70	74
JOB0021/00021	112	80	70	74
JOB0021/00022	112	80	70	74
JOB0021/00023	112	80	70	74
JOB0021/00024	377	105	75	85
JOB0021/00025	339	105	75	54
JOB0021/00026	561	105	75	73
JOB0021/00027	373	105	75	73
JOB0021/00028	553	120	81	75
JOB0021/00029	565	105	75	73
JOB0021/00030	583	105	75	73
JOB0021/00031	549	105	75	73
JOB0021/00032	353	80	70	74
JOB0021/00033	150	105	75	61
JOB0021/00034	156	105	75	61
JOB0021/00035	399	105	75	73
JOB0021/00036	321	105	75	73

Continued on next page

Table C.5 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0021/00037	405	105	75	72
JOB0021/00038	323	105	75	73
JOB0021/00039	392	105	75	72
JOB0021/00040	698	120	81	76
JOB0021/00041	344	105	75	72
JOB0021/00042	366	105	75	71
JOB0021/00043	518	120	81	85
JOB0022/00001	790	120	81	91
JOB0022/00002	914	120	81	91
JOB0022/00003	788	120	81	91
JOB0022/00004	940	120	81	91
JOB0022/00005	341	80	70	74
JOB0022/00006	741	120	81	76

Table C.6: Problem Set #6

Pallet number	Weight	Length	Breadth	Height
JOB0001/00001	262	80	70	65
JOB0001/00002	354	80	70	85
JOB0001/00003	655	120	81	78
JOB0001/00004	520	120	81	62
JOB0001/00005	753	120	81	95
JOB0001/00006	602	105	75	78
JOB0001/00007	396	80	70	88
JOB0001/00008	456	80	70	96
JOB0002/00001	578	105	75	69
JOB0002/00002	599	105	75	69
JOB0002/00003	468	105	75	57
JOB0002/00004	167	80	70	57
JOB0002/00005	600	105	75	69
JOB0002/00006	765	105	75	83
JOB0002/00007	625	105	75	69
JOB0002/00008	279	80	70	74
JOB0002/00009	585	105	75	69
JOB0002/00010	284	80	70	74
JOB0002/00011	590	105	75	69
JOB0002/00012	582	105	75	69
JOB0002/00013	585	105	75	69
JOB0002/00014	296	105	75	40
JOB0002/00015	719	105	75	87
JOB0002/00016	462	120	81	70
JOB0002/00017	646	105	75	72
JOB0002/00018	223	105	75	60
JOB0002/00019	476	80	70	88
JOB0002/00020	425	80	70	89

Continued on next page

Table C.6 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0002/00021	450	105	75	55
JOB0002/00022	566	105	75	69
JOB0002/00023	187	80	70	43
JOB0002/00024	323	105	75	73
JOB0003/00001	690	120	81	63
JOB0003/00002	300	80	70	89
JOB0003/00003	690	120	81	64
JOB0003/00004	669	120	81	64
JOB0003/00005	416	120	81	46
JOB0003/00006	815	120	81	64
JOB0003/00007	657	120	81	75
JOB0003/00008	689	120	81	75
JOB0003/00009	576	120	81	63
JOB0003/00010	610	120	81	63
JOB0003/00011	719	120	81	75
JOB0003/00012	604	120	81	75
JOB0003/00013	604	120	81	75
JOB0003/00014	360	80	70	65
JOB0003/00015	619	120	81	63
JOB0003/00016	411	120	81	63
JOB0004/00001	403	80	70	95
JOB0004/00002	243	80	70	57
JOB0004/00003	389	105	75	84
JOB0004/00004	391	105	75	84
JOB0004/00005	293	80	70	73
JOB0004/00006	340	80	70	73
JOB0004/00007	243	105	75	48
JOB0004/00008	308	80	70	74
JOB0004/00009	282	80	70	73
JOB0004/00010	329	80	70	73

Continued on next page

Table C.6 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0004/00011	312	80	70	76
JOB0004/00012	389	105	75	62
JOB0004/00013	217	80	70	43
JOB0004/00014	248	80	70	62
JOB0004/00015	530	105	75	74
JOB0004/00016	194	80	70	76
JOB0004/00017	199	80	70	76
JOB0004/00018	199	80	70	57
JOB0004/00019	273	80	70	77
JOB0004/00020	322	80	70	73
JOB0004/00021	412	80	70	83
JOB0004/00022	248	80	70	73
JOB0005/00001	460	80	70	65
JOB0005/00002	203	105	75	95
JOB0005/00003	303	105	75	76
JOB0005/00004	555	105	75	57
JOB0005/00005	271	80	70	78
JOB0005/00006	323	80	70	77
JOB0005/00007	236	80	70	66
JOB0005/00008	301	80	70	72
JOB0005/00009	323	105	75	74
JOB0005/00010	477	80	70	91
JOB0005/00011	321	105	75	73
JOB0005/00012	188	120	81	67
JOB0005/00013	239	105	75	69
JOB0005/00014	347	105	75	93
JOB0005/00015	347	105	75	93
JOB0005/00016	479	105	75	71
JOB0006/00001	364	105	75	72
JOB0006/00002	635	120	81	74

Continued on next page

Table C.6 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0006/00003	599	105	75	72
JOB0006/00004	729	120	81	74
JOB0006/00005	464	120	81	69
JOB0006/00006	247	105	75	58
JOB0006/00007	267	80	70	62
JOB0006/00008	408	105	75	61
JOB0006/00009	414	80	70	74
JOB0006/00010	287	80	70	77
JOB0006/00011	131	80	70	52
JOB0006/00012	222	80	70	66
JOB0006/00013	359	80	70	58
JOB0006/00014	422	80	70	87
JOB0006/00015	294	80	70	74
JOB0006/00016	743	105	75	72
JOB0006/00017	395	105	75	69
JOB0006/00018	322	80	70	66
JOB0006/00019	339	80	70	69
JOB0007/00001	330	105	75	71
JOB0007/00002	538	120	81	95
JOB0007/00003	704	120	81	90
JOB0007/00004	606	120	81	73
JOB0007/00005	346	105	75	76
JOB0007/00006	700	105	75	86
JOB0007/00007	340	80	70	95
JOB0007/00008	346	80	70	85
JOB0007/00009	577	105	75	71
JOB0007/00010	413	105	75	52
JOB0007/00011	690	105	75	71
JOB0007/00012	288	80	70	79
JOB0008/00001	217	80	70	58

Continued on next page

Table C.6 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0008/00002	352	80	70	82
JOB0008/00003	380	105	75	46
JOB0008/00004	386	80	70	70
JOB0008/00005	243	80	70	59
JOB0008/00006	353	80	70	80
JOB0008/00007	478	80	70	93
JOB0008/00008	237	80	70	79
JOB0009/00001	236	80	70	57
JOB0009/00002	258	105	75	73
JOB0009/00003	548	120	81	75
JOB0009/00004	186	80	70	39
JOB0009/00005	408	80	70	66
JOB0009/00006	353	80	70	75
JOB0009/00007	313	80	70	75
JOB0009/00008	508	105	75	89
JOB0009/00009	293	80	70	73
JOB0009/00010	583	105	75	72
JOB0010/00001	226	80	70	57
JOB0010/00002	684	120	81	77
JOB0010/00003	260	80	70	58
JOB0010/00004	316	80	70	71
JOB0010/00005	534	120	81	75
JOB0011/00001	679	105	75	83
JOB0011/00002	267	105	75	56
JOB0011/00003	310	105	75	74
JOB0011/00004	363	80	70	89
JOB0011/00005	60	80	60	40
JOB0011/00006	807	105	75	74
JOB0011/00007	329	80	70	85
JOB0011/00008	318	105	75	74

Continued on next page

Table C.6 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0011/00009	375	120	81	39
JOB0011/00010	825	105	75	74
JOB0011/00011	637	105	75	74
JOB0011/00012	348	80	70	82
JOB0011/00013	398	80	70	93
JOB0012/00001	660	120	81	74
JOB0012/00002	416	120	81	60
JOB0012/00003	442	80	70	82
JOB0013/00001	64	80	70	42
JOB0013/00002	380	105	75	72
JOB0013/00003	329	105	75	72
JOB0013/00004	275	80	70	70
JOB0013/00005	675	105	75	71
JOB0013/00006	516	120	81	59
JOB0013/00007	105	80	70	49
JOB0013/00008	748	105	75	86
JOB0013/00009	745	105	75	70
JOB0013/00010	303	80	70	76
JOB0013/00011	410	105	75	55
JOB0013/00012	465	105	75	69
JOB0013/00013	128	80	70	67
JOB0013/00014	482	105	75	70
JOB0014/00001	44	80	60	31
JOB0015/00001	14	80	60	30
JOB0016/00001	394	80	70	82
JOB0016/00002	385	105	75	62
JOB0016/00003	803	105	75	81
JOB0016/00004	197	80	70	73
JOB0016/00005	212	80	70	57
JOB0016/00006	347	80	70	93

Continued on next page

Table C.6 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0016/00007	305	80	70	88
JOB0016/00008	554	105	75	65
JOB0016/00009	410	105	75	89
JOB0016/00010	314	105	75	57
JOB0016/00011	325	80	70	89
JOB0016/00012	311	80	70	77
JOB0017/00001	104	80	70	28
JOB0018/00001	349	80	70	75
JOB0018/00002	361	105	75	47
JOB0018/00003	880	120	81	82
JOB0018/00004	710	105	75	71
JOB0018/00005	461	105	75	62
JOB0018/00006	533	105	75	71
JOB0018/00007	781	120	81	84
JOB0018/00008	435	105	75	86
JOB0018/00009	226	80	70	56
JOB0018/00010	240	80	70	80
JOB0018/00011	257	80	70	69
JOB0019/00001	224	80	70	55
JOB0019/00002	580	105	75	66
JOB0019/00003	642	105	75	69
JOB0019/00004	639	105	75	82
JOB0019/00005	392	105	75	56
JOB0019/00006	236	105	75	73
JOB0019/00007	479	80	70	83
JOB0019/00008	167	80	70	85
JOB0019/00009	190	105	75	87
JOB0019/00010	493	105	75	66
JOB0019/00011	232	105	75	50
JOB0019/00012	331	105	75	74

Continued on next page

Table C.6 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0019/00013	334	105	75	74
JOB0019/00014	463	105	75	72
JOB0019/00015	367	80	70	96
JOB0019/00016	187	80	70	58
JOB0019/00017	149	80	70	53
JOB0020/00001	504	105	75	63
JOB0020/00002	253	80	70	62
JOB0020/00003	273	80	70	92
JOB0020/00004	327	105	75	58
JOB0020/00005	744	105	75	70
JOB0020/00006	310	105	75	75
JOB0020/00007	953	105	75	83
JOB0020/00008	412	105	75	65
JOB0020/00009	659	105	75	80
JOB0020/00010	531	105	75	71
JOB0020/00011	368	105	75	70
JOB0020/00012	701	105	75	84
JOB0020/00013	369	105	75	70
JOB0020/00014	321	80	70	82
JOB0020/00015	205	80	70	58
JOB0020/00016	675	120	81	72
JOB0020/00017	280	105	75	57
JOB0021/00001	228	80	70	45
JOB0022/00001	646	120	81	71
JOB0022/00002	674	120	81	67
JOB0022/00003	656	105	75	70
JOB0022/00004	786	105	75	74
JOB0022/00005	444	80	70	76
JOB0022/00006	524	105	75	74
JOB0023/00001	277	105	75	41

Continued on next page

Table C.6 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0023/00002	326	105	75	54
JOB0023/00003	582	105	75	81
JOB0023/00004	399	105	75	62
JOB0023/00005	558	105	75	73
JOB0023/00006	750	105	75	88
JOB0023/00007	367	80	70	91
JOB0023/00008	240	80	70	72
JOB0023/00009	416	80	70	84
JOB0023/00010	384	80	70	84
JOB0023/00011	408	105	75	66
JOB0023/00012	447	120	81	62
JOB0024/00001	104	80	70	76
JOB0024/00002	104	80	70	76
JOB0024/00003	565	105	75	72
JOB0024/00004	105	80	70	74
JOB0024/00005	206	80	70	69
JOB0024/00006	112	80	70	74
JOB0024/00007	303	80	70	76
JOB0024/00008	161	80	70	51
JOB0024/00009	202	80	70	76
JOB0024/00010	105	80	70	76
JOB0024/00011	111	80	70	74
JOB0024/00012	353	80	70	76
JOB0024/00013	232	80	70	76
JOB0024/00014	112	80	70	74
JOB0024/00015	111	80	70	74
JOB0024/00016	155	80	70	91
JOB0024/00017	201	80	70	76
JOB0024/00018	106	80	70	74
JOB0024/00019	112	80	70	74

Continued on next page

Table C.6 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0024/00020	111	80	70	74
JOB0024/00021	112	80	70	74
JOB0024/00022	112	80	70	74
JOB0024/00023	112	80	70	74
JOB0024/00024	377	105	75	85
JOB0024/00025	339	105	75	54
JOB0024/00026	561	105	75	73
JOB0024/00027	373	105	75	73
JOB0024/00028	553	120	81	75
JOB0024/00029	565	105	75	73
JOB0024/00030	583	105	75	73
JOB0024/00031	549	105	75	73
JOB0024/00032	353	80	70	74
JOB0024/00033	150	105	75	61
JOB0024/00034	156	105	75	61
JOB0024/00035	399	105	75	73
JOB0024/00036	321	105	75	73
JOB0024/00037	405	105	75	72
JOB0024/00038	323	105	75	73
JOB0024/00039	392	105	75	72
JOB0024/00040	698	120	81	76
JOB0024/00041	344	105	75	72
JOB0024/00042	366	105	75	71
JOB0024/00043	518	120	81	85
JOB0025/00001	790	120	81	91
JOB0025/00002	914	120	81	91
JOB0025/00003	788	120	81	91
JOB0025/00004	940	120	81	91
JOB0025/00005	341	80	70	74

Continued on next page

Table C.6 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0025/00006	741	120	81	76

Table C.7: Problem Set #7

Pallet number	Weight	Length	Breadth	Height
JOB0001/00001	354	80	70	55
JOB0001/00002	589	120	81	63
JOB0001/00003	589	120	81	63
JOB0001/00004	586	120	81	63
JOB0001/00005	589	120	81	63
JOB0001/00006	590	120	81	63
JOB0001/00007	591	120	81	63
JOB0001/00008	241	80	70	61
JOB0001/00009	312	80	70	75
JOB0001/00010	410	80	70	75
JOB0001/00011	306	80	70	75
JOB0001/00012	898	120	81	75
JOB0001/00013	198	80	70	58
JOB0001/00014	201	80	70	59
JOB0001/00015	923	120	81	75
JOB0002/00001	351	120	81	64
JOB0002/00002	378	80	70	80
JOB0002/00003	398	80	70	75
JOB0002/00004	459	80	70	75
JOB0002/00005	468	80	70	75
JOB0003/00001	371	105	75	68
JOB0003/00002	643	105	75	72
JOB0003/00003	322	80	70	87
JOB0003/00004	327	105	75	75
JOB0003/00005	600	105	75	72
JOB0003/00006	328	80	70	84
JOB0003/00007	212	80	70	77
JOB0003/00008	269	105	75	74

Continued on next page

Table C.7 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0003/00009	310	80	70	87
JOB0003/00010	330	105	75	75
JOB0003/00011	369	80	70	78
JOB0003/00012	578	105	75	73
JOB0003/00013	566	105	75	72
JOB0003/00014	225	80	70	56
JOB0003/00015	331	80	70	73
JOB0003/00016	172	80	70	72
JOB0003/00017	179	80	70	69
JOB0004/00001	212	80	70	62
JOB0004/00002	304	80	70	73
JOB0004/00003	282	80	70	76
JOB0004/00004	310	80	70	75
JOB0004/00005	306	80	70	88
JOB0004/00006	136	80	70	60
JOB0004/00007	522	105	75	69
JOB0004/00008	303	80	70	74
JOB0005/00001	503	105	75	72
JOB0006/00001	179	80	70	79
JOB0006/00002	452	80	70	87
JOB0007/00001	384	105	75	57
JOB0007/00002	563	105	75	67
JOB0008/00001	558	120	81	61
JOB0008/00002	341	80	70	74
JOB0008/00003	296	80	70	74
JOB0009/00001	166	80	70	78
JOB0009/00002	388	80	70	84
JOB0009/00003	417	80	70	92
JOB0009/00004	464	80	70	96
JOB0009/00005	299	80	70	63

Continued on next page

Table C.7 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0009/00006	543	105	75	75
JOB0009/00007	350	80	70	90
JOB0009/00008	296	80	70	89
JOB0010/00001	294	80	70	82
JOB0010/00002	450	105	75	80
JOB0010/00003	169	80	70	63
JOB0010/00004	431	120	81	61
JOB0010/00005	580	105	75	70
JOB0010/00006	609	105	75	74
JOB0010/00007	241	80	70	65
JOB0010/00008	601	105	75	74
JOB0010/00009	428	80	70	82
JOB0010/00010	222	80	70	65
JOB0010/00011	327	80	70	74
JOB0010/00012	222	80	70	75
JOB0010/00013	760	105	75	89
JOB0010/00014	549	105	75	72
JOB0010/00015	474	105	75	72
JOB0011/00001	298	80	70	76
JOB0011/00002	342	105	75	71
JOB0011/00003	294	80	70	76
JOB0011/00004	296	80	70	74
JOB0011/00005	494	80	70	83
JOB0011/00006	392	80	70	75
JOB0011/00007	407	80	70	92
JOB0011/00008	394	80	70	77
JOB0011/00009	477	105	75	71
JOB0011/00010	346	80	70	73
JOB0011/00011	186	80	70	75
JOB0011/00012	354	80	70	73

Continued on next page

Table C.7 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0011/00013	411	80	70	74
JOB0011/00014	222	80	70	44
JOB0011/00015	115	80	60	47
JOB0012/00001	365	80	70	75
JOB0012/00002	625	105	75	60
JOB0012/00003	612	105	75	82
JOB0012/00004	404	105	75	57
JOB0012/00005	348	80	70	91
JOB0012/00006	627	105	75	61
JOB0012/00007	459	105	75	86
JOB0012/00008	357	105	75	71
JOB0012/00009	366	120	81	65
JOB0012/00010	244	80	70	65
JOB0012/00011	356	80	70	92
JOB0012/00012	445	80	70	97
JOB0012/00013	503	105	75	65
JOB0013/00001	551	105	75	72
JOB0013/00002	262	80	70	70
JOB0013/00003	605	105	75	71
JOB0013/00004	369	105	75	56
JOB0013/00005	641	105	75	71
JOB0013/00006	451	105	75	60
JOB0013/00007	374	105	75	60
JOB0013/00008	377	80	70	73
JOB0013/00009	407	80	70	73
JOB0013/00010	358	80	70	86
JOB0013/00011	624	120	81	85
JOB0013/00012	766	105	75	72
JOB0014/00001	61	80	70	58
JOB0014/00002	500	80	70	83

Continued on next page

Table C.7 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0014/00003	318	80	70	73
JOB0014/00004	803	105	75	81
JOB0014/00005	357	105	75	62
JOB0014/00006	478	105	75	65
JOB0014/00007	594	105	75	79
JOB0014/00008	421	80	70	80
JOB0014/00009	320	80	70	89
JOB0014/00010	432	80	70	73
JOB0014/00011	204	80	70	56
JOB0014/00012	173	80	70	49
JOB0015/00001	296	80	70	80
JOB0015/00002	341	80	70	72
JOB0015/00003	280	80	70	72
JOB0015/00004	479	80	70	91
JOB0015/00005	577	105	75	72
JOB0015/00006	116	105	75	56
JOB0015/00007	346	80	70	82
JOB0015/00008	500	105	75	72
JOB0015/00009	429	80	70	81
JOB0015/00010	337	80	70	72
JOB0015/00011	347	80	70	72
JOB0015/00012	360	80	70	70
JOB0015/00013	330	80	70	75
JOB0015/00014	431	80	70	73
JOB0016/00001	487	80	70	76
JOB0016/00002	492	80	70	84
JOB0016/00003	383	80	70	74
JOB0016/00004	249	80	70	64
JOB0016/00005	357	80	70	79
JOB0016/00006	323	80	70	81

Continued on next page

Table C.7 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0016/00007	293	80	70	61
JOB0016/00008	265	80	70	74
JOB0016/00009	515	105	75	72
JOB0016/00010	261	105	75	76
JOB0016/00011	629	105	75	72
JOB0016/00012	486	105	75	73
JOB0016/00013	269	80	70	72
JOB0016/00014	167	80	70	76
JOB0016/00015	244	80	70	63
JOB0017/00001	535	105	75	71
JOB0017/00002	252	80	70	64
JOB0017/00003	300	80	70	73
JOB0017/00004	244	80	70	73
JOB0017/00005	400	80	70	72
JOB0017/00006	379	80	70	82
JOB0017/00007	322	80	70	75
JOB0017/00008	370	80	70	61
JOB0017/00009	345	80	70	73
JOB0017/00010	440	80	70	79
JOB0017/00011	440	80	70	81
JOB0017/00012	405	105	75	63
JOB0018/00001	547	105	75	80
JOB0018/00002	273	80	70	65
JOB0018/00003	377	105	75	62
JOB0018/00004	279	80	70	61
JOB0018/00005	278	105	75	63
JOB0018/00006	513	105	75	75
JOB0018/00007	382	105	75	74
JOB0018/00008	175	80	70	64
JOB0018/00009	407	80	70	87

Continued on next page

Table C.7 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0018/00010	332	80	70	82
JOB0019/00001	703	120	81	63
JOB0019/00002	524	120	81	63
JOB0019/00003	703	120	81	63
JOB0019/00004	770	120	81	79
JOB0020/00001	241	105	75	71
JOB0020/00002	230	105	75	51
JOB0020/00003	254	105	75	72
JOB0020/00004	256	105	75	72
JOB0020/00005	167	80	70	81
JOB0020/00006	253	105	75	72
JOB0020/00007	199	80	70	51
JOB0020/00008	131	105	75	36
JOB0020/00009	369	105	75	64
JOB0021/00001	355	80	70	74
JOB0021/00002	176	80	60	53
JOB0021/00003	382	80	70	74
JOB0021/00004	569	120	81	77
JOB0021/00005	282	80	70	60
JOB0021/00006	380	80	70	73
JOB0021/00007	418	80	70	74
JOB0021/00008	320	80	70	74
JOB0021/00009	248	80	70	59
JOB0021/00010	852	120	81	69
JOB0021/00011	530	120	81	60
JOB0021/00012	618	120	81	69
JOB0021/00013	806	120	81	69
JOB0021/00014	351	120	81	53
JOB0021/00015	767	120	81	83
JOB0022/00001	428	80	70	90

Continued on next page

Table C.7 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0022/00002	511	120	81	63
JOB0022/00003	410	120	81	55
JOB0022/00004	511	120	81	63
JOB0022/00005	297	80	70	84
JOB0023/00001	269	80	70	67
JOB0023/00002	396	80	70	74
JOB0023/00003	208	80	70	74
JOB0023/00004	256	80	70	74
JOB0023/00005	171	80	70	69
JOB0023/00006	320	80	70	62
JOB0023/00007	632	105	75	71
JOB0023/00008	311	80	70	78
JOB0023/00009	180	80	70	54
JOB0023/00010	271	80	70	64
JOB0023/00011	235	80	70	66
JOB0023/00012	261	80	70	57
JOB0024/00001	391	80	70	80
JOB0024/00002	429	80	70	73
JOB0024/00003	366	80	70	73
JOB0024/00004	317	80	70	81
JOB0024/00005	410	80	70	98
JOB0024/00006	222	80	70	80
JOB0024/00007	500	80	70	98
JOB0024/00008	234	80	70	66
JOB0024/00009	717	105	75	71
JOB0024/00010	389	80	70	73
JOB0024/00011	322	80	70	91
JOB0024/00012	421	80	70	85

Table C.8: Problem Set #8

Pallet number	Weight	Length	Breadth	Height
JOB0001/00001	217	80	70	58
JOB0001/00002	352	80	70	82
JOB0001/00003	380	105	75	46
JOB0001/00004	386	80	70	70
JOB0001/00005	243	80	70	59
JOB0001/00006	353	80	70	80
JOB0001/00007	478	80	70	93
JOB0001/00008	237	80	70	79
JOB0002/00001	236	80	70	57
JOB0002/00002	258	105	75	73
JOB0002/00003	548	120	81	75
JOB0002/00004	186	80	70	39
JOB0002/00005	408	80	70	66
JOB0002/00006	353	80	70	75
JOB0002/00007	313	80	70	75
JOB0002/00008	508	105	75	89
JOB0002/00009	293	80	70	73
JOB0002/00010	583	105	75	72
JOB0003/00001	226	80	70	57
JOB0003/00002	684	120	81	77
JOB0003/00003	260	80	70	58
JOB0003/00004	316	80	70	71
JOB0003/00005	534	120	81	75
JOB0004/00001	679	105	75	83
JOB0004/00002	267	105	75	56
JOB0004/00003	310	105	75	74
JOB0004/00004	363	80	70	89
JOB0004/00005	60	80	60	40

Continued on next page

Table C.8 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0004/00006	807	105	75	74
JOB0004/00007	329	80	70	85
JOB0004/00008	318	105	75	74
JOB0004/00009	375	120	81	39
JOB0004/00010	825	105	75	74
JOB0004/00011	637	105	75	74
JOB0004/00012	348	80	70	82
JOB0004/00013	398	80	70	93
JOB0005/00001	660	120	81	74
JOB0005/00002	416	120	81	60
JOB0005/00003	442	80	70	82
JOB0006/00001	64	80	70	42
JOB0006/00002	380	105	75	72
JOB0006/00003	329	105	75	72
JOB0006/00004	275	80	70	70
JOB0006/00005	675	105	75	71
JOB0006/00006	516	120	81	59
JOB0006/00007	105	80	70	49
JOB0006/00008	748	105	75	86
JOB0006/00009	745	105	75	70
JOB0006/00010	303	80	70	76
JOB0006/00011	410	105	75	55
JOB0006/00012	465	105	75	69
JOB0006/00013	128	80	70	67
JOB0006/00014	482	105	75	70
JOB0007/00001	44	80	60	31
JOB0008/00001	14	80	60	30
JOB0009/00001	394	80	70	82
JOB0009/00002	385	105	75	62
JOB0009/00003	803	105	75	81

Continued on next page

Table C.8 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0009/00004	197	80	70	73
JOB0009/00005	212	80	70	57
JOB0009/00006	347	80	70	93
JOB0009/00007	305	80	70	88
JOB0009/00008	554	105	75	65
JOB0009/00009	410	105	75	89
JOB0009/00010	314	105	75	57
JOB0009/00011	325	80	70	89
JOB0009/00012	311	80	70	77
JOB0010/00001	104	80	70	28
JOB0011/00001	349	80	70	75
JOB0011/00002	361	105	75	47
JOB0011/00003	880	120	81	82
JOB0011/00004	710	105	75	71
JOB0011/00005	461	105	75	62
JOB0011/00006	533	105	75	71
JOB0011/00007	781	120	81	84
JOB0011/00008	435	105	75	86
JOB0011/00009	226	80	70	56
JOB0011/00010	240	80	70	80
JOB0011/00011	257	80	70	69
JOB0012/00001	224	80	70	55
JOB0012/00002	580	105	75	66
JOB0012/00003	642	105	75	69
JOB0012/00004	639	105	75	82
JOB0012/00005	392	105	75	56
JOB0012/00006	236	105	75	73
JOB0012/00007	479	80	70	83
JOB0012/00008	167	80	70	85
JOB0012/00009	190	105	75	87

Continued on next page

Table C.8 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0012/00010	493	105	75	66
JOB0012/00011	232	105	75	50
JOB0012/00012	331	105	75	74
JOB0012/00013	334	105	75	74
JOB0012/00014	463	105	75	72
JOB0012/00015	367	80	70	96
JOB0012/00016	187	80	70	58
JOB0012/00017	149	80	70	53
JOB0013/00001	504	105	75	63
JOB0013/00002	253	80	70	62
JOB0013/00003	273	80	70	92
JOB0013/00004	327	105	75	58
JOB0013/00005	744	105	75	70
JOB0013/00006	310	105	75	75
JOB0013/00007	953	105	75	83
JOB0013/00008	412	105	75	65
JOB0013/00009	659	105	75	80
JOB0013/00010	531	105	75	71
JOB0013/00011	368	105	75	70
JOB0013/00012	701	105	75	84
JOB0013/00013	369	105	75	70
JOB0013/00014	321	80	70	82
JOB0013/00015	205	80	70	58
JOB0013/00016	675	120	81	72
JOB0013/00017	280	105	75	57
JOB0014/00001	228	80	70	45
JOB0015/00001	646	120	81	71
JOB0015/00002	674	120	81	67
JOB0015/00003	656	105	75	70
JOB0015/00004	786	105	75	74

Continued on next page

Table C.8 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0015/00005	444	80	70	76
JOB0015/00006	524	105	75	74
JOB0016/00001	277	105	75	41
JOB0016/00002	326	105	75	54
JOB0016/00003	582	105	75	81
JOB0016/00004	399	105	75	62
JOB0016/00005	558	105	75	73
JOB0016/00006	750	105	75	88
JOB0016/00007	367	80	70	91
JOB0016/00008	240	80	70	72
JOB0016/00009	416	80	70	84
JOB0016/00010	384	80	70	84
JOB0016/00011	408	105	75	66
JOB0016/00012	447	120	81	62
JOB0017/00001	104	80	70	76
JOB0017/00002	104	80	70	76
JOB0017/00003	565	105	75	72
JOB0017/00004	105	80	70	74
JOB0017/00005	206	80	70	69
JOB0017/00006	112	80	70	74
JOB0017/00007	303	80	70	76
JOB0017/00008	161	80	70	51
JOB0017/00009	202	80	70	76
JOB0017/00010	105	80	70	76
JOB0017/00011	111	80	70	74
JOB0017/00012	353	80	70	76
JOB0017/00013	232	80	70	76
JOB0017/00014	112	80	70	74
JOB0017/00015	111	80	70	74
JOB0017/00016	155	80	70	91

Continued on next page

Table C.8 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0017/00017	201	80	70	76
JOB0017/00018	106	80	70	74
JOB0017/00019	112	80	70	74
JOB0017/00020	111	80	70	74
JOB0017/00021	112	80	70	74
JOB0017/00022	112	80	70	74
JOB0017/00023	112	80	70	74
JOB0017/00024	377	105	75	85
JOB0017/00025	339	105	75	54
JOB0017/00026	561	105	75	73
JOB0017/00027	373	105	75	73
JOB0017/00028	553	120	81	75
JOB0017/00029	565	105	75	73
JOB0017/00030	583	105	75	73
JOB0017/00031	549	105	75	73
JOB0017/00032	353	80	70	74
JOB0017/00033	150	105	75	61
JOB0017/00034	156	105	75	61
JOB0017/00035	399	105	75	73
JOB0017/00036	321	105	75	73
JOB0017/00037	405	105	75	72
JOB0017/00038	323	105	75	73
JOB0017/00039	392	105	75	72
JOB0017/00040	698	120	81	76
JOB0017/00041	344	105	75	72
JOB0017/00042	366	105	75	71
JOB0017/00043	518	120	81	85
JOB0018/00001	790	120	81	91
JOB0018/00002	914	120	81	91
JOB0018/00003	788	120	81	91

Continued on next page

Table C.8 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0018/00004	940	120	81	91
JOB0018/00005	341	80	70	74
JOB0018/00006	741	120	81	76

Table C.9: Problem Set #9

Pallet number	Weight	Length	Breadth	Height
JOB0001/00001	737	105	75	82
JOB0001/00002	736	105	75	82
JOB0001/00003	737	105	75	82
JOB0001/00004	409	80	70	84
JOB0001/00005	355	80	70	83
JOB0001/00006	402	80	70	83
JOB0001/00007	640	105	75	81
JOB0001/00008	656	105	75	81
JOB0001/00009	316	105	75	72
JOB0001/00010	733	105	75	82
JOB0001/00011	732	105	75	82
JOB0001/00012	212	80	70	77
JOB0001/00013	231	80	70	73
JOB0001/00014	584	105	75	83
JOB0001/00015	416	105	75	72
JOB0002/00001	616	105	75	72
JOB0002/00002	600	105	75	72
JOB0002/00003	125	80	60	46
JOB0002/00004	361	80	70	84
JOB0002/00005	472	80	70	92
JOB0002/00006	352	80	70	82
JOB0002/00007	583	105	75	71
JOB0002/00008	680	105	75	73
JOB0002/00009	370	80	70	83
JOB0003/00001	335	80	70	83
JOB0003/00002	444	80	70	83
JOB0003/00003	307	80	70	73
JOB0003/00004	825	120	81	82

Continued on next page

Table C.9 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0003/00005	315	80	70	81
JOB0003/00006	286	80	70	76
JOB0003/00007	250	80	70	72
JOB0003/00008	389	120	81	83
JOB0003/00009	683	105	75	82
JOB0003/00010	484	105	75	63
JOB0003/00011	230	105	75	41
JOB0003/00012	441	105	75	71
JOB0003/00013	696	105	75	82
JOB0003/00014	75	80	70	43
JOB0004/00001	193	80	70	60
JOB0004/00002	363	105	75	61
JOB0004/00003	394	80	70	82
JOB0004/00004	254	80	70	62
JOB0004/00005	484	105	75	62
JOB0004/00006	594	105	75	77
JOB0004/00007	212	80	70	60
JOB0004/00008	189	80	70	65
JOB0004/00009	446	105	75	56
JOB0004/00010	539	105	75	65
JOB0004/00011	613	105	75	75
JOB0004/00012	608	105	75	72
JOB0004/00013	598	105	75	72
JOB0004/00014	208	80	60	66
JOB0004/00015	555	105	75	70
JOB0004/00016	559	105	75	72
JOB0005/00001	599	120	81	74
JOB0005/00002	709	105	75	72
JOB0005/00003	708	105	75	72
JOB0005/00004	341	80	70	89

Continued on next page

Table C.9 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0005/00005	620	105	75	72
JOB0005/00006	641	105	75	72
JOB0005/00007	563	120	81	67
JOB0005/00008	663	105	75	71
JOB0005/00009	639	105	75	72
JOB0005/00010	522	105	75	71
JOB0005/00011	329	80	70	73
JOB0005/00012	227	80	70	76
JOB0005/00013	291	80	70	76
JOB0005/00014	291	80	70	51
JOB0005/00015	348	80	70	78
JOB0005/00016	506	80	70	76
JOB0005/00017	586	105	75	71
JOB0005/00018	328	80	70	82
JOB0005/00019	518	105	75	71
JOB0005/00020	494	80	70	76
JOB0005/00021	353	80	70	73
JOB0005/00022	234	80	70	76
JOB0005/00023	260	80	70	73
JOB0005/00024	234	80	70	73
JOB0005/00025	331	80	70	73
JOB0005/00026	378	80	70	76
JOB0005/00027	354	80	70	87
JOB0006/00001	279	80	70	85
JOB0006/00002	290	80	70	85
JOB0006/00003	325	80	70	84
JOB0006/00004	237	80	70	74
JOB0006/00005	355	80	70	88
JOB0006/00006	350	80	70	96
JOB0006/00007	348	80	70	78

Continued on next page

Table C.9 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0006/00008	396	80	70	84
JOB0006/00009	331	80	70	79
JOB0006/00010	358	105	75	67
JOB0006/00011	317	80	70	74
JOB0006/00012	398	80	70	85
JOB0006/00013	394	80	70	84
JOB0006/00014	398	105	75	61
JOB0006/00015	399	80	70	84
JOB0006/00016	362	120	81	78
JOB0006/00017	528	105	75	91
JOB0007/00001	679	120	81	90
JOB0007/00002	783	105	75	86
JOB0007/00003	188	80	70	62
JOB0007/00004	742	105	75	71
JOB0007/00005	268	80	70	58
JOB0007/00006	334	80	70	79
JOB0007/00007	656	105	75	85
JOB0007/00008	308	80	70	83
JOB0007/00009	725	105	75	84
JOB0007/00010	726	105	75	84
JOB0007/00011	399	80	70	88
JOB0007/00012	348	80	70	72
JOB0007/00013	348	80	70	72
JOB0007/00014	176	80	70	44
JOB0008/00001	435	80	70	92
JOB0008/00002	614	105	75	76
JOB0008/00003	263	80	70	57
JOB0008/00004	317	80	70	78
JOB0008/00005	642	105	75	77
JOB0008/00006	306	80	70	79

Continued on next page

Table C.9 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0008/00007	502	80	70	92
JOB0009/00001	324	80	70	79
JOB0009/00002	443	80	70	90
JOB0009/00003	433	80	70	82
JOB0009/00004	323	80	70	59
JOB0009/00005	795	120	81	75
JOB0009/00006	573	105	75	82
JOB0009/00007	666	120	81	79
JOB0009/00008	602	105	75	76
JOB0009/00009	620	105	75	68
JOB0009/00010	586	105	75	71
JOB0010/00001	477	80	70	87
JOB0010/00002	341	105	75	47
JOB0010/00003	601	105	75	70
JOB0010/00004	578	105	75	74
JOB0010/00005	442	80	70	94
JOB0010/00006	848	105	75	89
JOB0010/00007	268	80	70	71
JOB0010/00008	336	80	70	76
JOB0010/00009	232	80	70	72
JOB0011/00001	670	105	75	75
JOB0011/00002	281	105	75	66
JOB0011/00003	368	105	75	83
JOB0011/00004	534	105	75	71
JOB0011/00005	455	105	75	71
JOB0011/00006	371	80	70	78
JOB0011/00007	280	105	75	66
JOB0011/00008	535	105	75	71
JOB0011/00009	596	105	75	71
JOB0011/00010	595	105	75	71

Continued on next page

Table C.9 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0011/00011	534	105	75	71
JOB0011/00012	504	105	75	76
JOB0011/00013	536	105	75	71
JOB0011/00014	386	105	75	87
JOB0011/00015	364	80	70	79
JOB0011/00016	201	80	70	46
JOB0012/00001	281	80	70	92
JOB0012/00002	400	80	70	67
JOB0012/00003	778	105	75	71
JOB0012/00004	209	80	70	62
JOB0012/00005	359	80	70	83
JOB0012/00006	339	80	70	84
JOB0012/00007	460	80	70	94
JOB0012/00008	403	105	75	87
JOB0012/00009	311	105	75	71
JOB0013/00001	169	80	70	59
JOB0013/00002	306	80	70	76
JOB0013/00003	875	120	81	76
JOB0014/00001	359	80	70	75
JOB0014/00002	486	80	70	88
JOB0014/00003	630	105	75	80
JOB0014/00004	516	105	75	70
JOB0014/00005	651	105	75	74
JOB0014/00006	464	105	75	74
JOB0014/00007	490	105	75	60
JOB0014/00008	379	105	75	57
JOB0014/00009	291	80	70	73
JOB0015/00001	237	80	70	61
JOB0015/00002	352	80	70	72
JOB0015/00003	443	105	75	64

Continued on next page

Table C.9 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0015/00004	702	105	75	83
JOB0015/00005	476	80	70	96
JOB0015/00006	335	80	70	73
JOB0015/00007	581	105	75	79
JOB0015/00008	391	105	75	65
JOB0015/00009	607	105	75	64
JOB0016/00001	238	80	70	77
JOB0016/00002	412	105	75	69
JOB0016/00003	613	105	75	79
JOB0016/00004	217	80	70	64
JOB0016/00005	198	80	70	59
JOB0016/00006	389	80	70	76
JOB0016/00007	309	80	70	79
JOB0016/00008	358	80	70	66
JOB0016/00009	341	80	70	75
JOB0017/00001	14	80	60	31
JOB0018/00001	36	80	60	29
JOB0019/00001	220	120	81	38
JOB0020/00001	370	80	70	59
JOB0020/00002	880	105	75	92
JOB0020/00003	536	105	75	71
JOB0020/00004	580	120	81	72
JOB0020/00005	812	105	75	70
JOB0020/00006	304	80	70	76
JOB0020/00007	800	105	75	70
JOB0020/00008	798	105	75	70
JOB0020/00009	348	80	70	64
JOB0020/00010	288	80	70	63
JOB0021/00001	325	80	70	77
JOB0021/00002	458	80	70	87

Continued on next page

Table C.9 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0021/00003	937	105	75	85
JOB0021/00004	340	80	70	88
JOB0021/00005	280	105	75	71
JOB0021/00006	229	105	75	51
JOB0021/00007	983	105	75	85
JOB0021/00008	513	105	75	57
JOB0021/00009	709	105	75	70

Table C.10: Problem Set #10

Pallet number	Weight	Length	Breadth	Height
JOB0001/00001	354	80	70	55
JOB0001/00002	589	120	81	63
JOB0001/00003	589	120	81	63
JOB0001/00004	586	120	81	63
JOB0001/00005	589	120	81	63
JOB0001/00006	590	120	81	63
JOB0001/00007	591	120	81	63
JOB0001/00008	241	80	70	61
JOB0001/00009	312	80	70	75
JOB0001/00010	410	80	70	75
JOB0001/00011	306	80	70	75
JOB0001/00012	898	120	81	75
JOB0001/00013	198	80	70	58
JOB0001/00014	201	80	70	59
JOB0001/00015	923	120	81	75
JOB0002/00001	351	120	81	64
JOB0002/00002	378	80	70	80
JOB0002/00003	398	80	70	75
JOB0002/00004	459	80	70	75
JOB0002/00005	468	80	70	75
JOB0003/00001	371	105	75	68
JOB0003/00002	643	105	75	72
JOB0003/00003	322	80	70	87
JOB0003/00004	327	105	75	75
JOB0003/00005	600	105	75	72
JOB0003/00006	328	80	70	84
JOB0003/00007	212	80	70	77
JOB0003/00008	269	105	75	74

Continued on next page

Table C.10 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0003/00009	310	80	70	87
JOB0003/00010	330	105	75	75
JOB0003/00011	369	80	70	78
JOB0003/00012	578	105	75	73
JOB0003/00013	566	105	75	72
JOB0003/00014	225	80	70	56
JOB0003/00015	331	80	70	73
JOB0003/00016	172	80	70	72
JOB0003/00017	179	80	70	69
JOB0004/00001	212	80	70	62
JOB0004/00002	304	80	70	73
JOB0004/00003	282	80	70	76
JOB0004/00004	310	80	70	75
JOB0004/00005	306	80	70	88
JOB0004/00006	136	80	70	60
JOB0004/00007	522	105	75	69
JOB0004/00008	303	80	70	74
JOB0005/00001	503	105	75	72
JOB0006/00001	179	80	70	79
JOB0006/00002	452	80	70	87
JOB0007/00001	384	105	75	57
JOB0007/00002	563	105	75	67
JOB0008/00001	558	120	81	61
JOB0008/00002	341	80	70	74
JOB0008/00003	296	80	70	74
JOB0009/00001	61	80	70	58
JOB0009/00002	500	80	70	83
JOB0009/00003	318	80	70	73
JOB0009/00004	803	105	75	81
JOB0009/00005	357	105	75	62

Continued on next page

Table C.10 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0009/00006	478	105	75	65
JOB0009/00007	594	105	75	79
JOB0009/00008	421	80	70	80
JOB0009/00009	320	80	70	89
JOB0009/00010	432	80	70	73
JOB0009/00011	204	80	70	56
JOB0009/00012	173	80	70	49
JOB0010/00001	296	80	70	80
JOB0010/00002	341	80	70	72
JOB0010/00003	280	80	70	72
JOB0010/00004	479	80	70	91
JOB0010/00005	577	105	75	72
JOB0010/00006	116	105	75	56
JOB0010/00007	346	80	70	82
JOB0010/00008	500	105	75	72
JOB0010/00009	429	80	70	81
JOB0010/00010	337	80	70	72
JOB0010/00011	347	80	70	72
JOB0010/00012	360	80	70	70
JOB0010/00013	330	80	70	75
JOB0010/00014	431	80	70	73
JOB0011/00001	487	80	70	76
JOB0011/00002	492	80	70	84
JOB0011/00003	383	80	70	74
JOB0011/00004	249	80	70	64
JOB0011/00005	357	80	70	79
JOB0011/00006	323	80	70	81
JOB0011/00007	293	80	70	61
JOB0011/00008	265	80	70	74
JOB0011/00009	515	105	75	72

Continued on next page

Table C.10 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0011/00010	261	105	75	76
JOB0011/00011	629	105	75	72
JOB0011/00012	486	105	75	73
JOB0011/00013	269	80	70	72
JOB0011/00014	167	80	70	76
JOB0011/00015	244	80	70	63
JOB0012/00001	535	105	75	71
JOB0012/00002	252	80	70	64
JOB0012/00003	300	80	70	73
JOB0012/00004	244	80	70	73
JOB0012/00005	400	80	70	72
JOB0012/00006	379	80	70	82
JOB0012/00007	322	80	70	75
JOB0012/00008	370	80	70	61
JOB0012/00009	345	80	70	73
JOB0012/00010	440	80	70	79
JOB0012/00011	440	80	70	81
JOB0012/00012	405	105	75	63
JOB0013/00001	547	105	75	80
JOB0013/00002	273	80	70	65
JOB0013/00003	377	105	75	62
JOB0013/00004	279	80	70	61
JOB0013/00005	278	105	75	63
JOB0013/00006	513	105	75	75
JOB0013/00007	382	105	75	74
JOB0013/00008	175	80	70	64
JOB0013/00009	407	80	70	87
JOB0013/00010	332	80	70	82
JOB0014/00001	703	120	81	63
JOB0014/00002	524	120	81	63

Continued on next page

Table C.10 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0014/00003	703	120	81	63
JOB0014/00004	770	120	81	79
JOB0015/00001	241	105	75	71
JOB0015/00002	230	105	75	51
JOB0015/00003	254	105	75	72
JOB0015/00004	256	105	75	72
JOB0015/00005	167	80	70	81
JOB0015/00006	253	105	75	72
JOB0015/00007	199	80	70	51
JOB0015/00008	131	105	75	36
JOB0015/00009	369	105	75	64
JOB0016/00001	355	80	70	74
JOB0016/00002	176	80	60	53
JOB0016/00003	382	80	70	74
JOB0016/00004	569	120	81	77
JOB0016/00005	282	80	70	60
JOB0016/00006	380	80	70	73
JOB0016/00007	418	80	70	74
JOB0016/00008	320	80	70	74
JOB0016/00009	248	80	70	59
JOB0016/00010	852	120	81	69
JOB0016/00011	530	120	81	60
JOB0016/00012	618	120	81	69
JOB0016/00013	806	120	81	69
JOB0016/00014	351	120	81	53
JOB0016/00015	767	120	81	83
JOB0017/00001	428	80	70	90
JOB0017/00002	511	120	81	63
JOB0017/00003	410	120	81	55
JOB0017/00004	511	120	81	63

Continued on next page

Table C.10 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0017/00005	297	80	70	84
JOB0018/00001	269	80	70	67
JOB0018/00002	396	80	70	74
JOB0018/00003	208	80	70	74
JOB0018/00004	256	80	70	74
JOB0018/00005	171	80	70	69
JOB0018/00006	320	80	70	62
JOB0018/00007	632	105	75	71
JOB0018/00008	311	80	70	78
JOB0018/00009	180	80	70	54
JOB0018/00010	271	80	70	64
JOB0018/00011	235	80	70	66
JOB0018/00012	261	80	70	57
JOB0019/00001	391	80	70	80
JOB0019/00002	429	80	70	73
JOB0019/00003	366	80	70	73
JOB0019/00004	317	80	70	81
JOB0019/00005	410	80	70	98
JOB0019/00006	222	80	70	80
JOB0019/00007	500	80	70	98
JOB0019/00008	234	80	70	66
JOB0019/00009	717	105	75	71
JOB0019/00010	389	80	70	73
JOB0019/00011	322	80	70	91
JOB0019/00012	421	80	70	85

Table C.11: Problem Set #11

Pallet number	Weight	Length	Breadth	Height
JOB0001/00001	349	80	70	75
JOB0001/00002	361	105	75	47
JOB0001/00003	880	120	81	82
JOB0001/00004	710	105	75	71
JOB0001/00005	461	105	75	62
JOB0001/00006	533	105	75	71
JOB0001/00007	781	120	81	84
JOB0001/00008	435	105	75	86
JOB0001/00009	226	80	70	56
JOB0001/00010	240	80	70	80
JOB0001/00011	257	80	70	69
JOB0002/00001	224	80	70	55
JOB0002/00002	580	105	75	66
JOB0002/00003	642	105	75	69
JOB0002/00004	639	105	75	82
JOB0002/00005	392	105	75	56
JOB0002/00006	236	105	75	73
JOB0002/00007	479	80	70	83
JOB0002/00008	167	80	70	85
JOB0002/00009	190	105	75	87
JOB0002/00010	493	105	75	66
JOB0002/00011	232	105	75	50
JOB0002/00012	331	105	75	74
JOB0002/00013	334	105	75	74
JOB0002/00014	463	105	75	72
JOB0002/00015	367	80	70	96
JOB0002/00016	187	80	70	58
JOB0002/00017	149	80	70	53

Continued on next page

Table C.11 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0003/00001	504	105	75	63
JOB0003/00002	253	80	70	62
JOB0003/00003	273	80	70	92
JOB0003/00004	327	105	75	58
JOB0003/00005	744	105	75	70
JOB0003/00006	310	105	75	75
JOB0003/00007	953	105	75	83
JOB0003/00008	412	105	75	65
JOB0003/00009	659	105	75	80
JOB0003/00010	531	105	75	71
JOB0003/00011	368	105	75	70
JOB0003/00012	701	105	75	84
JOB0003/00013	369	105	75	70
JOB0003/00014	321	80	70	82
JOB0003/00015	205	80	70	58
JOB0003/00016	675	120	81	72
JOB0003/00017	280	105	75	57
JOB0004/00001	228	80	70	45
JOB0005/00001	646	120	81	71
JOB0005/00002	674	120	81	67
JOB0005/00003	656	105	75	70
JOB0005/00004	786	105	75	74
JOB0005/00005	444	80	70	76
JOB0005/00006	524	105	75	74
JOB0006/00001	277	105	75	41
JOB0006/00002	326	105	75	54
JOB0006/00003	582	105	75	81
JOB0006/00004	399	105	75	62
JOB0006/00005	558	105	75	73
JOB0006/00006	750	105	75	88

Continued on next page

Table C.11 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0006/00007	367	80	70	91
JOB0006/00008	240	80	70	72
JOB0006/00009	416	80	70	84
JOB0006/00010	384	80	70	84
JOB0006/00011	408	105	75	66
JOB0006/00012	447	120	81	62
JOB0007/00001	104	80	70	76
JOB0007/00002	104	80	70	76
JOB0007/00003	565	105	75	72
JOB0007/00004	105	80	70	74
JOB0007/00005	206	80	70	69
JOB0007/00006	112	80	70	74
JOB0007/00007	303	80	70	76
JOB0007/00008	161	80	70	51
JOB0007/00009	202	80	70	76
JOB0007/00010	105	80	70	76
JOB0007/00011	111	80	70	74
JOB0007/00012	353	80	70	76
JOB0007/00013	232	80	70	76
JOB0007/00014	112	80	70	74
JOB0007/00015	111	80	70	74
JOB0007/00016	155	80	70	91
JOB0007/00017	201	80	70	76
JOB0007/00018	106	80	70	74
JOB0007/00019	112	80	70	74
JOB0007/00020	111	80	70	74
JOB0007/00021	112	80	70	74
JOB0007/00022	112	80	70	74
JOB0007/00023	112	80	70	74
JOB0007/00024	377	105	75	85

Continued on next page

Table C.11 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0007/00025	339	105	75	54
JOB0007/00026	561	105	75	73
JOB0007/00027	373	105	75	73
JOB0007/00028	553	120	81	75
JOB0007/00029	565	105	75	73
JOB0007/00030	583	105	75	73
JOB0007/00031	549	105	75	73
JOB0007/00032	353	80	70	74
JOB0007/00033	150	105	75	61
JOB0007/00034	156	105	75	61
JOB0007/00035	399	105	75	73
JOB0007/00036	321	105	75	73
JOB0007/00037	405	105	75	72
JOB0007/00038	323	105	75	73
JOB0007/00039	392	105	75	72
JOB0007/00040	698	120	81	76
JOB0007/00041	344	105	75	72
JOB0007/00042	366	105	75	71
JOB0007/00043	518	120	81	85
JOB0008/00001	790	120	81	91
JOB0008/00002	914	120	81	91
JOB0008/00003	788	120	81	91
JOB0008/00004	940	120	81	91
JOB0008/00005	341	80	70	74
JOB0008/00006	741	120	81	76

Table C.12: Problem Set #12

Pallet number	Weight	Length	Breadth	Height
JOB0001/00001	599	120	81	74
JOB0001/00002	709	105	75	72
JOB0001/00003	708	105	75	72
JOB0001/00004	341	80	70	89
JOB0001/00005	620	105	75	72
JOB0001/00006	641	105	75	72
JOB0001/00007	563	120	81	67
JOB0001/00008	663	105	75	71
JOB0001/00009	639	105	75	72
JOB0001/00010	522	105	75	71
JOB0001/00011	329	80	70	73
JOB0001/00012	227	80	70	76
JOB0001/00013	291	80	70	76
JOB0001/00014	291	80	70	51
JOB0001/00015	348	80	70	78
JOB0001/00016	506	80	70	76
JOB0001/00017	586	105	75	71
JOB0001/00018	328	80	70	82
JOB0001/00019	518	105	75	71
JOB0001/00020	494	80	70	76
JOB0001/00021	353	80	70	73
JOB0001/00022	234	80	70	76
JOB0001/00023	260	80	70	73
JOB0001/00024	234	80	70	73
JOB0001/00025	331	80	70	73
JOB0001/00026	378	80	70	76
JOB0001/00027	354	80	70	87
JOB0002/00001	279	80	70	85

Continued on next page

Table C.12 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0002/00002	290	80	70	85
JOB0002/00003	325	80	70	84
JOB0002/00004	237	80	70	74
JOB0002/00005	355	80	70	88
JOB0002/00006	350	80	70	96
JOB0002/00007	348	80	70	78
JOB0002/00008	396	80	70	84
JOB0002/00009	331	80	70	79
JOB0002/00010	358	105	75	67
JOB0002/00011	317	80	70	74
JOB0002/00012	398	80	70	85
JOB0002/00013	394	80	70	84
JOB0002/00014	398	105	75	61
JOB0002/00015	399	80	70	84
JOB0002/00016	362	120	81	78
JOB0002/00017	528	105	75	91
JOB0003/00001	679	120	81	90
JOB0003/00002	783	105	75	86
JOB0003/00003	188	80	70	62
JOB0003/00004	742	105	75	71
JOB0003/00005	268	80	70	58
JOB0003/00006	334	80	70	79
JOB0003/00007	656	105	75	85
JOB0003/00008	308	80	70	83
JOB0003/00009	725	105	75	84
JOB0003/00010	726	105	75	84
JOB0003/00011	399	80	70	88
JOB0003/00012	348	80	70	72
JOB0003/00013	348	80	70	72
JOB0003/00014	176	80	70	44

Continued on next page

Table C.12 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0004/00001	435	80	70	92
JOB0004/00002	614	105	75	76
JOB0004/00003	263	80	70	57
JOB0004/00004	317	80	70	78
JOB0004/00005	642	105	75	77
JOB0004/00006	306	80	70	79
JOB0004/00007	502	80	70	92
JOB0005/00001	324	80	70	79
JOB0005/00002	443	80	70	90
JOB0005/00003	433	80	70	82
JOB0005/00004	323	80	70	59
JOB0005/00005	795	120	81	75
JOB0005/00006	573	105	75	82
JOB0005/00007	666	120	81	79
JOB0005/00008	602	105	75	76
JOB0005/00009	620	105	75	68
JOB0005/00010	586	105	75	71
JOB0006/00001	477	80	70	87
JOB0006/00002	341	105	75	47
JOB0006/00003	601	105	75	70
JOB0006/00004	578	105	75	74
JOB0006/00005	442	80	70	94
JOB0006/00006	848	105	75	89
JOB0006/00007	268	80	70	71
JOB0006/00008	336	80	70	76
JOB0006/00009	232	80	70	72
JOB0007/00001	670	105	75	75
JOB0007/00002	281	105	75	66
JOB0007/00003	368	105	75	83
JOB0007/00004	534	105	75	71

Continued on next page

Table C.12 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0007/00005	455	105	75	71
JOB0007/00006	371	80	70	78
JOB0007/00007	280	105	75	66
JOB0007/00008	535	105	75	71
JOB0007/00009	596	105	75	71
JOB0007/00010	595	105	75	71
JOB0007/00011	534	105	75	71
JOB0007/00012	504	105	75	76
JOB0007/00013	536	105	75	71
JOB0007/00014	386	105	75	87
JOB0007/00015	364	80	70	79
JOB0007/00016	201	80	70	46
JOB0008/00001	281	80	70	92
JOB0008/00002	400	80	70	67
JOB0008/00003	778	105	75	71
JOB0008/00004	209	80	70	62
JOB0008/00005	359	80	70	83
JOB0008/00006	339	80	70	84
JOB0008/00007	460	80	70	94
JOB0008/00008	403	105	75	87
JOB0008/00009	311	105	75	71
JOB0009/00001	169	80	70	59
JOB0009/00002	306	80	70	76
JOB0009/00003	875	120	81	76
JOB0010/00001	359	80	70	75
JOB0010/00002	486	80	70	88
JOB0010/00003	630	105	75	80
JOB0010/00004	516	105	75	70
JOB0010/00005	651	105	75	74
JOB0010/00006	464	105	75	74

Continued on next page

Table C.12 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0010/00007	490	105	75	60
JOB0010/00008	379	105	75	57
JOB0010/00009	291	80	70	73
JOB0011/00001	237	80	70	61
JOB0011/00002	352	80	70	72
JOB0011/00003	443	105	75	64
JOB0011/00004	702	105	75	83
JOB0011/00005	476	80	70	96
JOB0011/00006	335	80	70	73
JOB0011/00007	581	105	75	79
JOB0011/00008	391	105	75	65
JOB0011/00009	607	105	75	64
JOB0012/00001	238	80	70	77
JOB0012/00002	412	105	75	69
JOB0012/00003	613	105	75	79
JOB0012/00004	217	80	70	64
JOB0012/00005	198	80	70	59
JOB0012/00006	389	80	70	76
JOB0012/00007	309	80	70	79
JOB0012/00008	358	80	70	66
JOB0012/00009	341	80	70	75
JOB0013/00001	14	80	60	31
JOB0014/00001	36	80	60	29
JOB0015/00001	220	120	81	38
JOB0016/00001	370	80	70	59
JOB0016/00002	880	105	75	92
JOB0016/00003	536	105	75	71
JOB0016/00004	580	120	81	72
JOB0016/00005	812	105	75	70
JOB0016/00006	304	80	70	76

Continued on next page

Table C.12 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0016/00007	800	105	75	70
JOB0016/00008	798	105	75	70
JOB0016/00009	348	80	70	64
JOB0016/00010	288	80	70	63
JOB0017/00001	325	80	70	77
JOB0017/00002	458	80	70	87
JOB0017/00003	937	105	75	85
JOB0017/00004	340	80	70	88
JOB0017/00005	280	105	75	71
JOB0017/00006	229	105	75	51
JOB0017/00007	983	105	75	85
JOB0017/00008	513	105	75	57
JOB0017/00009	709	105	75	70

Table C.13: Problem Set #13

Pallet number	Weight	Length	Breadth	Height
JOB0001/00001	435	80	70	92
JOB0001/00002	614	105	75	76
JOB0001/00003	263	80	70	57
JOB0001/00004	317	80	70	78
JOB0001/00005	642	105	75	77
JOB0001/00006	306	80	70	79
JOB0001/00007	502	80	70	92
JOB0002/00001	324	80	70	79
JOB0002/00002	443	80	70	90
JOB0002/00003	433	80	70	82
JOB0002/00004	323	80	70	59
JOB0002/00005	795	120	81	75
JOB0002/00006	573	105	75	82
JOB0002/00007	666	120	81	79
JOB0002/00008	602	105	75	76
JOB0002/00009	620	105	75	68
JOB0002/00010	586	105	75	71
JOB0003/00001	477	80	70	87
JOB0003/00002	341	105	75	47
JOB0003/00003	601	105	75	70
JOB0003/00004	578	105	75	74
JOB0003/00005	442	80	70	94
JOB0003/00006	848	105	75	89
JOB0003/00007	268	80	70	71
JOB0003/00008	336	80	70	76
JOB0003/00009	232	80	70	72
JOB0004/00001	670	105	75	75
JOB0004/00002	281	105	75	66

Continued on next page

Table C.13 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0004/00003	368	105	75	83
JOB0004/00004	534	105	75	71
JOB0004/00005	455	105	75	71
JOB0004/00006	371	80	70	78
JOB0004/00007	280	105	75	66
JOB0004/00008	535	105	75	71
JOB0004/00009	596	105	75	71
JOB0004/00010	595	105	75	71
JOB0004/00011	534	105	75	71
JOB0004/00012	504	105	75	76
JOB0004/00013	536	105	75	71
JOB0004/00014	386	105	75	87
JOB0004/00015	364	80	70	79
JOB0004/00016	201	80	70	46
JOB0005/00001	281	80	70	92
JOB0005/00002	400	80	70	67
JOB0005/00003	778	105	75	71
JOB0005/00004	209	80	70	62
JOB0005/00005	359	80	70	83
JOB0005/00006	339	80	70	84
JOB0005/00007	460	80	70	94
JOB0005/00008	403	105	75	87
JOB0005/00009	311	105	75	71
JOB0006/00001	169	80	70	59
JOB0006/00002	306	80	70	76
JOB0006/00003	875	120	81	76
JOB0007/00001	359	80	70	75
JOB0007/00002	486	80	70	88
JOB0007/00003	630	105	75	80
JOB0007/00004	516	105	75	70

Continued on next page

Table C.13 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0007/00005	651	105	75	74
JOB0007/00006	464	105	75	74
JOB0007/00007	490	105	75	60
JOB0007/00008	379	105	75	57
JOB0007/00009	291	80	70	73
JOB0008/00001	237	80	70	61
JOB0008/00002	352	80	70	72
JOB0008/00003	443	105	75	64
JOB0008/00004	702	105	75	83
JOB0008/00005	476	80	70	96
JOB0008/00006	335	80	70	73
JOB0008/00007	581	105	75	79
JOB0008/00008	391	105	75	65
JOB0008/00009	607	105	75	64
JOB0009/00001	238	80	70	77
JOB0009/00002	412	105	75	69
JOB0009/00003	613	105	75	79
JOB0009/00004	217	80	70	64
JOB0009/00005	198	80	70	59
JOB0009/00006	389	80	70	76
JOB0009/00007	309	80	70	79
JOB0009/00008	358	80	70	66
JOB0009/00009	341	80	70	75
JOB0010/00001	14	80	60	31
JOB0011/00001	36	80	60	29
JOB0012/00001	220	120	81	38
JOB0013/00001	370	80	70	59
JOB0013/00002	880	105	75	92
JOB0013/00003	536	105	75	71
JOB0013/00004	580	120	81	72

Continued on next page

Table C.13 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0013/00005	812	105	75	70
JOB0013/00006	304	80	70	76
JOB0013/00007	800	105	75	70
JOB0013/00008	798	105	75	70
JOB0013/00009	348	80	70	64
JOB0013/00010	288	80	70	63
JOB0014/00001	325	80	70	77
JOB0014/00002	458	80	70	87
JOB0014/00003	937	105	75	85
JOB0014/00004	340	80	70	88
JOB0014/00005	280	105	75	71
JOB0014/00006	229	105	75	51
JOB0014/00007	983	105	75	85
JOB0014/00008	513	105	75	57
JOB0014/00009	709	105	75	70

Table C.14: Problem Set #14

Pallet number	Weight	Length	Breadth	Height
JOB0001/00001	354	80	70	55
JOB0001/00002	589	120	81	63
JOB0001/00003	589	120	81	63
JOB0001/00004	586	120	81	63
JOB0001/00005	589	120	81	63
JOB0001/00006	590	120	81	63
JOB0001/00007	591	120	81	63
JOB0001/00008	241	80	70	61
JOB0001/00009	312	80	70	75
JOB0001/00010	410	80	70	75
JOB0001/00011	306	80	70	75
JOB0001/00012	898	120	81	75
JOB0001/00013	198	80	70	58
JOB0001/00014	201	80	70	59
JOB0001/00015	923	120	81	75
JOB0002/00001	351	120	81	64
JOB0002/00002	378	80	70	80
JOB0002/00003	398	80	70	75
JOB0002/00004	459	80	70	75
JOB0002/00005	468	80	70	75
JOB0003/00001	371	105	75	68
JOB0003/00002	643	105	75	72
JOB0003/00003	322	80	70	87
JOB0003/00004	327	105	75	75
JOB0003/00005	600	105	75	72
JOB0003/00006	328	80	70	84
JOB0003/00007	212	80	70	77
JOB0003/00008	269	105	75	74

Continued on next page

Table C.14 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0003/00009	310	80	70	87
JOB0003/00010	330	105	75	75
JOB0003/00011	369	80	70	78
JOB0003/00012	578	105	75	73
JOB0003/00013	566	105	75	72
JOB0003/00014	225	80	70	56
JOB0003/00015	331	80	70	73
JOB0003/00016	172	80	70	72
JOB0003/00017	179	80	70	69
JOB0004/00001	212	80	70	62
JOB0004/00002	304	80	70	73
JOB0004/00003	282	80	70	76
JOB0004/00004	310	80	70	75
JOB0004/00005	306	80	70	88
JOB0004/00006	136	80	70	60
JOB0004/00007	522	105	75	69
JOB0004/00008	303	80	70	74
JOB0005/00001	503	105	75	72
JOB0006/00001	179	80	70	79
JOB0006/00002	452	80	70	87
JOB0007/00001	384	105	75	57
JOB0007/00002	563	105	75	67
JOB0008/00001	558	120	81	61
JOB0008/00002	341	80	70	74
JOB0008/00003	296	80	70	74
JOB0009/00001	61	80	70	58
JOB0009/00002	500	80	70	83
JOB0009/00003	318	80	70	73
JOB0009/00004	803	105	75	81
JOB0009/00005	357	105	75	62

Continued on next page

Table C.14 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0009/00006	478	105	75	65
JOB0009/00007	594	105	75	79
JOB0009/00008	421	80	70	80
JOB0009/00009	320	80	70	89
JOB0009/00010	432	80	70	73
JOB0009/00011	204	80	70	56
JOB0009/00012	173	80	70	49
JOB0010/00001	296	80	70	80
JOB0010/00002	341	80	70	72
JOB0010/00003	280	80	70	72
JOB0010/00004	479	80	70	91
JOB0010/00005	577	105	75	72
JOB0010/00006	116	105	75	56
JOB0010/00007	346	80	70	82
JOB0010/00008	500	105	75	72
JOB0010/00009	429	80	70	81
JOB0010/00010	337	80	70	72
JOB0010/00011	347	80	70	72
JOB0010/00012	360	80	70	70
JOB0010/00013	330	80	70	75
JOB0010/00014	431	80	70	73
JOB0011/00001	487	80	70	76
JOB0011/00002	492	80	70	84
JOB0011/00003	383	80	70	74
JOB0011/00004	249	80	70	64
JOB0011/00005	357	80	70	79
JOB0011/00006	323	80	70	81
JOB0011/00007	293	80	70	61
JOB0011/00008	265	80	70	74
JOB0011/00009	515	105	75	72

Continued on next page

Table C.14 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0011/00010	261	105	75	76
JOB0011/00011	629	105	75	72
JOB0011/00012	486	105	75	73
JOB0011/00013	269	80	70	72
JOB0011/00014	167	80	70	76
JOB0011/00015	244	80	70	63
JOB0012/00001	535	105	75	71
JOB0012/00002	252	80	70	64
JOB0012/00003	300	80	70	73
JOB0012/00004	244	80	70	73
JOB0012/00005	400	80	70	72
JOB0012/00006	379	80	70	82
JOB0012/00007	322	80	70	75
JOB0012/00008	370	80	70	61
JOB0012/00009	345	80	70	73
JOB0012/00010	440	80	70	79
JOB0012/00011	440	80	70	81
JOB0012/00012	405	105	75	63
JOB0013/00001	547	105	75	80
JOB0013/00002	273	80	70	65
JOB0013/00003	377	105	75	62
JOB0013/00004	279	80	70	61
JOB0013/00005	278	105	75	63
JOB0013/00006	513	105	75	75
JOB0013/00007	382	105	75	74
JOB0013/00008	175	80	70	64
JOB0013/00009	407	80	70	87
JOB0013/00010	332	80	70	82

Table C.15: Problem Set #15

Pallet number	Weight	Length	Breadth	Height
JOB0001/00001	465	80	70	96
JOB0001/00002	428	80	70	78
JOB0001/00003	512	80	70	100
JOB0001/00004	412	80	70	82
JOB0001/00005	215	80	70	77
JOB0001/00006	386	80	70	74
JOB0001/00007	382	80	70	84
JOB0001/00008	99	80	70	39
JOB0001/00009	329	80	70	74
JOB0001/00010	716	105	75	71
JOB0001/00011	291	80	70	69
JOB0001/00012	220	80	70	60
JOB0001/00013	326	80	70	78
JOB0001/00014	313	80	70	80
JOB0001/00015	599	105	75	71
JOB0001/00016	453	120	81	74
JOB0001/00017	329	105	75	71
JOB0002/00001	326	80	70	74
JOB0002/00002	391	80	70	74
JOB0002/00003	308	80	70	74
JOB0002/00004	620	105	75	71
JOB0002/00005	332	80	70	68
JOB0002/00006	363	80	70	74
JOB0002/00007	361	80	70	82
JOB0002/00008	417	105	75	70
JOB0002/00009	260	105	75	72
JOB0002/00010	225	120	81	60
JOB0002/00011	356	80	70	78

Continued on next page

Table C.15 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0003/00001	470	80	70	76
JOB0003/00002	370	80	70	96
JOB0003/00003	576	105	75	72
JOB0003/00004	490	105	75	72
JOB0003/00005	351	80	70	66
JOB0003/00006	297	80	70	73
JOB0003/00007	267	80	70	75
JOB0003/00008	386	80	70	73
JOB0003/00009	504	80	70	83
JOB0003/00010	399	80	70	100
JOB0003/00011	458	80	70	97
JOB0003/00012	366	105	75	74
JOB0003/00013	411	105	75	72
JOB0003/00014	423	80	70	86
JOB0003/00015	419	80	70	86
JOB0003/00016	377	80	70	83
JOB0003/00017	142	80	70	46
JOB0003/00018	359	80	70	73
JOB0003/00019	359	80	70	79
JOB0003/00020	606	105	75	73
JOB0003/00021	417	80	70	73
JOB0003/00022	423	80	70	73
JOB0003/00023	283	80	70	79
JOB0003/00024	270	105	75	60
JOB0003/00025	639	105	75	73
JOB0003/00026	630	105	75	73
JOB0003/00027	637	105	75	73
JOB0003/00028	632	105	75	73
JOB0003/00029	575	105	75	73
JOB0003/00030	634	105	75	73

Continued on next page

Table C.15 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0003/00031	633	105	75	73
JOB0003/00032	289	80	70	80
JOB0003/00033	567	105	75	72
JOB0004/00001	111	80	70	48
JOB0004/00002	345	80	70	95
JOB0004/00003	325	80	70	84
JOB0004/00004	320	80	70	78
JOB0004/00005	240	80	70	84
JOB0004/00006	266	80	70	76
JOB0004/00007	408	80	70	73
JOB0004/00008	388	80	70	83
JOB0004/00009	266	80	70	79
JOB0004/00010	352	80	70	88
JOB0004/00011	449	105	75	39
JOB0004/00012	882	105	75	65
JOB0004/00013	551	105	75	52
JOB0005/00001	496	120	81	61
JOB0005/00002	824	120	81	76
JOB0005/00003	653	120	81	76
JOB0005/00004	296	80	70	74
JOB0005/00005	325	80	70	74
JOB0005/00006	320	80	70	74
JOB0005/00007	355	80	70	74
JOB0005/00008	657	120	81	76
JOB0005/00009	370	80	70	74
JOB0005/00010	306	80	70	74
JOB0005/00011	388	80	70	74
JOB0005/00012	615	105	75	75
JOB0005/00013	790	120	81	76
JOB0005/00014	659	120	81	76

Continued on next page

Table C.15 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0005/00015	315	80	70	74
JOB0005/00016	579	120	81	76
JOB0005/00017	721	120	81	76
JOB0005/00018	863	120	81	91
JOB0005/00019	612	120	81	76
JOB0005/00020	604	120	81	76
JOB0005/00021	335	105	75	44
JOB0005/00022	830	120	81	76
JOB0005/00023	470	120	81	60
JOB0006/00001	572	80	70	92
JOB0006/00002	664	105	75	60
JOB0006/00003	368	80	70	66
JOB0006/00004	290	80	70	84
JOB0006/00005	226	80	70	74
JOB0006/00006	319	80	70	66
JOB0006/00007	331	80	70	64
JOB0006/00008	624	105	75	71
JOB0006/00009	352	105	75	67
JOB0006/00010	490	105	75	61
JOB0006/00011	376	105	75	46
JOB0006/00012	645	105	75	71
JOB0006/00013	179	80	60	61
JOB0006/00014	550	105	75	74
JOB0006/00015	681	105	75	81
JOB0006/00016	542	105	75	73
JOB0006/00017	375	105	75	64
JOB0007/00001	98	80	70	45
JOB0008/00001	106	80	60	43
JOB0009/00001	53	80	60	30
JOB0010/00001	10	80	60	26

Continued on next page

Table C.15 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0011/00001	326	80	70	79
JOB0011/00002	507	80	70	85
JOB0011/00003	297	80	70	75
JOB0011/00004	325	80	70	85
JOB0011/00005	655	105	75	81
JOB0011/00006	523	105	75	79
JOB0011/00007	774	105	75	93
JOB0011/00008	307	80	70	74
JOB0011/00009	323	80	70	74
JOB0011/00010	467	80	70	94
JOB0011/00011	670	105	75	81
JOB0011/00012	383	80	70	83
JOB0011/00013	416	80	70	84
JOB0011/00014	326	105	75	56
JOB0012/00001	345	80	70	71
JOB0012/00002	359	80	70	75
JOB0012/00003	595	120	81	77
JOB0012/00004	678	120	81	77
JOB0012/00005	332	80	70	76
JOB0012/00006	311	80	70	74
JOB0012/00007	527	120	81	77
JOB0012/00008	313	80	70	75
JOB0012/00009	539	120	81	76
JOB0012/00010	701	120	81	77
JOB0012/00011	532	120	81	62
JOB0012/00012	689	120	81	78
JOB0012/00013	532	120	81	76
JOB0012/00014	541	120	81	76
JOB0012/00015	413	80	70	75
JOB0012/00016	497	120	81	61

Continued on next page

Table C.15 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0012/00017	537	120	81	76
JOB0012/00018	270	80	70	74
JOB0012/00019	650	120	81	76
JOB0012/00020	296	80	70	74
JOB0012/00021	543	120	81	75
JOB0012/00022	456	105	75	72
JOB0012/00023	696	120	81	77
JOB0012/00024	287	80	70	75
JOB0012/00025	459	120	81	62
JOB0012/00026	538	120	81	76
JOB0012/00027	626	120	81	77
JOB0012/00028	538	120	81	76
JOB0012/00029	541	120	81	76
JOB0012/00030	540	120	81	77
JOB0012/00031	537	120	81	77
JOB0013/00001	542	120	81	60
JOB0013/00002	476	80	70	89
JOB0013/00003	830	120	81	75
JOB0013/00004	538	120	81	77
JOB0013/00005	676	120	81	64
JOB0013/00006	669	120	81	64
JOB0013/00007	670	120	81	64
JOB0013/00008	260	80	70	60
JOB0013/00009	453	80	70	89
JOB0013/00010	498	80	70	87
JOB0013/00011	345	80	70	75
JOB0013/00012	353	80	70	75
JOB0013/00013	572	120	81	76
JOB0013/00014	681	120	81	76
JOB0013/00015	359	80	70	75

Continued on next page

Table C.15 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0013/00016	262	80	70	60
JOB0013/00017	511	120	81	62
JOB0013/00018	406	120	81	60
JOB0013/00019	535	120	81	77
JOB0013/00020	108	80	70	43
JOB0013/00021	816	120	81	77
JOB0013/00022	515	120	81	62
JOB0013/00023	523	120	81	77
JOB0013/00024	534	120	81	77
JOB0013/00025	817	120	81	77
JOB0014/00001	273	80	70	89
JOB0014/00002	301	80	70	78
JOB0014/00003	366	80	70	89
JOB0014/00004	216	80	70	71
JOB0014/00005	204	80	70	81
JOB0014/00006	316	80	70	70
JOB0014/00007	522	80	70	80
JOB0014/00008	480	120	81	67
JOB0014/00009	144	80	70	50
JOB0014/00010	574	120	81	65
JOB0014/00011	355	120	81	94
JOB0014/00012	163	120	81	72
JOB0015/00001	530	120	81	66
JOB0015/00002	484	105	75	62
JOB0015/00003	481	105	75	63
JOB0015/00004	481	105	75	64
JOB0015/00005	487	105	75	64
JOB0015/00006	488	105	75	63
JOB0015/00007	130	80	70	40
JOB0016/00001	685	105	75	69

Continued on next page

Table C.15 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0016/00002	224	105	75	72
JOB0016/00003	515	105	75	69
JOB0016/00004	304	105	75	56
JOB0016/00005	293	80	70	72
JOB0016/00006	185	80	70	75
JOB0016/00007	309	80	70	79
JOB0016/00008	334	80	70	78
JOB0016/00009	244	120	81	76
JOB0016/00010	211	120	81	76
JOB0016/00011	509	105	75	75
JOB0016/00012	277	105	75	60
JOB0016/00013	597	105	75	69
JOB0016/00014	311	80	70	83
JOB0016/00015	553	105	75	73
JOB0016/00016	343	105	75	93
JOB0016/00017	235	105	75	67
JOB0017/00001	407	80	70	85
JOB0017/00002	316	80	70	58
JOB0017/00003	437	80	70	68
JOB0017/00004	252	80	70	55
JOB0017/00005	413	80	70	73
JOB0017/00006	421	80	70	73
JOB0017/00007	247	80	70	54
JOB0017/00008	514	120	81	62
JOB0017/00009	455	120	81	62
JOB0017/00010	832	120	81	62
JOB0017/00011	467	80	70	68
JOB0017/00012	432	80	70	68
JOB0017/00013	513	80	70	90
JOB0017/00014	385	80	70	68

Continued on next page

Table C.15 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0018/00001	286	120	81	84
JOB0018/00002	296	105	75	94
JOB0018/00003	303	105	75	94
JOB0018/00004	296	105	75	94
JOB0018/00005	158	80	70	48
JOB0018/00006	298	105	75	94
JOB0018/00007	331	80	70	78
JOB0018/00008	627	105	75	71
JOB0018/00009	473	105	75	56
JOB0018/00010	276	105	75	47
JOB0018/00011	315	80	70	58
JOB0018/00012	358	80	70	72
JOB0018/00013	466	80	70	76
JOB0018/00014	273	80	70	88
JOB0018/00015	458	80	70	85
JOB0018/00016	288	80	70	69
JOB0019/00001	273	80	70	74
JOB0019/00002	389	80	70	72
JOB0019/00003	341	80	70	73
JOB0019/00004	225	80	70	75
JOB0019/00005	399	80	70	86
JOB0019/00006	181	80	70	42
JOB0019/00007	350	80	70	72
JOB0019/00008	256	80	70	76
JOB0019/00009	275	80	70	68
JOB0019/00010	474	105	75	79
JOB0019/00011	479	105	75	79
JOB0020/00001	362	80	70	84
JOB0020/00002	295	80	70	74
JOB0020/00003	491	80	70	95

Continued on next page

Table C.15 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0020/00004	369	80	70	80
JOB0020/00005	231	80	70	75
JOB0020/00006	308	80	70	74
JOB0020/00007	368	80	70	81
JOB0020/00008	212	80	70	71
JOB0020/00009	400	80	70	93
JOB0020/00010	289	80	70	79
JOB0020/00011	600	105	75	82
JOB0020/00012	351	80	70	76
JOB0020/00013	335	80	70	75
JOB0020/00014	304	80	70	75
JOB0020/00015	417	80	70	87
JOB0020/00016	344	80	70	81
JOB0020/00017	454	80	70	79
JOB0020/00018	215	120	81	45
JOB0021/00001	291	80	70	92
JOB0021/00002	278	105	75	76
JOB0021/00003	398	80	70	55
JOB0021/00004	736	105	75	74
JOB0021/00005	383	80	70	73
JOB0021/00006	353	80	70	81
JOB0021/00007	341	80	70	81
JOB0021/00008	333	80	70	83
JOB0021/00009	261	80	70	75
JOB0021/00010	306	80	70	77
JOB0021/00011	467	105	75	72
JOB0021/00012	393	105	75	67
JOB0021/00013	434	105	75	65
JOB0021/00014	390	105	75	72
JOB0021/00015	338	80	70	73

Continued on next page

Table C.15 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0021/00016	553	105	75	71
JOB0021/00017	597	105	75	81
JOB0021/00018	407	80	70	81
JOB0021/00019	401	80	70	73
JOB0021/00020	255	80	70	89
JOB0021/00021	381	80	70	88
JOB0021/00022	335	105	75	74
JOB0021/00023	61	80	60	45
JOB0021/00024	331	105	75	73
JOB0021/00025	600	120	81	78
JOB0022/00001	633	105	75	70
JOB0022/00002	255	80	70	66
JOB0022/00003	274	80	70	74
JOB0022/00004	638	105	75	70
JOB0022/00005	329	80	70	85
JOB0022/00006	466	80	70	93
JOB0022/00007	280	105	75	61
JOB0022/00008	313	80	70	84
JOB0022/00009	384	80	70	74
JOB0022/00010	337	80	70	74
JOB0022/00011	255	80	70	74
JOB0022/00012	157	80	70	74
JOB0022/00013	398	80	70	74
JOB0022/00014	385	80	70	74
JOB0022/00015	326	80	70	85
JOB0022/00016	420	80	70	85
JOB0022/00017	186	80	70	77
JOB0022/00018	210	80	70	77
JOB0022/00019	713	105	75	70
JOB0023/00001	305	80	70	83

Continued on next page

Table C.15 – continued from previous page

Pallet number	Weight	Length	Breadth	Height
JOB0023/00002	350	80	70	91
JOB0023/00003	289	80	70	78
JOB0023/00004	354	80	70	76
JOB0023/00005	257	80	70	77
JOB0023/00006	352	80	70	84
JOB0023/00007	233	80	70	60
JOB0024/00001	83	80	70	30

References

Alcivar, I. and Abad, A. G. (2016). Design and evaluation of a gamified system for erp training. *Computers in Human Behavior*, 58(Supplement C):109 – 118. 129, 130

Alonso, M., Alvarez-Valdes, R., Iori, M., Parreo, F., and Tamarit, J. (2017). Mathematical models for multicontainer loading problems. *Omega*, 66(Part A):106 – 117. 93

Bischoff, E. and Ratcliff, M. (1995a). Issues in the development of approaches to container loading. *Omega*, 23(4):377 – 390. 14

Bischoff, E. E. and Ratcliff, M. S. W. (1995b). Loading multiple pallets. *The Journal of the Operational Research Society*, 46(11):1322–1336. 93

Black, S. E. and Lynch, L. M. (2001). How to compete: The impact of workplace practices and information technology on productivity. *The Review of Economics and Statistics*, 83(3):434–445. 129

Bortfeldt, A. and Gehring, H. (2001). A hybrid genetic algorithm for the container loading problem. *European Journal of Operational Research*, 131(1):143–161. 14, 93

Bortfeldt, A., Gehring, H., and Mack, D. (2003). A parallel tabu search algorithm for solving the container loading problem. *Parallel Computing*, 29(5):641 – 662. Parallel computing in logistics. 14, 15

Bortfeldt, A. and Wäscher, G. (2013). Constraints in container loading a state-of-the-art review. *European Journal of Operational Research*, 229(1):1 – 20. 16, 20, 125

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43. 94

Bruns, F., Knust, S., and Shakhlevich, N. V. (2016). Complexity results for storage loading problems with stacking constraints. *European Journal of Operational Research*, 249(3):1074–1081. 1

Brynjolfsson, E. and Hitt, L. M. (2000). Beyond computation: Information technology, organizational transformation and business performance. *Journal of Economic Perspectives*, 14(4):23–48. 129

Cant, R., Langensiepen, C., and Burton, A. (2012). Entropy as a computational aesthetic measure. *International Journal of Simulation: Systems, Science and Technology*, 13(3A):24–34. 75, 78, 94

Cant, R. J. and Langensiepen, C. S. (2010). Entropy measurement within graphical scenes. In *UKSim2010 - UKSim 12th International Conference on Computer Modelling and Simulation*, pages 474–481. IEEE. 76, 94

Cheng, C.-p., Liu, C.-w., and Liu, C.-c. (2000). Unit commitment by Lagrangian relaxation and genetic algorithms. *IEEE Transactions on Power Systems*, 15(2):707–714. 33

Christensen, S. G. and Rousøe, D. M. (2009). Container loading with multi-drop constraints. *International Transactions in Operational Research*, 16(6):727–743. 123

Costa, M. d. G. and Captivo, M. E. (2016). Weight distribution in container loading: a case study. *International Transactions in Operational Research*, 23(1-2):239–263. 1

Coulom, R. (2007). Efficient selectivity and backup operators in monte-carlo tree search. In *Proceedings of the 5th International Conference on Computers and Games*, CG'06, pages 72–83, Berlin, Heidelberg. Springer-Verlag. 94

Davis, L. (1989). Adapting Operator Probabilities in Genetic Algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 61–69, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 33

De Jong, K. A. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, Ann Arbor, MI, USA. AAI7609381. 33

de Marcos, L., Garcia-Lopez, E., and Garcia-Cabot, A. (2015). On the Effectiveness of Game-like and Social Approaches in Learning: Comparing Educational Gaming, Gamification & Social Networking. *Computers & Education*, 95:99–113. 130

de Queiroz, T. A. and Miyazawa, F. K. (2013). Two-dimensional strip packing problem with load balancing, load bearing and multi-drop constraints. *International Journal of Production Economics*, 145(2):511–530. 123

Debrabander, B. and Edstrom, A. (1977). Successful Information System Development Projects. *Management Science*, 24(2):191–199. 129

Dereli, T. and Sena Das, G. (2010). A Hybrid Simulated Annealing Algorithm for Solving Multi-Objective Container-Loading Problems. *Applied Artificial Intelligence*, 24(5):463–486. 15, 118

Dereli, T. T. and Da, G. S. (2010). Development of a Decision Support System for solving Container Loading Problems. *Transport*, 25(2):138–147. 13

Deterding, S., Nacke, L., Dixon, D., and Khaled, R. (2011a). From Game Design Elements to Gamefulness: Defining "Gamification". *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems - CHI EA '11*, pages 9–15. 126, 129

Deterding, S., Sicart, M., Nacke, L., O'Hara, K., and Dixon, D. (2011b). Gamification. using game-design elements in non-gaming contexts. *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems - CHI EA '11*, pages 2425–2428. 126

Dowsland, K. A. and Dowsland, W. B. (1992). Packing problems. *European Journal of Operational Research*, 56(1):2–14. 13, 38

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159. 13, 20, 23

Eley, M. (2002). Solving container loading problems by block arrangement. *European Journal of Operational Research*, 141(2):393–409. 14, 121

Eley, M. (2003). A bottleneck assignment approach to the multiple container loading problem. *OR Spectrum*, 25(1):45–60. 18

Gehring, H. and Bortfeldt, A. (1997). A genetic algorithm for solving the container loading problem. *International Transactions in Operational Research*, 4(5-6):401–418. 14, 15, 19, 35, 93

Gendreau, M., Iori, M., Laporte, G., and Martello, S. (2006). A Tabu Search Algorithm for a Routing and Container Loading Problem. 93

George, J. and Robinson, D. (1980). A heuristic for packing boxes into a container. *Computers & Operations Research*, 7(3):147–156. 14

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. 45

Groh, F. (2012). Gamification: State of the Art Definition and Utilization. *Research Trends in Media Informatics*, pages 39–46. 140

Hamari, J., Koivisto, J., and Sarsa, H. (2014). Does gamification work? - A literature review of empirical studies on gamification. *Proceedings of the Annual Hawaii International Conference on System Sciences*, pages 3025–3034. 140

Hopgood, A. A. (2001). *Intelligent Systems for Engineers and Scientists (2nd Ed.)*. CRC Press, Inc., Boca Raton, FL, USA. 45

Huang, Y. H., Hwang, F. J., and Lu, H. C. (2016). An effective placement method for the single container loading problem. *Computers and Industrial Engineering*, 97:212–221. 1

Huotari, K. and Hamari, J. (2012). Defining gamification: a service marketing perspective. *Proceeding of the 16th International Academic MindTrek Conference*, pages 17–22. 129

Iori, M. and Riera-Ledesma, J. (2015). Exact algorithms for the double vehicle routing problem with multiple stacks. *Computers & Operations Research*, 63:83–101. 93

Junqueira, L., Morabito, R., and Sato Yamashita, D. (2012a). MIP-based approaches for the container loading problem with multi-drop constraints. 123

Junqueira, L., Morabito, R., and Sato Yamashita, D. (2012b). Three-dimensional container loading models with cargo stability and load bearing constraints. *Computers & Operations Research*, 39(1):74–85. 17

Jylänki, J. (2010). A thousand ways to pack the bin - a practical approach to two-dimensional rectangle bin packing. 53

Kocsis, L. and Szepesvári, C. (2006). *Bandit Based Monte-Carlo Planning*, pages 282–293. Springer Berlin Heidelberg, Berlin, Heidelberg. 94

Le, X. T. and Knust, S. (2017). MIP-based approaches for robust storage loading problems with stacking constraints. *Computers & Operations Research*, 78:138–153. 1

Li, W., Grossman, T., and Fitzmaurice, G. (2012). Gamicad: A gamified tutorial system for first time autocad users. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 103–112, New York, NY, USA. ACM. 130

Li, X., Zhao, Z., and Zhang, K. (2014). A genetic algorithm for the three-dimensional bin packing problem with heterogeneous bins. pages 2039–2048. 14

Li Wang, Hui Zhang, Yan Xiong, and Dawei Li (2010). Ant colony optimization algorithm based on space division for container loading problem. In *2010 Chinese Control and Decision Conference*, pages 3448–3451. IEEE. 15

Lim, A. and Zhang, X. (2005). The container loading problem. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 913–917. ACM. 121

Liu, J., Yue, Y., Dong, Z., Maple, C., and Keech, M. (2011). A novel hybrid tabu search approach to container loading. *Computers & Operations Research*, 38(4):797–807. 15

Lodi, A., Martello, S., and Monaci, M. (2002). Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241–252. 53

Lodi, A., Martello, S., and Vigo, D. (1999). Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing*, 11(4):345–357. 38

Majchrzak, A. and Klein, K. J. (1987). Things are always more complicated than you think: an open systems approach to the organizational effects of computer automated technology. *Journal of Business and Psychology*, 2(1):27–50. 129

Männel, D. and Bortfeldt, A. (2017). Solving the pickup and delivery problem with three-dimensional loading constraints and reloading ban. *European Journal of Operational Research*. 1

Markus Ewald (2009). Nuclex Framework - Documentation (Rectangle Packing). Accessed: 2017-04-24. 53, 66, 70

Markus Ewald (2011a). Nuclex Framework - Dowoload (r1404). Accessed: 2017-04-24. 54, 66

Markus Ewald (2011b). Nuclex Framework - Home. Accessed: 2017-04-24. 70

Martello, S. and Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., New York, NY, USA. 34

McDaniel, R., Lindgren, R., and Friskics, J. (2012). Using badges for shaping interactions in online learning environments. *IEEE International Professional Communication Conference*, pages 12–15. 130

Morabito, R. and Arenalest, M. (1994). An AND/OR-graph Approach to the Container Loading Problem. *International Transactions in Operational Research*, 1(1):59–73. 15

Mostaghimi, H., St Amour, B., and Abdul-Kader, W. (2017). Three-Dimensional Container Loading: A Simulated Annealing Approach. *International Journal of Applied Engineering Research ISSN*, 12(7):973–4562. 1

Moura, A. and Bortfeldt, A. (2016). A two-stage packing problem procedure. *International Transactions in Operational Research*, 24(1–2):43–58. 1, 93, 94

Moura, A. and Oliveira, J. (2005). A GRASP approach to the container-loading problem. *IEEE Intelligent Systems*, 20. 14

Moura, A. and Oliveira, J. F. (2008). An integrated approach to the vehicle routing and container loading problems. *OR Spectrum*, 31:775–800. 94

Nepomuceno, N., Pinheiro, P., and Coelho, A. L. V. (2007). Tackling the container loading problem: a hybrid approach based on integer linear programming and genetic algorithms. In Cotta, C. and Hemert, J., editors, *Evolutionary Computation in Combinatorial Optimization*, volume 4446 of *Lecture Notes in Computer Science*, pages 154–165. Springer-Verlag Berlin Heidelberg, Heidelberger Platz 3, D-14197 Berlin, Germany. 13, 15

Palmer, C. C. and Kershenbaum, A. (1995). An approach to a problem in network design using genetic algorithms. *Networks*, 26(3):151–163. 33

Peng, Y., Zhang, D., and Chin, F. Y. L. (2009). A Hybrid Simulated Annealing Algorithm for Container Loading Problem. pages 919–922, 1515 Broadway, New York, NY 10036-9998 USA. Assoc Computing Machinery. 15, 93

Pires de Araujo, L. J. and Pinheiro, P. (2010). Combining Heuristics Backtracking and Genetic Algorithm to Solve the Container Loading Problem with Weight Distribution. volume 73 of *Advances in Intelligent and Soft Computing*, pages 95–102, Heidelberger Platz 3, D-14197 Berlin, Germany. Springer-Verlag Berlin. 15

Pisinger, D. (1995). *Algorithms for Knapsack Problems*. PhD thesis, University of Copenhagen. 12, 33, 34

Ramos, A. G., Oliveira, J. F., Gonçalves, J. F., and Lopes, M. P. (2016a). A container loading algorithm with static mechanical equilibrium stability constraints. *Transportation Research Part B: Methodological*, 91(Supplement C):565 – 581. 1

Ramos, A. G., Oliveira, J. F., and Lopes, M. P. (2016b). A physical packing sequence algorithm for the container loading problem with static mechanical equilibrium conditions. *International Transactions in Operational Research*, 23(1-2):215–238. 1

Reeves, C. R. (1997). Genetic algorithms for the operations researcher. *INFORMS Journal on Computing*, 9(3):231–250. 33

Remi-Omosowon, A., Cant, R., and Langensiepen, C. (2015). Deriving an Entropy Measure for 2D Container Layouts. In *IEEE UKSim-AMSS 17th International Conference on Computer Modelling and Simulation, UKSim2015 (UKSim2015)*, pages 103–108, Cambridge, United Kingdom. 4

Remi-Omosowon, A., Cant, R., and Langensiepen, C. (2016). Applying Gamification Principles to a Container Loading System in a Warehouse Environment. In *IEEE UKSim-AMSS 18th International Conference on Computer Modelling and Simulation, UKSim2016 (UKSim2016)*, Cambridge, United Kingdom. 5

Rudolph, G. (1994). Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1):96–101. 33

Scheithauer, G. (1992). Algorithms for the container loading problem. In *Operations Research Proceedings 1991*, volume 1991 of *ORP*, pages 445–452. Springer-Verlag, Berlin, Heidelberg. 12

Shannon, C. E. (1948). The mathematical theory of communication. *MD computing computers in medical practice*, 14(4):306–17. 75

Sheng, L., Xiuqin, S., Changjian, C., Hongxia, Z., Dayong, S., Feiyue, W., Changjia, C., Hongxia, Z., Dayong, S., and Feiyue, W. (2017). Heuristic Algorithm for the Container Loading Problem with Multiple Constraints. *Computers & Industrial Engineering*, 108:149–164. 1

Sridhar, R., Chandrasekaran, M., Sriramya, C., and Page, T. (2017). Optimization of heterogeneous Bin packing using adaptive genetic algorithm. *IOP Conference Series: Materials Science and Engineering*, 183(1):12026. 1

Sy, S., Zichermann, G., and Cunningham, C. (2011). *Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps*. Number 10. O'Reilly Media, Inc. 129

Tang, C. H. (2011). A scenario decomposition-genetic algorithm method for solving stochastic air cargo container loading problems. *Transportation Research Part E: Logistics and Transportation Review*, 47(4):520–531. 33

Tarantilis, C., Zachariadis, E., and Kiranoudis, C. (2009). A Hybrid Metaheuristic Algorithm for the Integrated Vehicle Routing and Three-Dimensional Container-Loading Problem. *IEEE Transactions on Intelligent Transportation Systems*, 10. 19

Techanitisawad, A. and Tangwiwatwong, P. (2004). A GA-based Heuristic for the Interrelated Container Selection Loading Problems. 93

Thierens, D. and Goldberg, D. (1994). Elitist recombination: an integrated selection recombination GA. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 508–512 vol.1. 33

Wäscher, G., Haußner, H., and Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130. 2, 13, 14, 20, 23

Wei, L., Oon, W.-C., Zhu, W., and Lim, A. (2011). A skyline heuristic for the 2d rectangular packing and strip packing problems. *European Journal of Operational Research*, 215(2):337 – 346. 99

Werbach, K. and Hunter, D. (2012). *For the Win: How Game Thinking Can Revolutionize Your Business*. Wharton Digital Press (October 30, 2012). 129

Yap, C. N., Lee, L. S., Majid, Z. A., and Seow, H. V. (2012). Ant Colony Optimization for Container Loading Problem. *Journal of Mathematics and Statistics*, 8(2):169–175. 15

Zhao, X., Bennell, J. A., Bekta, T., and Dowsland, K. (2014). A comparative review of 3D container loading algorithms. *International Transactions in Operational Research*, pages 287–320. 20

Zheng, J. N., Chien, C. F., and Gen, M. (2015). Multi-objective multi-population biased random-key genetic algorithm for the 3-D container loading problem. *Computers and Industrial Engineering*, 89:80–87. 14

Zhu, W. and Lim, A. (2012). A new iterative-doubling Greedy-Lookahead algorithm for the single container loading problem. *European Journal of Operational Research*, 222(3):408–417. 14

Zichermann, G. (2011). Designing Gamification Level 1. *Udemy*, https://www.udemy.com/designing-gamification-level-1-certification/. 123