Heuristics Approaches For Three-Dimensional Strip Packing And Multiple Carrier Transportation Plans

Ha Thai Duong, BSc.

Thesis submitted to The University of Nottingham

for the degree of Doctor of Philosophy

December 2015

Abstract

In transport logistic operations, an efficient delivery plan and better utilisation of vehicles will result in fuel cost savings, reduced working hours and even reduction of carbon dioxide emissions. This thesis proposes various algorithmic approaches to generate improved performance in automated vehicle load packing and route planning. First, modifications to best-fit heuristic methodologies are proposed and then incorporated into a simple but effective "look-ahead" heuristic procedure. The results obtained are very competitive and in some cases best-known results are found for different sets of constraints on three-dimensional strip packing problems. Secondly, a review and comparison of different clustering techniques in transport route planning is presented. This study shows that the algorithmic approach performs according to the specific type of real-world transport route planning scenario under consideration. This study helps to achieve a better understanding of how to conduct the automated generation of vehicle routes that meet the specific conditions required in the operations of a transport logistics company. Finally, a new approach to measuring the quality of transportation route plans is presented showing how this procedure has a positive effect on the quality of the generated route plans. In summary, this thesis proposes new tailored and effective heuristic methodologies that have been tested and incorporated into the real-world operations of a transport logistics company. The research work presented here is a modest yet significant advance to better understanding and solving the difficult problems of vehicle loading and routing in real-world scenarios.

Acknowledgments

Firstly, I would like to thank my supervisor Dr. Dario Landa-Silva for all his support and encouragement throughout the final year of my PhD. Also, I would like to thank Prof. Edmund Burke and Prof. Graham Kendall who gave support and advice during my first and second year. I also thank the School of Computer Science and the Graduate School, both at the University of Nottingham, for providing excellent facilities for my research. This doctoral programme would have not been possible without the financial support of the Engineering and Physical Sciences Research Council (EPSRC), the School of Computer Science, The University of Nottingham, and the Vietnamese Education and Training Department. I would like to thank my parents and the rest of my family for their help, support and even sacrifices from the start until the end. Hopefully, there will be a happy ending because they deserve it.

Contents

List of Tables

List of Figures

1 Introduction

This chapter provides an introduction to the research presented in this PhD thesis. Firstly, the background and motivation for investigating the three-dimensional packing problem and the single-customer multiple-carrier transport planning problem are outlined. The scope and objectives of this work are described next. Then, the contributions to knowledge arising from this PhD thesis are listed. Finally, an outline of the remaining chapters in this work is presented.

1.1 Background and Motivation

Transport logistics have always been a very important factor in many industrial and business scenarios. The provision of logistic services is also an extremely large and competitive market. For example, a 2011 report by the Department for Transport Road Freight Statistics stated that by the end of 2010 there were nearly 400,000 vehicles of over 3.5 tonnes operating in Great Britain with a turn-over of about £24 billion and 30,149 enterprises in road transportation (DFT (2011)). Even with the significant amount of resources and investment that are dedicated to transport logistics, maximising the use of the current infrastructure still requires extensive research. For example, according to the Barclays Corporate report in 2008, around 29% of Heavy Goods Vehicles (HGVs) were running empty on the road and this figure has not changed significantly according to a similar report in 2012 (Team (2012)). With the recent economic changes and increasing concerns regarding issues such as fuel price, efficiency and environmental impact, the

need for optimisation in transport logistics is more crutical than ever. Important aspects to consider in the optimisation of transport logistics operations include optimising vehicle's space utilisation, as well as more efficient transport planning through the routing of vehicles. Therefore, research and development into an automated cutting or packing method and transport planning tool could bring significant cost savings, improve time efficiency and reduce environmental impact of transport operations. During the research period of this PhD, a collaboration between 3T Logistics Ltd and The University of Nottingham was established to develop and improve an automated planning system for the 4PL logistics model presented in Landa-Silva et al. (2011). This offered an excellent opportunity to receive insightful feedback regarding practical elements of vehicle utilisation and transport planning.

1.2 Objectives and Scope

The main research focus of this thesis is heuristics and its application to the transport logistics market. The first objective was to improve vehicle utilisation by means of a more efficient three-dimensional packing methods. There are many different heuristic approaches proposed in the literature. The best-fit heuristic technique presented in Burke et al. (2004) and Allen et al. (2011) are shown to be very effective when applied on its own or as part of some meta-heuristics. One aim of this PhD project was to improve the best-fit heuristic not only on benchmark packing problems but also, and more importantly for the scope of this thesis, on problems arising in real-world transport logistics scenarios. In addition to the problem definition given in previous works cited above, the stability constraint was also included in the present work. A three-dimensional packing problem in a real-world scenario was also investigated, heuristics designed and performance evaluated using a variety of scenarios. Another aim of this PhD project was to improve the algorithmic approach to transport planning originally developed through a collaboration between The University of Nottingham and 3T Logistics Ltd. This required updating

the problem description and modelling to meet the current business requirements and incorporating additional requirements from live operations. The existing approach was extended to address a number of changes in live operations and improve the performance of the automated planning approach.

1.3 Contributions:

The following is a list of the major contributions of this PhD thesis:

- An extension of the best-fit heuristic for the three-dimensional strip packing problem is introduced. The extension includes two block generation variations, block reallocation and candidate point generation.
- An overhead estimation approach to improve the heuristics result for the three-dimensional strip packing problem is introduced. It produces good results over a set of benchmark data sets.
- An improved heuristic with overhead estimation for the three-dimensional strip packing problem with stability constraint is developed. The proposed approach achieves competitive results when compared to other approaches from the literature.
- Arising from live transport logistic operational scenarios, a new problem pallet space equivalent is presented. This is a real-world problem that incorporates stability and stability constraint.
- A best-fit heuristic is proposed for a pallet space equivalent problem and positive results are obtained through computational experiments. The proposed heuristic has been incorporated into a real-world automated planning system and used in live operations.

- A new quality factor for transport planning is identified to measure inefficient mileage of a route. A tailored meta-heuristic operator was designed to target this new factor in order to reduce inefficient mileage.

- A investigation into the compatibility of different clustering algorithms with the algorithmic approaches for various real-world transport planning profiles was conducted. As a result, better understanding of the influence of the clustering on the overall approach was also achieved.

1.4 Thesis Outline

This PhD thesis contains seven chapters. The first chapter introduces the background, objectives and overview of the thesis. Chapter 2 describes the three-dimensional packing problem and the single-customer multiple-carrier planning problem investigated in this thesis. The definition and benchmark data sets used in the literature for these problems are presented in that chapter too, as well as an overview of the different approaches in the literature to solve related problems. Chapters 3, 4 and 5 address the packing problems. Chapter 3 presents an extension to best-fit heuristics for the three-dimensional strip packing problem. Also, different block generation approaches, procedures for block reallocation and possible block generation are introduced to enhance the used heuristics. That chapter also presents an overhead approach to improving heuristics as an alternative to meta-heuristics. Following from chapter 3, chapter 4 focuses on the three-dimensional strip packing problem with the addition of stability constraint. A variety of heuristics, including best-fit and best support heuristics, are evaluated for combination with overhead estimation approaches. Modifications to heuristics for compatibility with stability constraint are also presented. Experimental results show that the approaches developed here are very competitive when compared to other approaches found in the literature. Chapter 5 establishes a variation on the three-dimensional strip packing problem with

stability and stackability constraints arising in real-world transport planning operations. A pallet space equivalent problem is presented and a constructive heuristic approach was developed. Chapter 6 focuses on a real-world problem in live transport planning operations. A new factor in transport planning was identified and a study on the effect of different clustering approaches using a variety of different scenarios is presented. The approaches from both chapters 5 and 6 have been implemented on live operations and their performance evaluated highly by the business management team hence they were incorporated into the company's live transport planning operations.

2 Background and Related Work

2.1 Brief Note on Computational Complexity

A computational problem belongs to class P, this means it can be solved in polynomial time by a deterministic Turing Machine. A problem belongs to class NP when it can be solved in polynomial time by a non-deterministic Turing Machine. Then, problems in P are those that can be solved in polynomial time by some deterministic algorithm (i.e. solved efficiently) while problems in NP are those that can be solved in polynomial time by a non-deterministic algorithm. It is not known if P = NP and this perhaps the most important open question in computational complexity. For many problems proven to be in NP no efficient algorithm has been found, strengthening the belief that P $\neq$ NP but this conjecture is still not proven (Cormen et al. 2001). There is a class of problems in NP called the NP-complete class and these are considered the hardest problems to solve in this class. A problem is in the NP-complete class if there is a polynomial reduction that can be used to transform that problem into any other problem in this class. NP-hard problems can be described as those problems that are *at least as hard as the hardest problems in the NP class* and therefore it is believed that no efficient algorithm exists for solving these problems unless P = NP. Some of the NP-hard problems have not yet been proven to be in NP. Therefore, it is believed that when tackling a problem that is NP-hard or NP-complete, the focus should not be in finding an efficient algorithm (it is believed that such an algorithm does not exist) but instead on designing algorithms that produce high-quality solutions in practical time. Figure 2.1 shows a Euler diagram for

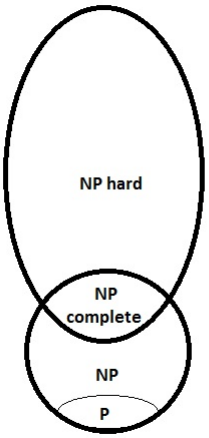


Figure 2.1: Euler diagram for different classes of complexity

these different classes of complexity.

Since solving NP-hard problems with exact algorithms is not efficient in terms of computational time, non-exact solving methods such as heuristics, meta-heuristics and recently hyper-heuristics have received more attention and their potential revealed. These approaches do not offer a guarantee of finding optimal solutions. However, it is possible to produce high-quality solutions in reasonable computational time.

2.2 Three-Dimensional Strip Packing Problems (3D-SPP)

One of the problems investigated in this PhD thesis within the context of freight transport operations is the *three-dimensional strip packing problem (3D-SPP)* for which some tailored heuristics have been developed. In the 3D-SPP we are given a container and a set of rectangle boxes. The problem is to pack all the boxes inside the container in the most efficient way. The container has fixed width and height but the length can be extended as needed. Each of the boxes has fixed given dimensions for width, height and length. The goal when solving the 3D-SPP is to minimise the length of the container required to pack all the boxes. There are different constraints that arise in the 3D-SPP which impose

additional restrictions on how the boxes can be packed into the container, e.g. support constraint, stability constraint etc. The 3D-SPP is a combinatorial optimisation problem (Papadimitriou & Steiglitz 1982) which is also NP-hard (Hopper & Turton 2001).

The 3D-SPP is classified as a 3/B/O following the classification proposed by Dyckhoff (1990). Wäscher et al. (2007) classify this problem as a three-dimensional rectangular open dimension problem with one variable dimension (3D-R-ODP). The 3D-SPP can be defined as follows:

- Input:
  - A set of rectangular boxes with given fixed dimensions for width, height and length.
  - A container with given fixed width and height, length can be extended as needed in order to pack all the boxes.
- Output:
  - A packing plan showing the position of each rectangular box within the container.
- Objective:
  - Minimise the length of the total packing or the length of the container used.
- Constraints:
  - All boxes must be packed.
  - All boxes must be packed fully inside the container.
  - All boxes must be placed orthogonally (i.e. the edges of the boxes should be parallel to the edges of the container)
  - Boxes cannot overlap.

One key difference between the 3D-SPP and the related three-dimensional container packing problem (3D-CPP) is that in container packing, all three dimensions of the container are pre-defined and fixed. Another difference is that in the 3D-SPP all the boxes must be packed but this is not always the case in a solution to a 3D-CPP. As mentioned above, the original objective in 3D-SPP is to minimise the required length of the container needed for packing all boxes. However, in order to compare different packing methods for 3D-SPP's instances, measuring only the container length is not sufficient. For example, consider an instance with one box, the width and height of the box are the same as the container and the length of the box is 1. Obviously, this instance has an optimal solution of length 1. Similarly, an instance with a box of length 100 (same width and height as the container) will have an optimal solution of 100. The two solutions for these two different instances have different lengths but both are optimal. When comparing the performance of packing methods for a collection of instances, the used length is not always the true reflection of the method's efficency. Therefore, alternative measurements are used to assess the quality of the packing. The optimal length is defined as the length of the container required to accommodate the volume of all the boxes. That is, if we could melt all the boxes into liquid form and then pour the liquid into the container of fixed width and height but expandable length, then the length of the container required to hold all the liquid is called the *optimal length*. The optimal length is calculated as the total volume of all boxes divided by the surface area of the container (width multiplied by height). The formula to calculate the *optimal length* is as follows:

$optimalLength = [\sum b.Volume/C.Width * C.Height]$

where $b.Volume$ is the volume of all input boxes, and the width and height of the container are given by $C.Width$ and $C.Height$ respectively.

The utilisation of the length of the container is calculated by:

$utilisation = optimalLength/actualLength$

The higher the utilisation achieved, the shorter the *optimal length* required for packing

all boxes in a 3D-SPP instance.

Strip packing problems have many practical applications. In the case of 2D-SPP applications include material cutting, pallet loading and process scheduling while 3D-SPP applications include carrier load building, container design and resource allocation (Coffman et al. 1978). In the literature, the two-dimensional strip packing problem has received considerable attention from researchers, but research on the three-dimensional problem is limited in comparison. The 3D-SPP can be seen as a generalisation of the 2D-SPP, therefore we can solve the two-dimensional case using a method for 3D-SPP by assuming that the height of each box and the container is 1.

Bischoff and Ratcliff (1995) introduced many practical requirements for cutting and packing problems. A number of practical constraints include: orientation constraints, handling constraints, load stability, grouping of items, multi-drop situations, separation of items within a container, complete shipment of certain item groups, shipment priorities, complexity of the loading arrangement, container weight limit and weight distribution within a container. In the research work described in this thesis, rotation constraints and stability constraints are taken into account.

Approaches to tackling 3D-SPP include: heuristics and other approximation algorithms, as well as meta-heuristics and hyper-heuristics. The following subsections review some of these methods and briefly describe some of the most relevant ones for the work developed in this thesis. For a more detailed discussion of these and other search methodologies and optimisation techniques, please refer to Burke & Kendall (2005).

2.2.1 Heuristics and Approximation Algorithms

A heuristic can be described as a *"rule of thumb"* approach. Based on experience and knowledge about the problem at hand, such an approach can be developed to produce a solution. A heuristic approach can generate reasonably good solutions within a short computational time and moderate memory requirements. These advantages tend to

make heuristics a good choice in practice where optimality is not essential. However, since there is no guarantee about the quality of solution achieved, it is of course possible for a heuristic to produce poor or even infeasible solutions. Approximation algorithms also offer good solutions in practice and they also offer some guarantee for the quality of the solution by providing a bound for the worst case solution. In general, approximation algorithms are preferred over heuristics because of this quality assurance. However, to the best of our knowledge, for cutting and packing problems in general, and particularly for the 3D-SPP, state-of-the-art approximation algorithms only offer solutions with lower bounds that are very close to the quality of solutions found by simple or sometimes even inefficient packing algorithms. Therefore, heuristic approaches have received much more attention in 3D-SPP research.

The Next Fit Decreasing Height (NFDH) approach was proposed by Li & Cheng (1990) to tackle the 3D-SPP. NFDH first sorts boxes in decreasing order of height. Then, one box is packed at a time with the next box in the order packed to form horizontal strips. The next strip is packed on top of the previous strip to form layers of boxes. The process continues until all boxes have been packed. Li & Cheng (1992) improved their previous approach and called it LLm. The input boxes are sorted in non-increasing order of height and then they are split into subsets. The total bottom area of each subset has to statisfy a range of values. Each subset is packed into the container by a two-dimensional subroutine. Subsequent approximation algorithms, which concentrate on performance guarantee, have been introduced such as Miyazawa & Wakabayashi (2007), Miyazawa & Wakabayashi (2009), Jansen & Solis-Oba (2006) and Bansal et al. (2007).

Many of the heuristics that have been proposed for 3D-SPP are adaptations of approaches for the 2D-SPP. Most of the heuristics for 3D-SPP are constructive algorithms. A constructive algorithm starts with an empty container and at each iteration, a box or a group of boxes is packed into the container until there are no boxes left or a given termination criterion is reached. Usually, the selection of the next box to pack and where to

pack it in the container, are decisions based on experience or knowledge of the problem. Some of the constructive heuristics proposed in the literature are described next.

Baker et al. (1980) proposed the bottom left (BL) heuristic where input boxes are sorted according to their bottom area. Boxes are packed by placing them at the top right part of the container then falling down to the bottom of the container and then moving to the left as far as possible. Chazelle (1983) improved the BL heuristic by placing the box at the bottom most position then moving it to the left as far as possible, and called this approach bottom left most fill (BLF) heuristic. Hopper (2000) used different criteria to sort the sequence of boxes before placing them into the container. Then, the best result of all the criteria is selected for actually placing the boxes. The DBLF (deepest bottom left fill) heuristic chooses the deepest position in the container and then moves to the bottom or lowest position to finally move to the left as much as possible. The best-fit heuristic (BF) was introduced by Burke et al. (2004). This BF procedure considers the bottom most place in the container as candidate position and then considers the box or shape that is a best-fit for that candidate position. If there is no shape that fits into the current candidate position, the next candidate position (possible higher position in the container) is considered. Karabulut & İnceoğlu (2005) proposed the deepest bottom left fill (DBLF) for 3D-SPP based on the BLF heuristic for 2D-SPP (outlined above). Allen et al. (2011) introduced a three-dimensional best-fit heuristic (3BF) for the 3D-SPP, based on the BF heuristic for the 2D-SPP. This heuristic is a constructive approach that finds the boxes that fit the best in the remaining gaps of the container. The *gap* is defined as a free-area on the surface parallel to the deepest surface of the container. To cater for the situation where there might be more than one box that fits in a gap, four different criteria were proposed to break ties. Similar to the process followed in the 2D-BF, if the deepest gap cannot be filled, the next deepest gap is considered. This process continues until there are no boxes left to pack. Other different constructive heuristics have been proposed in the literature. George & Robinson (1980) proposed a layer approach where

the container is divided into vertical layers. Each layer is then divided into horizontal strips and each strip is filled by a row of boxes. Bischoff & Marriott (1990) combined the 2D heuristic from George & Robinson (1980) to fill the layer of boxes. Bortfeldt (1999) proposed approaches based on algorithms for container loading algorithms.

As stated by Bortfeldt (1999), the first algorithms for tackling the container loading problem were proposed in Bortfeldt & Gehring (1998) and Bortfeldt & Gehring (2001). In order to solve the strip packing problem using an algorithm for the container loading problem, there are two approaches: open container and closed container. In the open container approach, the algorithm solves a container loading problem considering an unlimited length for the container. In the closed container approach, the container is given a certain length large enough for the problem in hand. After each successful feasible packing is achieved, the length of the container is reduced and the packing repeated. This process of finding a packing for a container with smaller length is repeated until there is no feasible packing found. It is important to note that the stability constraint is included in Bortfeldt (1999). In Bortfeldt & Mack (2007), the container loading algorithm by Pisinger (2002) was adapted to solve strip packing problems using both open and closed container approaches. Recursive tree searches are carried out to determine the layer depth and strip height and weight. In both Bortfeldt (1999) and Bortfeldt & Mack (2007), the closed container approach produces superior results when compared to the open container approach. A property that layer building approaches have is that the quality of the packing depends on the quality of the layer depth selection.

2.2.2 Meta-heuristics

As mentioned in subsection 2.2.1, constructive heuristic approaches for strip packing problems normally consist of a sequence of boxes being packed into the container or a sequence of boxes that define the layer depth. It is then very useful to apply meta-heuristics to improve the result provided by the constructive heuristics. In general, the

term *"meta-heuristic"* describes a technique that seeks to generate better candidate solutions from the current solution. It is generally accepted that meta-heuristics are search techniques that can be applied without much knowledge of the optimisation problem domain. A number of well-established meta-heuristics are outlined next.

Hill Climbing

Hill climbing is one of the simplest meta-heuristic algorithms. It takes the current solution and makes a modification to it in order to create a new candidate solution. The most common modification involves simple local moves like swapping or changing an item in the current solution. The modification also involves some degree of randomness to increase the explorative degree of the search. If the new solution is better than the current solution then the current solution is replaced by the new solution to then continue with the next iteration of the search. This process is repeated until the algorithm reaches some termination condition such as fixed computation time or no further improvement to the current solution for a number of iterations. It is possible that hill climbing generates more than one new candidate solution from the current one, these are usually called neighbourhood solutions and typically the best neighbour is selected. Given that hill climbing explores solutions that are in the neighbourhood of the current solution, it is common that hill climbing gets stuck in local optima, i.e. best solutions in the current neighbourhood. However, a local optimum solution might not be the best global solution. There are different methods to escape local optima. One common approach is to generate a random solution and use this one as the current solution in order to then explore a different part of the search space. For this, new solutions generated randomly should not be reachable from previous solutions by the local moves used in the algorithm to avoid returning to previously visited solutions.

Simulated Annealing

Simulated annealing (SA) was introduced by Kirkpatrick et al. (1983) and can be considered as an extension of hill climbing but with the probability of accepting some worse

solutions. It took inspiration from the annealing process in metal production whereby metal is heated and cooled a number of times to form a better structure and reduce defects. SA as a search algorithm starts from a "high temperature" and goes through a "cooling period" in which, after each iteration, the temperature is reduced gradually. At each iteration, if a new candidate solution is worse than the current solution, there is still a chance, based on some probability calculated using the current temperature, to take the new candidate solution as the current solution. The higher the temperature the higher the chance to accept a worse candidate solution to become the current solution. It is also possible for the temperature to be increased again, i.e. re-heating the search, after a period of cooling. The standard acceptance criteria function in SA is $P(\triangle d, T)$ where $\triangle d$ is the difference in fitness between the new candidate solution and the current solution, T is the temperature which normally changes with the search time, the longer the time the lower the value of T (unless re-heating takes place).

Tabu Search

Tabu search was first introduced by Glover & McMillan (1986) and is also a kind of hill climbing meta-heuristic but incorporates memory. Tabu search maintains a fixed length tabu list to escape the local optima. The tabu list can contain previously found solutions, or solution's attributes, or modifications of solution that are avoided. Tabu search aims to prevent visiting already seen solutions by constantly updating the tabu list information. At each iteration, tabu search generates a number of neighbour solutions and selects one that is not in the tabu list. It is possible that the best of all neighbour solutions is considered even if it is worse than the current solution. This is to avoid staying in the current local optimum. Tabu search is often considered as hill climbing with fixed size memory.

Genetic Algorithms

Genetic algorithms (GA) are inspired by natural evolution. An introduction to GA can be found in Goldberg & Holland (1988), however the application of GA can be traced

back to much earlier reports by Fraser (1960) and Bremermann (1962). In GA, a solution is encoded in the form of a chromosome. The GA starts with an initial process where a set of solutions is generated. This process normally involves some random generation of solutions. GA use a fitness function to evaluate the quality of each chromosome or solution. From the initialisation, GA maintain a selection of solutions called "population" and evolves this population to obtain better solutions. The evolution process starts with the selection procedure where a number of chromosomes from the current population are selected, normally good quality chromosomes are preferred. Genetic operators such as crossover and mutation are used to generate new chromosomes which are called offspring. The new offspring are evaluated and some of the best offspring will be selected (survive) to form the population in the next generation. The evolution process continues until the termination criteria are reached. Common termination criteria are a specified number of iterations (generations) or a condition is found. Other factors involved in GA are, for example, the selection procedure, the parameter values (such as population size), crossover and mutation probabilities, etc.

The application of meta-heuristics to the 3D-SPP has been reported in many papers in the literature. Bortfeldt (1999) proposed two meta-heuristics for the 3D-SPP, one was tabu search and the other one was a genetic algorithm or GA. In that work, a parallel implementation was also used where multiple settings of meta-heuristics worked in collaboration. Allen et al. (2011) integrated tabu search with the best-fit (BF) heuristic to overcome the difficulty at the end of the packing where the sequence of boxes to pack is critical for the quality of the final result. The initial part of the packing is completed using 3BF until a certain number of boxes are left to be packed. Then, tabu search is used to generate packing sequences for the remaining boxes. Each box is packed in the container using the deepest bottom left fill placement strategy. It is also worth noting that for packing and cutting problems, solutions are represented by sequences of boxes and then it becomes difficult to apply genetic algorithms or other evolutionary algorithms.

This is because the genetic operators for generating offspring are very likely to violate the hard constraints in the 3D-SPP. For example, in two different solutions a box can be packed at the beginning in one solution and at the end in another solution. Then, a crossover of these solutions might create two offspring, one with the same box twice, at the beginning and at the end of the solution, and the other offspring not containing the box at all. This would clearly violate one of the key constraints in the 3D-SPP hence repair operators would be needed.

2.2.3 Hyper-Heuristics

A recent research direction that has been explored for tackling optimisation problems are the so-called *"hyper-heuristics"* which can be described as "heuristics to choose heuristics" (Burke et al. 2003). The idea is to automate the design of heuristics which normally requires human experience or knowledge of the problem domain. Designing a very efficient heuristic or meta-heuristic is likely to involve high cost and a long development time. A heuristic can be very effective in tackling a certain set of constraints but could have limited effect when other constraints are introduced. Moreover, there is a demand for reasonably good solutions to be generated in a reasonable amount of time. Therefore, a hyper-heuristic framework is based on trying to automate the learning process by combining heuristics or the generation of heuristics. This "hyper-heuristics" approach can be classified as the automated combination of heuristics or the automated generation of heuristics (Burke et al. 2010).

Hyper-heuristics have had success in solving other optimisation problems such as production scheduling (Tay & Ho 2008, Vázquez-Rodríguez & Petrovic 2010), educational scheduling (Burke et al. 2006) and vehicle routing problems (Garrido & Riff 2010, Garrido & Castro 2009) among others. For cutting and packing related problems, hyper-heuristics have also been applied, for example to one-dimensional packing (Ross et al. 2002, 2003). For the 3D-SPP in particular, Pham (2011) proposed a univariate marginal distribu-

tion algorithm-based (UDMA) hyper-heuristic. This algorithm splits the container into sections and stores a collection of possible heuristics and measures the possibility of heuristic selection for each section of the container. The basic approach for the packing is an improved 3BF heuristic. The UDMA approach starts the packing with an empty container. The modified 3BF selects the smallest gap and the gap belongs to a section of the container. For each section, a heuristic is selected based on some heuristic selection probability. The packing is then processed until a full packing plan is completed. In each iteration, good solutions are collected and the heuristic selection probability is updated. As the process continues, good heuristics for a particular section will have a higher chance of being selected leading to a good mapping of which heuristics to choose for different sections of the container. For example, a box with more restricted rotation should be packed first to allow a box with more flexibility at the end of the packing. Therefore, a heuristic which chooses boxes that have a more restricted rotation constraint or less possible rotation for packing, will have a higher chance of being selected at the beginning of the packing but less chance of being selected towards the end of the packing.

2.2.4 Benchmark Data Sets

There are different benchmark data sets that have been proposed for packing problems. One of the most popular is the BR data set introduced by Bischoff & Ratcliff (1995) which contains 7 data sets (BR1-BR7) for container loading problems each with 100 instances. Each instance includes a set of boxes and a single container. The container has fixed width, length and height. Each BR data set has a fixed number of box types. For each box type, the dimensions of the box, the number of boxes and the rotation constraint are provided. The rotation constraint indicates the rotation ability of the box type around the x, y or z axes. If a box type has an axis rotation ability then it can rotate around the corresponding axis. An example of a box type from the BR data set with z axis rotation ability is shown in Figure 2.2. If a box type has rotation

ability on all three axes then there are maximum 6 possible rotations. The instance characteristics of the BR data sets BR1 to BR7 change from weakly heterogeneous to strongly heterogeneous. A weakly heterogeneous problem can be described as instances with a "small" range of box types. Whereas the strongly heterogeneous problem has a "large" range of box types. For example, the number of box types in each instance of the set BR1 is 3 and this increases to 20 box types in the set BR7. The BR data set was extended further to sets BR8-B15 by Davies & Bischoff (1999). These data sets have the same format as sets BR1-BR7 with a single container and 100 instances per set but the number of box types are increased, each instance in BR8 has 30 box types and each instance in BR15 has 100 box types. This BR data set is available from the OR Library (http://people.brunel.ac.uk/~mastjjb/jeb/info.html). To investigate the 3D-SPP, the BR data set can be adapted by keeping the same width and height of the container but extending the container length to fit all boxes.

In this thesis, in addition to the BR data set, the BRXL data set proposed by Bortfeldt & Mack (2007) is also used. The BRXL data set is an extension of the BR1-BR10 data sets to BRXL1-BRXL10 respectively. The container in the BRLX instances has the same width and height as that in the BR data set, and the length can be extended. However, the number of each box type is increased by a factor of $1000/n$ where n is the original number in the BR data set. Since this can result in a non-integer value, the number of boxes is rounded and the quantity of the last box type is adapted to have a total box number of 1000. The rotation constraint of each box type is the same in the BRXL data set as that in the BR data set.

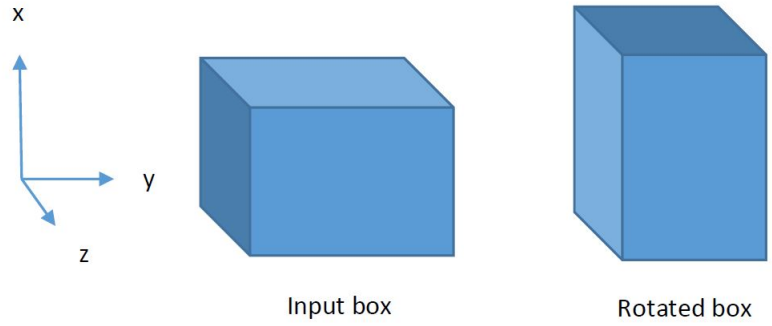


Figure 2.2: Box with z axis rotation properties

2.3 Multiple Carrier Transportation

In this section, the transportation planning problem arising in a real-world business operation at 3T Logistics Ltd is described. This problem is identified as a Single-customer Multiple-carrier Transport Planning problem in section 2.3.1. In Section 2.3.2 we provide some background about clustering algorithms which are a critical component in the solution approach for this problem. Section 2.3.3 presents a brief literature review of the vehicle routing problem with time windows which shares some similarities with the Single-customer Multiple-carrier Transport Planning problem tackled in this thesis.

2.3.1 Single-Customer Multi-Carrier Transportation Planning

3T Logistics Ltd (3T) is a fourth-party logistics (4PL) company based in Leicester, UK. The 4PL concept was introduced by Andersen Consulting (now Accenture) in 1997 as a result of a consultant contract with its customers (AliReza & Mehdi 2010, Bedeman & Gattorna 2003). In this particular model, customers outsource logistic operations to a 4PL company. Different to the 3PL model, where the 3PL company owns vehicles and operates the physical deliveries, 4PL companies only deal with the management of the logistic operations for the customer. Due to nature of the model, most 4PL compa-

nies work with intellectual capital and IT systems. 3T provides logistic solutions via its own transportation management system. A considerable part of the services provided by 3T is planning the delivery of goods from customers to consignees via a network of carriers. This problem is identified by Landa-Silva et al. (2011) as a Single-Customer Multi-Carrier Transportation Planning problem. The problem is to build vehicle loads (Eilon & Christofides 1971, Agbegha et al. 1998) and plan the routing of vehicles considering delivery time windows (Berger & Barkaoui 2003, Bräysy 2003*b*, Ibaraki et al. 2002). Landa-Silva et al. (2011) proposed a hybrid method including clustering, heuristic, local search and integer programming which generated significant savings in 3T's France operations. After a long process of observation and identification of practical operational requirements, analysis and modelling, 3T's transport planning problem was defined as Single-Customer Multi-Carrier Transportation Planning (SMTP). SMTP can be described in short as follows: a set of shipments is collected from the same source and then delivered to a range of destinations using a set of carriers. The basic requirement is to generate the most cost effective and high-quality transportation plans. The problem can be described in more detail as follows. From a source location, shipments are to be sent to customer destinations. Each shipment has different sizes and delivery time windows. Depending on their size, shipments are classified into Full Truck Load (FTL), Less Than Truck Load (LTL) and Groupage. It is possible to have multiple shipments going to the same destination on the same day of delivery. Shipments will be allocated to a plan with different transportation modes: load mode or parcel mode. A set of carriers is available to delivery the shipments. Each carrier will have its own pricing defined according to the transportation mode required and their availability. A common requirement in the SMTP faced by 3T is that backward mileage is undesirable. Backward mileage is when the next delivery is closer to the source than the previous delivery, i.e. it is expected that each delivery is further from the source than the previous one. The following are the constraints arising in SMTP:

- Vehicle capacity must not be exceeded.
- Carrier availability must be obeyed.
- The working time of each vehicle must not exceed more than 12 hours continuously.
- The starting point of each vehicle must be the source location.
- Backward mileage must not exceed 15 miles.
- For each location, loading time is considered to be 30 minutes.
- Each vehicle can visit up to 6 destinations.
- Delivery must be made during the specified time window.

The quality of a transportation plan is measured by considering carrier cost, time window violations, driving mileage, driving time, vehicle utilisation and backward mileage. Due to commercial sensitivity of the information, details of the evaluation function cannot be published. In summary, it is a weighted sum of a number of factors such as working hours, distance, etc. However, time window violations and cost are the major factors and the other factors are then used as tie breakers. The current automated solution acts as a decision support system and there is no evaluation of the overall quality of all plans. At the end of the algorithm execution, it is up to the human planner to decide if there are changes required to the generated plan in order to suit live operations. This approach is necessary because of the dynamic changes that happen during operations and such changes are not anticipated in the automated solution process. For example, the availability of a carrier may change or special and urgent deliveries may arise after the plans have been generated.

To the best of our knowledge, single-customer multiple-carrier problem was first mentioned by Brown & Ronen (1997) and were referred to as consolidation customer order into truckloads. There are two main differences in that problem by Brown & Ronen

(1997) from the SMTP problem described by Landa-Silva et al. (2011): the truck may load from more than one source and there may be a requirement to have flexibility in the result. Brown & Ronen (1997) approach starts by generating all combinations of the order into loads. Due to the restriction of tight operational rules, less than 10% of the combinations are evaluated. From validated combinations, an Elastic Step Partition model has been used to build the load where no order is assigned onto more than one load. The use of the Elastic Step Partition model is critical to allow constraint violation at a cost as described by Brown & Ronen (1997) . This is different to the use of Linear Programming by Landa-Silva et al. (2011). Caputo et al. (2006) investigate transport planning with two different modes: full truck load (FTL) and less than truck loads (LTL). These are two popular costing methods which 3T Logistics Ltd has encountered within the transport industry. In Caputo et al. (2006), orders are combined into compatible order group by compatible geographical and cost requirement criteria. An optimisation process is performed on each group. The optimisation starts with the allocation of a large order which can only be delivered by full truck load into an optimal carrier. The rest of the orders are heuristically divided into FTL and LTL groups. Division processes start with all orders allocated to FTL groups. Then, orders with the highest cost difference to the average of the FTL group are transfered to LTL group. The division process continues until the number of FTL is reduced by 2. The core of the remaining process is the load building from orders in FTL groups. Caputo et al. (2006) use GA to find a solution by encoding the assignment of each order to each truck into a binary format. There are two main points which are different to Landa-Silva et al. (2011). These are: order quantity can be split between trucks and there are no constraints in available quantity of carrier. Günther & Seiler (2009) investigate a similar problem compared to Caputo et al. (2006) with additional constraints such as time windows and temperature specifications. Günther & Seiler (2009) propose a two-phase approach. The first phase combines orders using four combination schemes: bundling, inbound milk run, outbound milk run

and pick-up delivery. The bundling scheme targets similar orders with the same source and destination. The milk run schemes combine orders which share either the source or destination. The pickup delivery scheme combines orders where the source is close to the destination of other orders. The second phase selects generated order combinations to maximise cost savings. There are two options: first to use linear programming and second to use heuristic. The heuristic approach starts by selecting the maximum cost saving combination until all orders have been selected. The linear programming option always performs better than the heuristic one, however it was also mentioned in the same article that heuristics were developed so that it can be integrated into a Transportation Management System (TMS) environment. There are two major differences to Landa-Silva et al. (2011): there is no restriction of carrier availability and the orders can come from different sources.

2.3.2 Clustering Algorithms

The approach described in Landa-Silva et al. (2011) starts with clustering the various destinations and then building routes based on those clusters. Therefore, good clustering plays a major role in producing high quality transportation plans. In this section, a literature review is provided on clustering algorithms that are related or applicable to the SMTP.

A clustering algorithm is a form of unsupervised learning in which data elements that do not have pre-defined label are grouped into clusters. A clustering algorithm automatically takes input data and separates it into a finite and discrete number of classes so that each class contains data elements that share some similarity. In general, there is no one clustering algorithm that provides a high quality solution for all problem domains. In the context of this thesis, clustering algorithms are used to classify the destination locations into geographical clusters. Then, vehicles will make deliveries of shipments to those destinations that belong to the same cluster or group. For example, shipments

around Nottingham will be loaded and delivered with the same vehicle. Several surveys have been published to review different types of clustering algorithms and studying their advantages as well as disadvantages. Clustering algorithms can work with many types of data input and similarity measures. In the SMTP context, data input is refered to as points (locations) on a two-diemensional surface. Comprehensive reviews of clustering algorithms can be found in Jain et al. (1999), Xu & Wunsch (2005) and Kotsiantis & Pintelas (2004). Clustering algorithms can be classified into different categories: hierarchical method, partition method, density-based method, grid-based method (Kotsiantis & Pintelas 2004).The fuzzy-based, kernel-based and neural network-based methods are summarised by Xu & Wunsch (2005). A combination of clustering algorithms is also possible, for example Strehl & Ghosh (2003) proposed different ways to combine different clustering techniques.

Hierarchical Clustering

Hierarchical clustering algorithms take data input and build a tree structure of clusters called a *dendrogram*. There are two major approaches in hierarchical clustering. The first approach is ameliorative (bottom-up) where data points are merged together to form larger clusters for the next level up the tree (Jain & Dubes 1988). The other approach is divisive (top-down) which starts with all data points belonging to a single large cluster(Kaufman & Rousseeuw 1990). At each level going down from the top, clusters are separated into smaller clusters. The process continues until some termination condition is met. A typical condition for termination is when the required number of clusters is found. It is also possible to fully build a tree structure of clusters so that different clusters can be identified at different levels of the tree. One of the most popular algorithms in hierarchical clustering is linkage metrics. As mentioned above, hierarchical clustering involves splitting or merging data points to form clusters. The mechanism of merging and splitting clusters depends on cluster similarity or distance measurement. The similarity of the clusters is called linkage metrics. Using different linkages can greatly

affect the tree structure of the cluster and therefore affect the overall outcome of the algorithms. Common types of linkage are: single linkage, complete linkage and average linkage. Single linkage is the distance between the closest items of two clusters. Complete linkage is the distance between the furthest items between two clusters. Average linkage is the average distance of all possible distances between items of two clusters. One of the most popular hierarchical clustering algorithms is SLINK which was introduced by Sibson (1973). SLINK is an ameliorative single linkage cluster algorithm. An example of complete linkage can be found in Defays (1977) and an example of average linkage can be found in Voorhees (1986). One of the properties of this particular type of approach is that linkage based clustering naturally produces a cluster with a convex shape. The performance of linkage based algorithms is affected if the data contains non-convex clusters. This leads to the next class of hierarchical clustering where arbitrary shapes are taken into account. Guha et al. (2001) presented clustering using a representative (CURE) algorithm. Instead of using one point to represent a cluster, CURE uses a number of points that are selected to represent a cluster. The similarity measurement between clusters is calculated by combining the distance between representative points. CURE can detect non-spherical shapes by choosing representatives at different locations across clusters. After merging, representative points of the cluster are sunk to the centroid of the cluster helping to avoid the situation where an outlier point is selected to represent the cluster. Robust clustering using linkage ROCK algorithm was introduced by Guha et al. (2000) which clusters the data set in a similar manner to CURE, however, it works with categorical attributes of data. A different presentation of the cluster of CHAMELEON algorithms was proposed by Karypis et al. (1999). For each point, a number of closest or most similar points are stored in a graph and the others are removed. CHAMELEON algorithms have two stages: the first stage generates a graph of each point with its nearest neighbour point called k-nearest neighbour graph; the second stage merges clusters together and forms a larger cluster in an agglomerative manner. The measurement of

similarity between clusters is made using both inter-connectivity and relative closeness (Berkhin 2006).

Partition Method

As described above, hierarchical clustering, data input or data points are merged with other points to form clusters. Partition method algorithms start by assigning data points to clusters and iteratively improve the clustering by relocating points between clusters. This is very different to hierarchial clustering where there is no change to a cluster after the cluster has been formed. In order to start partition clustering, it is important to have a representation of a cluster. There are three main types of representations: centroid, medoid and probabilistic models. Centroid algorithms use generated points to represent the cluster centre. K-Mean is one of the most popular algorithms of partition methods and it is a centroid clustering algorithm. This algorithm separates input points into a predefined number of k clusters. It starts by randomly choosing k centroid points. Then, at each iteration, input points are allocated to the nearest centroid. After each allocation, the centroid points are re-calculated based on the current points in each cluster. The process continues until there is no change in the points allocated to each centroid. A corresponding algorithm to K-Mean is the K-Medoid algorithm introduced by Kaufman & Rousseeuw (1987). Medoid algorithms use actual input data points to represent the cluster centre. Similar to K-mean, K-Medoid selects k points from input points to become medoid. Other points are then associated to the closest medoid. In each iteration, each medoid is evaluated with all other non-medoid points and swaps between points to improve the clustering. This process continues until there is no change in the medoid. K-Mean and K-Medoid are suitable algorithms for large data sets and when a fast running time is required. However, these methods are sensitive to noise and termination normally results in a local optimal point. Performance of clustering algorithms depends on the number of clusters selected. To overcome this issue, different values for the number of clusters are normally tried for the given problem domain or, alternatively, heuristics can

be used to find the best solution. Probabilistic clustering methods use statistical distribution models to represent clusters. In probabilistic clustering, a point belongs to a cluster when it has the highest possibility of belonging to a corresponding distribution model. Most of the probabilistic models algorithms are based on the Expectation-Maximisation (EM) algorithms described by McLachlan & Krishnan (2007). One of the most popular models for EM clustering is the Gaussian Mixture Model. EM starts with a randomly generated model with random parameters. The model is used to calculate the possibility of a point belonging to a cluster and separates them by assigning them to the cluster with the highest probability. Points belonging to the same cluster are then used to re-evaluate the parameters of the model. This process continues until it reaches a stable model.

Density-Based Clustering

Density-based clustering algorithms classify the data input into clusters by using the density of points in an area. An area with high density can be a non-convex or arbitrary shape cluster and therefore density-based clustering can identify non-convex clusters. This is an advantage over the partition method clustering algorithms such as K-Mean. The most popular density-based clustering algorithm is DBSCAN as described by Ester et al. (1996). DBSCAN algorithm is driven by two parameters: ε - the maximum distance between two neighbour points and MinPts - the minimum number of neighbour points required. DBSCAN defines the following:

- Core object is a point which has more than MinPts point within a distance ε.
- Point x is directly density reachable to point y when the distance from x to y is less than ε and x is the core object.
- Point x is density reachable to point y when there is a directly density reachable path from one point to the other point with starting point x and ending point y.
- Point x is density connectivity to point y when there is another point z that is density reachable to x or y.

Based on the above definitions, a cluster is formed by core points and points with density connectivity to a core point. Points belonging to a cluster that are not the core point are border points. Other points which do not have any connection are considered noise. The DBSCAN algorithm has limitations in searching for neighbours of a point with high dimensional data. However, in the context of this thesis, two-diemensional data is the main interest. A generalisation of DBSCAN, called GDBSCAN, was introduced by Sander et al. (1998). DBSCAN is also sensitive to the selection of parameter values. There is no simple method to identify the best value for ε and MinPts given the data points. To overcome this DBCLASD, introduced by Xu et al. (1998) can be used as it does not need both parameters. In addition, DBSCAN has a fixed ε which effectively restricts the density of clusters, therefore, clusters with variable density are not recognisable by DBSCAN. OPTICS clustering algorithms proposed by Ankerst et al. (1999) also address the limitations of DBSCAN. OPTICS introduce two additional definitions: core-distance and reachability-distance. The output cluster analysis of OPTIC algorithms is not the clustering itself, but a cluster ordering structure which can then be used to extract the cluster. OPTICS can find clusters which have density less than ε.

Grid-Based Clustering

Grid-based clustering splits the area into smaller segments and applies clustering operators to the segments. Each segment (e.g a cube in three-dimensions or a region in two-dimensions) can contain items. A single item segment is called a unit. Segments that contain many elements are dense segments and clusters are made by combining dense segments together. It is worth noting that while density-based clustering described above works best with numerical attributes, grid-based clustering works best with different types of attributes (Berkhin 2006). STING clustering algorithms proposed by Wang et al. (1997) divide data input into tree structures of grid cells. For each cell, two types of parameters are stored: attribute-dependent and attribute-independent. The attribute-dependent parameters are: mean, standard deviation, minimum, maximum and distribu-

tion type. The attribute independent parameter is the number of items in the cell. After the tree structure is generated, similar cells are merged as in DBSCAN. CLIQUE is a clustering algorithm for high dimensional data input. For each of the attribute dimensions, the value distribution of each attribute is stored in the one-dimensional array. CLIQUE combines a set of two attributes to create a two-diemensional distribution space. Dense rectangles in two-dimensional distribution space are represented in a connected graph. The cluster is formed in a bottom-up fashion by merging the vertices of the graph.

2.3.3 Vehicle Routing Problems With Time Windows

The vehicle routing problem (VRP) is a real-world optimisation problem which has received much research interest over the last few decades. In simple terms, the vehicle routing problem refers to finding routes for the delivery of shipments to customers using a fleet of vehicles and subject to some constraints whilst aiming to maximise a specific objective. For example, the objective could be to maximise vehicle utilisation while the constraint could be to satisfy the time windows given for the deliveries (Balakrishnan 1993, Desrosiers et al. 1995, Tang et al. 2009). Other objectives that can be considered are the minimisation of the number of vehicles required (Solomon 1987, Bräysy 2003*a*) or minimisation of the distance travelled and cost (Gendreau et al. 1996, Tas et al. 2013). Several extensive surveys have been conducted for the VRP and its variants, for example Laporte (1992), Solomon (1987), Laporte et al. (2000), Berbeglia et al. (2007) and Golden et al. (2008). For the scope of this thesis, some VRP literature that is related to our SMTP problem is reviewed. As described in Section 2.3, the SMTP problem has some of the constraints that arise in VRP problems, such as time windows, vehicle capacity, etc. Therefore, the focus of this thesis is on the Vehicle Routing Problem with Time Windows (VRPTW). The VRPTW can be described as the problem of generating delivery routes for a fleet of vehicles with the following constraints:

1. All deliveries have the same starting point, i.e. a single depot.

2. The sum of the size of all shipments in a single route cannot exceed the given vehicle capacity.

3. Each delivery destination has a time window with a given start time and end time.

4. Each delivery must be made within the given time window. If the vehicle arrives at a delivery point before the start time of the time window, then the vehicle has to wait until the time window starts.

5. After completing the last delivery, each vehicle comes back to the starting point or depot.

6. Each destination is only visited once.

The VRPTW has received much attention from researchers. Surveys of solution techniques are given by Cordeau et al. (2002) and Bräysy & Gendreau (2001*b*, 2005*a*,*b*). Since VRP is an NP-hard problem, VRPTW is also NP-hard. Indeed, VRPTW with a constant number of vehicles is described as an NP-hard problem by Solomon (1987). Therefore, heuristic based approaches are the most common in the literature for tackling this problem. In practice, in some scenarios the time windows can be relaxed (soft time windows). Balakrishnan (1993) proposed three heuristics for the Vehicle Routing Problem With Soft Time Windows (VRPSTW). The following sections provide a literature review of constructive heuristics and local search methods applied to this problem.

2.3.3.1 Route Building Heuristics

Solomon (1987) proposed four heuristics for constructing routes: Max Time Saving, Nearest Neighbour, Insertion and Sweep Heuristic. The Max Time Saving heuristic is based on the heuristic first introduced by Clarke & Wright (1964). This heuristic starts by assigning each delivery point to one dedicated vehicle. Then all routes go through a tour building procedure in which routes are merged so that cost saving is maximized.

Solomon (1987) included time window constraints into the route orientation during route merging. The Nearest Neighbour heuristic starts building routes by adding the "closest" delivery point from the depot. The next delivery is the delivery point that is the "closest" to the last current delivery in the route without any constraint violation (vehicle capacity or time window). If no feasible route can be formed by adding the delivery, then a new route with an empty vehicle is started. The Sweep Heuristic uses the heuristic given by Gillett & Miller (March/April 1974). Firstly, the delivery point locations are divided into geographical segments defined around a centre point. This is a representation of real-life practice when planning vehicles coming out from depots in different segments, the human planner will try to make sure the segments are disjointed. The shipments in each segment are allocated to a route using the insertion heuristic. The Insertion Heuristic selects an unplanned shipment based on two criteria c_1 and c_2. The first criterion c_1 determines the best position to insert a unplanned shipment. The second criterion selects which unplanned shipment will be inserted. Solomon (1987) introduces three different formulae for the two criteria which have different performance. In the most effective formula, c_1 selects the insert position that minimises the change in the combination of distance and time using a weighed sum formula and c_2 chooses to insert the shipment that has the lowest cost by directly combining distance to unplanned shipment and the first criterion.

Foisy & Potvin (1993) provide a parallel implementation of the insertion heuristic in Solomon (1987) with significant improvement in computation time. Ioannou et al. (2001) proposed an IMPACT algorithm as another insertion heuristic. Ioannou et al. (2001) puts emphasis on the requirement of the plan building method: short distances between deliveries, minimal number of vehicles required, minimal impact of planning shipments. IMPACT starts by seeding routes with a number of routes and a route is built using a tour building approach similar to that in Solomon (1987). The IMPACT algorithm introduces three criteria:

- IS_u - measures the impact of the arrival time on the candidate shipment u itself.

- IU_u- measures the impact of the candidate shipment time window on the unplanned shipments.

- IR_u - measures the impact of the candidate shipment on the other shipments. It is the weighted sum of c_1 and c_2 as in Solomon (1987) and c_3 which utilises the time windows of the candidate shipment.

The weighted sum of IS_u, IU_u and IR_u forms the overall IMPACT criterion and the shipment that minimises this weighted sum is selected. Dullaert & Bräysy (2003) identified that the insertion of a candidate shipment into the beginning of the route might increase the waiting time of the next shipment. Therefore, the original c_2 criterion can be under-estimated. A Push Backward Maximum Push Forward (PBmaxPF) criterion is proposed to avoid such under-estimation, hence the formula is modified to c_{12} as in the following equation:

$$c_{12}(i,u,j) = \begin{cases} [(b_j - t_{ij}) - (b_u - t_{iu})] + (b_{ju} - b_j) & i = i_0 \\ (b_{ju} - b_j) & otherwise \end{cases} \tag{2.1}$$

PBmaxPF gives a significant increase in the cost saving if there is a small number of deliveries on the route. However, as the number of shipments in the route increases, the effect of PBmaxBF is reduced. This is due to the significant cost saving of the first delivery being less than when there are many deliveries in a route. Bräysy (2003*a*) proposed two route construction heuristics: Hybrid Construction and Merge Heuristic. The basic ideas behind these two heuristics are from Solomon (1987) and Clarke & Wright (1964). Hybrid Construction extends the seeding selection of shipments at the beginning of route construction. After choosing furthest from the source shipments, the subsequent seeds are selected using different criteria. The cost for the insert function is a weighted sum evaluation of saving distance and saving waiting time and direct distance from source to the insert shipment. In Merge Heuristic, the cost of a route is a weighted

sum of the total distance and the total waiting time in the route. The cost saving formula is the saving of route cost. In addition, the merged route is re-sequenced to have better cost saving after a certain number of shipments on the route. Details of the Hybrid Construction and Merge Heuristics can be found in Bräysy (2003*b*), Bräysy & Gendreau (2001*b*).

2.3.3.2 Local Search Methods

A common approach for tackling the VRPTW is to start with an initial solution and improve that solution iteratively using local search until no further improvement can be made. At each local search iteration, one or more neighbourhood solutions are generated from the current solution using some local move operators. Neighbour solutions are compared to the current solution and based on some acceptance criteria, the neighbourhood solution could be selected to become the new current solution for the next iteration. This process continues until some termination criterion is met, such as no improvement for some number of iterations or limited computation time. Popular candidate solution acceptance criteria are first accept or best accept. In first accept, the first improving neighbour solution encountered is selected to become the current solution. In best accept, all neighbour solutions for the current solution are generated and the best one is selected. The selection of move operators is critical for a good performance of a local search. A solution to the VRP can be represented as a graph where each vertex is a shipment delivery point and each edge is the route between delivery points.

There is a wide range of move operators and local search approaches that have been proposed in the literature for the VRP and its variants. Most of these procedures are edge-exchange approaches (Bräysy & Gendreau 2005*a*). There are two main types of operators: intra-route and inter-route. Intra-route operators make a modification to a single route, they conduct a type of re-allocation of deliveries within a route. Inter-route operators make changes between routes, they act like a swapping of deliveries between

routes. The most common inter-route operators make changes between two routes only. Croes (1958) introduced the 2-opt operator for the Traveling Salesman Problem (TSP). This operator selects part of a route, reverses its direction and then combines this with the rest of the route. Lin (1965) implements a 3-opt operator as an extension of 2-opt, where one more edge is included in the change. Or (1976) introduced the Or-opt operator where a number of shipments in the route are selected and moved to another part of the route while maintaining the direction of the route. Potvin & Rousseau (1995) proposed 2-opt* operators which combine exchanging parts of two routes. The 2-opt* operator splits two original routes into two start-parts and two end-parts. Then, 2-opt* exchanges the parts so that the start-part of one route is connected to the end-part of the other route whilst maintaining the direction of each part.

Prosser & Shaw (1996) presented a maximise saving local search approach using four operators: 2-opt, Cross, Reallocate and Exchange. The Cross operator is an inter-route operator which makes changes to two selected routes. The Cross operator selects and swaps single shipments between routes. It selects a shipment from one route and moves it to a different route. The Exchange operator selects and swaps two shipments on two routes. Prosser & Shaw (1996) also studied the effect of the operators on the overall performance of the local search procedure and showed that the Reallocate operator had the most positive effect and the Exchange operator had the least positive effect. Osman (1993) introduced a λ-interchange generation mechanism which performs a modification between two routes. For each selected route, n and m number of shipments are selected where n, $m \leq \lambda$. The selected shipments are swapped between two routes. Typical values for λ are 1 or 2. A special case for λ-interchange is CROSS exchange which was proposed by Taillard et al. (1997). Instead of single shipments as in Prosser & Shaw (1996), the CROSS exchange operator selects sequences of shipments which are then swapped between two routes whilst maintaining the direction of the routes and swapped sections. Glover (1996) employed a stem-and-cycle reference structure and ejection chain

approach for TSP. The Sub-path Ejection Method is an intra-route operator which generates neighbour solutions of single routes. The idea of an ejection chain is to create a sequence of remove and insert moves of shipments on a route (Lin & Kernighan (1973)). Gendreau et al. (1992) applied two types of GENI operators where a shipment is inserted into a route between delivery points that are not adjacent in the sequence. After the insertion, the direction of part of the route might change. Two types of GENI-Unstringing operators were also proposed, these are reversed versions of GENI operators. Most of the inter-route operators modify a solution by applying changes between two routes. Thompson & Psaraftis (1993) proposed a cyclic k-transfer operator where a k number of shipments are swapped between multiple routes. Due to the increase in the size of the search space when using cyclic transfers, a general methodology for cyclic transfer neighborhood searches was used by Thompson & Orlin (1989).

Caseau & Laburthe (1999) suggested an incremental local optimisation approach. The initial solution starts with a fixed k number of empty routes, then shipments are iteratively inserted into the route, k is the maximum number of routes allowed. Instead of applying a solution improvement operator to a complete initial solution, improvement operators are applied after a shipment is inserted into the route. Three operators have been chosen: 2-edge exchange (2-opt operator), 3-edge exchange (or-opt operator) and node transfer operator. The node transfer operator applies to the route that was not affected by the shipment insertion. Shipments which are close to any shipment on the selected route and have a cost saving are transferred on to the selected route. The node transfer operator is applied if the other two operators found some improvement. All operators use the first accept criterion. So, as soon as an improved solution is found, the process goes to the next iteration. Since the maximum number of routes is fixed, it is possible that an insertion cannot be made. In this case, three operators have been selected to resolve the situation: 1) Swap, 2) Relocate and Flush and 3) Relocate. The idea of these three operators is very similar to the ejection chain method where a sequence of

modifications are made to create a feasible solution. The Swap and Relocate operators are similar to the Or-opt and Relocate Operators introduced by Or (1976) and Prosser & Shaw (1996). The Relocate and Flush operator removes all possible shipments that can be moved to other routes so that new shipments can then be inserted.

Caseau & Laburthe (1999) showed that incremental local optimisation is not only faster but also produces better results than applying improvement moves after the initial solution is built. This is especially the case in problems of large size. Bräysy (2003*a*) proposed a three-phrase approach. The first phase was to create an initial solution using one of two heuristics: Hybrid Construction Heuristic or Merger Heuristic (described in section 2.3.3.1). The second phase is a local search method based on an ejection chain method with reordering of shipments during the local search. Each route is selected to search for improvements. If no improvement can be made, then other shipments near to the route are added to the current route as far as possible. This is to improve the chance of improvement neighbourhood route. The third phase is to use an Or-opt operator to minimise the total distance of routes.

3 Overhead Estimation and Constructive Heuristics for 3D-SPP

In this chapter, some modifications of the best-fit methodology for the three-dimensional strip pack problem are developed. First, section 3.1 revisits the three-dimensional best-fit (3BF) heuristic that was introduced by Allen et al. (2011). Modifications to the 3BF heuristic with block generation and block reallocation are proposed in section 3.2. The modified heuristic is denoted as 3BFBL. Later, in section 3.3, an overhead estimation approach to work with 3BFBL, called OH-3BFBL, is presented. The performance of OH-3BFBL is evaluated using data sets from the literature and compared to other approaches published in the literature in section 3.4. Further modifications to block generation and to identify candidate positions are introduced in section 3.5. Finally, section 3.6 presents experimental results.

3.1 Three-Dimensional Best Fit (3BF) Algorithm

The 3BF heuristic developed by Allen et al. (2011) draws inspiration from the 2D-packing algorithm from Burke et al. (2004) which utilises a best-fit methodology. The 3BF heuristic packs each box in the lowest possible gap in order to fill as much gap as possible. The packing process continues until all boxes are packed into the container. When a gap is selected, boxes are allowed to rotate in order to find the best-fit orientation. There are three different cases:

• One or more boxes can totally fill the gap. If there is more than one box, a tie breaker policy is used.

• One or more boxes can partially fill the gap. The box that fills the gap the most is selected using a tie breaker policy if necessary.

• No box can fill the gap. The gap is discarded and the next available gap is selected.

Four tie breaker policies were proposed by Allen et al. (2011): deepest bottom left most, maximum contact, smallest extrusion and neighbour score. Although not mentioned in the paper, preliminary experimentation showed that the maximum contact policy has the highest utilisation compared to other policies. The maximum contact policy chooses the box with maximum volume and places it so that the contact area with other boxes and the container is maximised. In Allen et al. (2011), the contact area of the box with other boxes and with the container are weighted differently. Different weighting parameters could affect the quality of the final solution. Details of the parameter values used in this thesis will be described in section 3.2. The "Extreme point" method described by Crainic et al. (Summer 2008) is used for the representation of gaps and boxes. At the beginning of the packing, there is only one candidate point, given by the coordinates (0,0,0). After placing a box, that position is removed from the candidate list and new points are added. In the candidate list, points are sorted in the deepest bottom left order. The gap of one layer is represented by a set of candidate points that have the same depth.

In the 3BF heuristic, only one box is placed per iteration. This can lead to different selections of the next box to pack. For example, it is possible to have a single larger box or a group of smaller boxes. 3BF will select the single biggest box to pack first. However, it is possible to group smaller boxes together to form a block which is bigger than the single big box. This can produce a sub-optimal plan as shown on the left of figure 3.1. 3BF will select the blue box to be packed first. However, this will provoke a situation where the yellow boxes have no other option than being packed as shown on the left of the figure. However, if the smaller yellow boxes are grouped together first, then 3BF

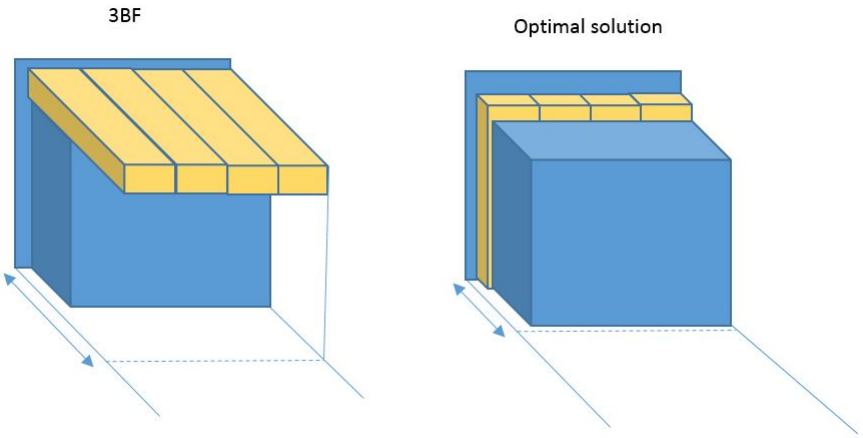


Figure 3.1: Solution obtained with 3BF heuristic and optimal solution with a large box and a group of smaller boxes

will output the optimal solution as seen on the right of figure 3.1, i.e the grouped yellow boxes will be packed first.

Another issue of the 3BF heuristic is the use of extreme point which positions the block to the deepest bottom left-most valid position. This can result in an unusable space as shown on the left of figure 3.2. Box B is packed adjacent to block A and this creates an unusable empty space which cannot be filled by any remaining boxes. It will be more effective to reallocate B to the furthest possible position along the X axis. This creates a bigger gap to be used for packing subsequent boxes as shown on the right of the figure.

In order to improve the resulting packing, Allen et al. (2011) applied tabu search (TS) and deepest bottom left fill heuristics at the end of the packing instead of using only 3BF. This helps to avoid a "tower building" effect with the last few boxes. However, this can still be a problem when boxes in the second phase have limited rotation and a large dimension in the Y axis. An example is shown on the left of figure 3.3, 3BF+TS starts the second phase with the yellow boxes which have only one possible rotation. There are limited options with 3BF+TS to improve the overall solution. In weakly heterogeneous cases, where the quantity of each box type is large, this can have a significant negative

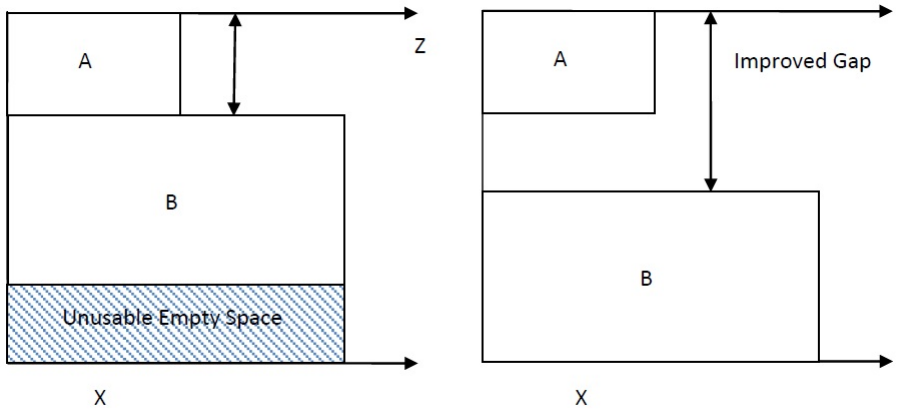


Figure 3.2: Example showing how a bigger gap is created by appling block reallocation

impact on the final result produced by 3BF+TS.

3.2 Modification to 3BF+TS

In this section, we introduce three modifications to the 3BF+TS procedure in order to address the issues discussed in the previous section. The first modification is that instead of using a single box, boxes are grouped into blocks for packing, this is described in detail in section 3.5.1. The second modification introduces a reallocation process and is described in section 3.2.2. Then, the proposed three-dimensional best-fit heuristic with blocks (3BFBL) is described in section 3.2.3. The third modification is the introduction of an overhead estimation approach instead of tabu search to improve the result from the heuristic (OH-3BFBL), details are given in section 3.3.

3.2.1 Block Generation

Instead of packing one single box per iteration, packing blocks of boxes can be considered. Two types of blocks are defined in this thesis: *Simple block* and *Group block*. A Simple block is a group of boxes of the same type. If boxes have more than one orientation, different orientations can be considered. A Simple block can be defined as a box with

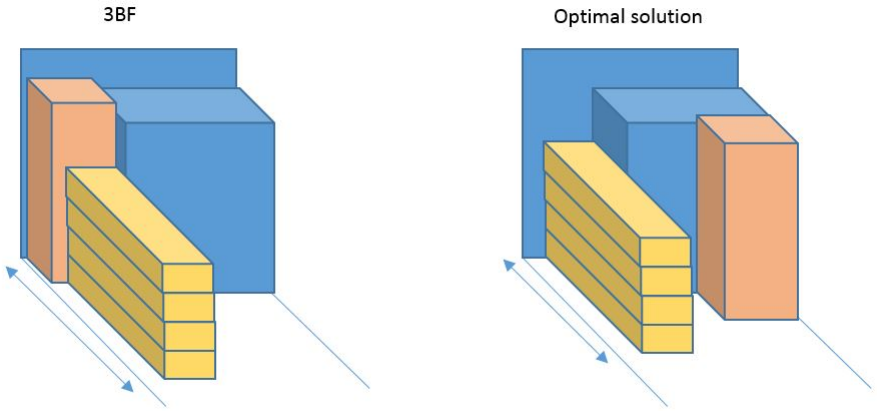


Figure 3.3: 3BF+TS where the second phase starts with the yellow boxes compared with the optimal solution

particular orientation, number of boxes across the X axis and number of boxes across the Y axis. The number of boxes across the Z axis is always 1. Figure 3.4 shows an example of a Simple block with 3 boxes across the X axis and 2 boxes across the Y axis, the total number of boxes is 6. A Simple block is valid when:

- Width of block $\leq$ width of container.
- Height of block $\leq$ height of container.
- Boxes required to form the block is a subset of the input boxes.

A *Group block* is a group of two simple blocks: first block - fBl and second block - sBl. There are 2 different types of arrangement for the two simple blocks: "Next" and "Above" as shown in figure 3.5. When combining simple blocks together there might not be a perfect match because of the different dimensions hence some space is lost. In order to measure the quality of a Group block, the volume utilisation has to be greater or equal to v. The value for this parameter v measures the block's volume utilisation and is chosen empirically. The value that seems to work well with the 3BFBL heuristic is $v = 98\%$.

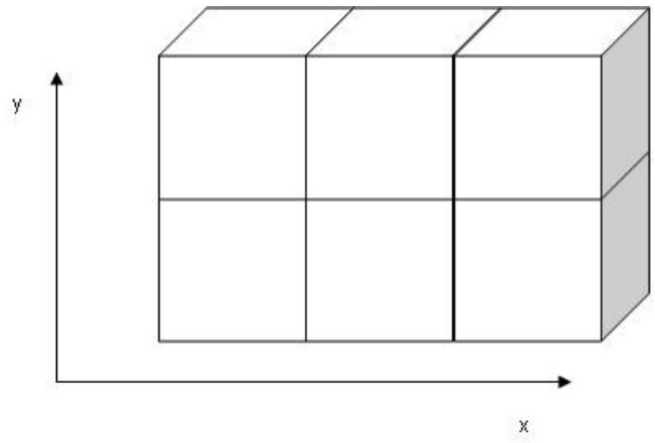


Figure 3.4: Simple block example

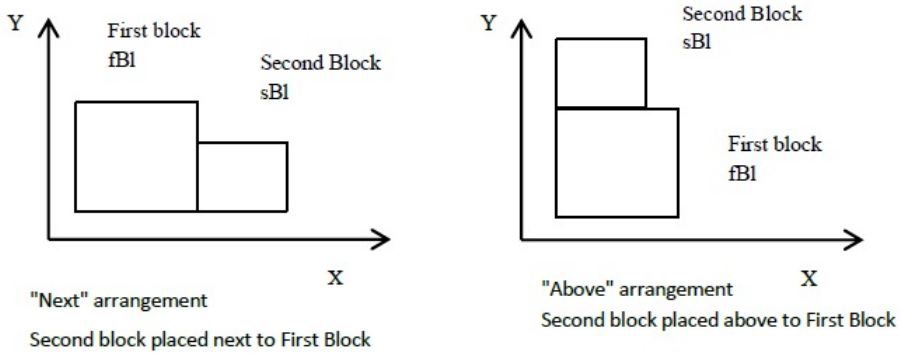


Figure 3.5: Group block examples

The volume utilisation of the block v is calculated as: $\frac{\sum SimpleBlockVolume}{EnvelopeBoxVolume}$, where the envelope box volume is the volume of the smallest rectangular box that can contain the simple blocks. Figure 3.6 shows an envelope box (the rectangular box denoted by dashed lines) containing a block with 2 boxes.

In addition to the condition $v \geq 98\%$ for volume utilisation, other conditions for a Group Block are as follows:

- Width of block $\leq$ width of container.
- Height of block $\leq$ height of container.
- Boxes required to form a group block are a subset of the input boxes.
- For an "Above" arrangement:
  - fBl.length $\geq$ sBl.length.
  - fBl.height $\geq$ sBl.height.
  - sBl is placed in point (fBl.width, 0, 0) relative to the position of fBl.
- For a "Next" arrangement:
  - fBl.length $\geq$ sBl.length.
  - fBl.height $\geq$ sBl.height.
  - sBl is placed in point (0, 0, fBl.height) relative to the position of fBl.

Blocks are generated with a block generation process before the packing process. Block generation starts by generating simple blocks first. Subsequently, group blocks are generated using the simple blocks generated earlier. Then, in terms of procedures we have two for block generation: SIMPLE which only generates simple blocks and GROUP which generates simple blocks first followed by generated group blocks. The setting for these block generation procedures is explained in section 3.4.1.

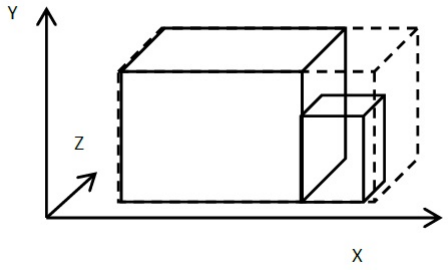


Figure 3.6: Example of an envelope box - minimum size of rectangular box that can contain all the boxes in a block

3.2.2 Block Reallocation

When block and position are selected, the block reallocation procedure is applied to investigate if there is a better position for future packing. This idea of reallocation was introduced by Gehring & Bortfeldt (1997). In this thesis we use a block reallocation procedure as illustrated in figure 3.7. First, we have to measure Max Left Gap which is the largest distance from the current block to other block or the container. As shown in figure 3.7, if the Max Left Gap is smaller than the smallest width of available boxes, then we shift the yellow block to be next to the red block. This is done by moving the position of the yellow block by Min Shift across the x-axis. Min Shift is the smallest distance from the current block to other block or to the container. In this example, by shifting the yellow block we have an Improved Right Gap which is wider and therefore can be used for future blocks. If the Max Left Gap is not smaller than the smallest width of available boxes then no reallocation is performed.

3.2.3 Procedure 3BFBL

In this section, the 3BFBL procedure is described including the modification mentioned above. The coordinate system shown in figure 3.8 is used, where the y dimension is the non-restricted dimension corresponding to the length of the container. The pseudo code for 3BFBL is shown in Algorithm 3.1. The first difference when compared to the

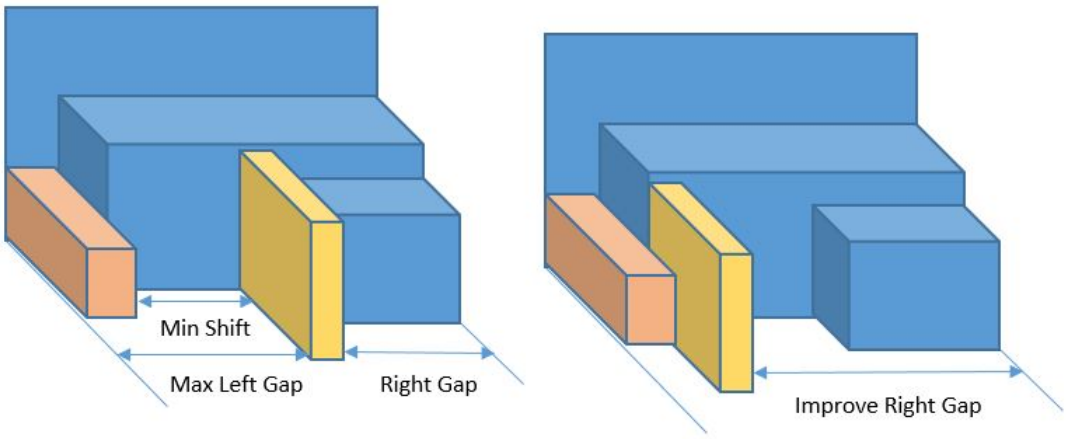


Figure 3.7: Block reallocation example

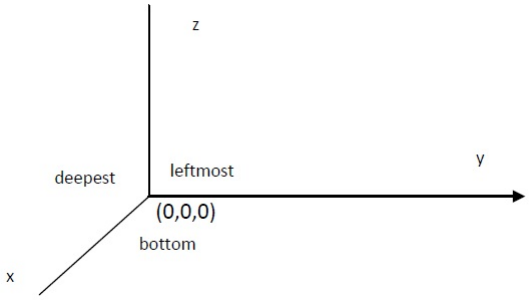


Figure 3.8: Coordinate system used where y is the non-restricted dimension

3BF heuristic is that blocks are generated in advance as described in section 3.2.1. Once the block generation is completed, packing is started and the finished when there is no box left in B. The heuristic starts by finding the lowest gaps. Instead of finding one box for the selected gap, 3BFBL finds the best-fit block p and the position to place it. The procedure to select block p is best-fit with Maximum Contact tie breaker as in 3BF. Before adding p to the current packing plan P, 3BFBL applies the reallocation procedure described in section 3.2.2 to p. If there is no block available then the current lowest gap is marked as invalid for the next iteration so that the next lowest gap is tried instead.

Algorithm 3.1 3BFBL Heuristic

Input: : container C, set of boxes to pack B, set of generated blocks BL
Output: : packing plan P, with all blocks and its position defined

```
1: P ← ∅
2: while |B| >0 do
3:     G ← GetLowestGap(C, P)
4:     p ← FindBestFitBlock(G, BL)
5:     if p is found then
6:         Reallocation(p)
7:         Add p to P
8:         Remove from B the boxes in p
9:     else
10:        Mark all gaps in G as invalid
11:    end if
12: end while
```

3.3 Overhead Estimation

The 3BFBL procedure is a "greedy" heuristic which arrives at a local optimum when building a solution (Pieterse & Black 2005). In particular, 3BFBL always chooses the block with the best-fit that fills most of the selected gap. However, a best-fit block is not necessarily the best block to select for the overall packing plan. In order to improve the overall planning result, an *overhead estimation* can be implemented. The 3BF+TS procedure is used by Allen et al. (2011). The rationale for using tabu search is to improve the result produced by 3BF towards the end of the packing process. Instead, here we propose the overhead estimation to improve the result produced by 3BFBL at the start of the packing process. The idea behind the overhead estimation is that instead of only choosing the best-fit block, a selection of blocks and their positions are considered. For each selection, a complete packing is produced using 3BFBL. The completed packing plans are evaluated to provide an overhead estimation of the corresponding utilisation. The block selection that achieves the highest utilisation for the completed packing plan is selected. The pseudo code for OH-3BFBL, the algorithm incorporating the overhead estimation procedure is shown in Algorithm 3.2.

OH-3BFBL takes the container, C, box set, B and generated block, BL. OH-3BFBL also has configuration parameters but for simplicity these have been omitted from the pseudo code. There are three parameters: number of blocks to estimate - n, gap step - g, time limit - t. The value of parameters will be specified in section 3.4.1. Similar to 3BFBL, OH-3BFBL starts by selecting the lowest gap (line 3). Different to 3BFBL, OH-3BFBL uses the Overhead Estimation procedure presented in Algorithm 3.3 to identify what block is to be packed (line 4). During the packing process, it is noticed that once one block is packed using Overhead Estimation then there is a limited choice for the following blocks. Therefore, instead of performing an overhead utilisation at every iteration, g number of blocks are packed just using 3BFBL (line 8). OH-3BFBL will stop when there are no boxes left to pack or the time limit has been reached (line 2). This will limit the run time of the algorithm so comparisons can be made with previous published approaches.

Algorithm 3.2 3BFBL With Overhead Estimation(OH-3BFBL)

Input: : container, C set of boxes to pack B and set of generated block, BL
Output: : packing plan, P with all blocks and its position

```
1:  P ← ∅
2:  while |B| >0 and time < t do
3:      G ← GetLowestGap(C, P)
4:      p ← OverheadEstimation(G, BL, P)
5:      if p is found then
6:          Add p to P
7:          Remove box in p from B
8:          Pack g block using 3BFBL
9:      else
10:         Mark all marks in G is invalid for future use
11:     end if
12: end while
```

The overhead estimation procedure shown in Algorithm 3.3 will find all valid blocks and select n block and position combinations (line 1). Each combination is then packed into the container. The rest of the boxes are packed using 3BFBL (line 7). The combination resulting in the highest utilisation completed packing plan - p is returned.

Algorithm 3.3 Overhead Estimation procedure

Input: : A gap, G and set of block, validBL
Output: : a block and position, p

1: $bestFitBL \leftarrow GetBestFitBlock(G, BL)$
2: $bestS \leftarrow 0$
3: $bestP \leftarrow nil$
4: **for all** p in bestFitBL **do**
5: $P' \leftarrow Copy(P)$
6: pack p in packing plan P'
7: $score \leftarrow CompletePackingUsing3BFBL(P')$
8: **if** score $>$ s **then**
9: $bestS \leftarrow score$
10: $bestP \leftarrow p$
11: **end if**
12: **end forreturn** $bestP$

3.4 Experiment

3.4.1 Experimental Setup

Algorithms are implemented in Java single thread code. Experiments are run on a PC with processor AMD at 2.0 GHz and 1GB of memory. The CrateViewer application is used to visualise the solution (Allen et al. 2011). Two experiments were carried out. The first experiment compares the performance of OH-3BFBL to other published approaches from the literature: TSACC-4P (Bortfeldt 1999), SPBBL-CC4 (Bortfeldt & Mack 2007) and 3BF+MH (Allen et al. 2011) using the first 10 instances of each BR and BRXL datasets which were described in chapter 2. The second experiment investigates the performance of OH-3BFBL on all instances from the BR and BRXL data sets. The run time limit was set at 160 seconds for comparison with previous published methods. It is not possible to have exactly the same hardware however similar specifications to those reported by Allen et al. (2011) were used.

For each experiment, there were two setups: Single and Mix. For Single setup, there was only one run with SIMPLE block generation mode and the time limit is 160 seconds. For Mix setup, there were two runs. The first run was with SIMPLE block generation

and the second run was with GROUP block generation. The time limit for each run was set at 80 seconds so that total time limit was 160 seconds. The reported result for the second setup was the best result of the two runs. A summary of the experiment setup is as follows:

- OH-3BFBL Single Setting - One Run
  - Block generation mode (blMode) = SIMPLE
  - Maximum number of block are estimated (n) = 70
  - Run time limit (t)= 160 seconds
  - Gap Step (g) = 5
- OH-3BFBL Mix Setting - Two Runs
  - First Run Setting:
    - Block generation mode (blMode) = SIMPLE
    - Maximum number of block are estimated (n) = 70
    - Run time limit (t) = 80 seconds
    - Gap Step (g) = 5
  - Second Run Setting:
    - Block generation mode (blMode) = GROUP
    - Maximum number of block are estimated (n) = 70
    - Run time limit (t) = 80 seconds
    - Gap Step (g) = 5

Test	TSACC-4P Bortfeldt (1999)	SPBBL-CC4 Bortfeldt & Mack (2007)	3BF+TS Allen et al. (2011)	OH-3BFBL Single	OH-3BFBL Mix
BR1	**92.3**	87.3	90.0	91.2	91.7
BR2	**93.5**	88.6	89.6	91.7	92.7
BR3	**92.3**	89.4	89.0	91.3	92.2
BR4	90.8	90.1	88.8	91.2	**91.6**
BR5	89.9	89.3	88.5	90.9	**91.6**
BR6	89.2	89.7	88.6	90.8	**91.3**
BR7	87.1	89.2	88.7	90.8	**91.1**
BR8	84.0	87.9	88.3	**90.0**	**90.0**
BR9	80.9	87.3	87.9	**89.7**	89.6
BR10	79.1	87.6	87.9	**89.4**	89.0
AVERAGE	87.9	88.6	88.7	90.7	**91.08**

Table 3.1: OH3BFBL results compared to TSACC-4P, SPBBL-CC4 and 3BF+TS with first 10 instances of BR dataset.

3.4.2 Experimental Results

3.4.2.1 Result Evaluation BR and BRXL - 10 Instances

The experimental results of the BR data set are shown in table 3.1 and table 3.2 and the BRXL data set result is shown in table 3.3. Table 3.2 shows a comparison between the performance of OH-3BFBL and TSACC-4P, SPBBL-CC4 and 3BF+TS. The results using OH-3BFBL are competitive compared to other published works within the same run time limit. The OH-3BFBL Mix set up has the highest utilisation for BR4 - BR8. OH-3BFBL with Single setting gives the best result for BR8 - BR10. Across BR1 - BR10, OH-3BFBL Mix has the highest average utilisation compared to other approaches including OH-3BFBL with Single setup. Allen et al. (2011) is the published approach with the highest utilisation. A paired t-test was performed between 3BF+TS and OH-3BFBL Mix setup. This shows that the result using OH-3BFBL is significantly different to that using 3BF+TS at a 95% confidence interval.

Table 3.2 shows the utilisation, standard deviation (STDDEV) and actual run time of Single and Mix setups for the BR data set. The Mix setup performs better for BR1 - BR8 which are weakly heterogeneous test cases. The Single setup performs better in strongly heterogeneous cases. The reported run time for the Mix setup is the total run time from the two runs. The actual run time of both setups is significantly less than

Test	Single Setup Utilisation (%)	Single Setup STDDEV	Single Setup Run Time (s)	Mix Setup Utilisation (%)	Mix Setup STDDEV	Mix Setup Run Time (s)
BR 1	91.2	3.1275	8	**91.7**	0.3601	22
BR 2	91.7	1.4371	12	**92.7**	0.1586	36
BR 3	91.3	1.4789	24	**92.2**	0.1130	50
BR 4	91.2	1.0354	24	**91.6**	0.5941	60
BR 5	90.9	1.3067	27	**91.6**	1.3603	71
BR 6	90.8	0.6229	31	**91.3**	0.7521	96
BR 7	90.8	0.7722	48	**91.1**	0.8927	114
BR 8	**90.0**	0.8789	76	**90.0**	0.5844	152
BR 9	**89.7**	0.5675	102	89.6	0.6536	156
BR 10	**89.4**	0.9575	131	89.0	1.0263	160
AVERAGE	90.7	1.21846	48.3	**91.08**	0.64952	91.7

Table 3.2: Utilisation, standard deviation and run time for OH-3BFBL using the first 10 instances of the BR data set

Test	SPBBL-CC Bortfeldt & Mack (2007)	3BF+TS Allen et al. (2011)	OH-3BFBL Single	OH-3BFBL Single STDDEV	OH-3BFBL Mix	OH-3BFBL Mix STDDEV
BRXL1	86.9	92.4	**96.4**	0.8911	96.0	1.5263
BRXL 2	88.3	92.4	95.3	1.2048	**95.4**	1.3996
BRXL 3	89.8	91.9	93.7	2.7836	**94.6**	1.3779
BRXL 4	90.2	92.1	93.6	1.2779	**94.9**	0.6923
BRXL 5	89.9	92.5	92.6	0.9739	**94.2**	1.2594
BRXL 6	91.5	92.6	92.7	0.8820	**94.5**	0.9195
BRXL 7	91.0	92.6	92.9	1.1905	**94.2**	0.8082
BRXL 8	90.8	92.8	93.1	0.8624	**94.4**	0.6205
BRXL 9	90.9	92.3	93.6	0.8639	**94.9**	0.5757
BRXL 10	90.4	92.7	93.6	0.7822	**94.8**	0.7606
AVERAGE	90.0	92.4	93.63	1.1713	**94.79**	0.9940

Table 3.3: Results of OH-3BFBL compared to SPBBL-CC4 and 3BF+TS using the first 10 instances of the BRXL data set

the time limit of 160 seconds for most of cases except for the Mix setup in the stronger heterogenous cases.

The result for the first 10 instances from the BRXL data set is shown in table 3.3. For this data set, there is no published result for TSACC-4P. Both setups of OH-3BFBL produced a significant improvement compared to SPBBL-CC4 and 3BF-TS. OH-3BFBL Mix setup dominates here with the exception of BRXL1 where OH-3BFBL Single has slightly higher utilisation. There is no run time reported in the BRXL data set as OH-3BFBL always used the entire allocated time.

3.4.2.2 Result Evaluation BR and BRXL - 100 Instances

In this section, the performance of OH-3BFBL is presented using all instances from the BR and BRXL data sets. Results for the BR data set are shown in table 3.4 and for the

BRXL data set results are shown in table 3.5. For both data sets, OH-3BFBL with both setups show consistent results with the larger number of test instances. The performance using the BR data set shows a similar trend to previous experiments. OH-3BFBL Mix setup produces better results in weakly heterogeneous cases. However, Single setup performs better in stronger heterogeneous cases. In the BRXL data set, OH-3BFBL Mix setup gives better results in all cases and the Single setup has the best result only in BRXL1. One reason for the Mix setup performing better in weakly heterogeneous cases is that these cases require a shorter run time to complete the packing at each iteration. Single setup performs only one run well within the time limit and did not use any of the remaining time. However, Mix setup has one additional run with different settings and produces improved utilisation. However, in the weakly heterogeneous cases, if grouped blocks are generated with large internal loss, it is better to use the simple block only. In stronger heterogeneous cases, the Single setup utilises more of the allocated run time and finds a better block at the end of the packing. On the other hand, overhead estimation might have to stop in the middle of packing in the Mix setup. As shown in the above experiments, OH-3BFBL Mix has a better average result in both the BR and BRXL data sets. The performance of Mix setup was investigated in more detail. From figure 3.9, BR1 has the widest range of results. This is due to BR1 has the least number of boxes in the BR data set. Therefore, there are fewer number of different box dimensions available and it might not be possible to fit the current packing plan. When the number of box types is increased from BR1 to BR10, there are better chances to find a suitable box for the selected gap. In addition, the average utilisation was also decreased because of the increased complexity from BR1 - BR10. In contrast, OH-3BFBL has more consistent results in the BRXL data set. The average result using the BRXL data sets are shown in figure 3.10. The average result from BRXL1 to BRXL10 is not widely varied and there is no decreasing trend from BRXL1 to BRXL10. Standard deviation is smaller compared to the BR data set showing consistency of OH-3BFBL with the BRXL data set.

Test	Single Utilisation	Single STDDEV	Single Run Time	Mix Utilisation	Mix STDDEV	Mix Run Time
BR 1	92.2	2.6345	14	**92.5**	2.6978	17
BR 2	91.8	1.8334	16	**92.5**	1.7962	21
BR 3	91.7	1.5378	18	**92.2**	1.3461	30
BR 4	91.3	1.4482	22	**91.9**	1.2431	32
BR 5	91.0	1.2365	26	**91.6**	1.1581	37
BR 6	90.9	1.0995	33	**91.4**	0.9576	47
BR 7	90.8	0.9806	48	**91.0**	0.9207	61
BR 8	90.0	0.9408	81	**90.1**	0.8489	71
BR 9	**89.6**	0.9543	108	89.4	0.8164	75
BR 10	**89.3**	0.8496	143	88.9	0.7918	80
AVERAGE	90.86	1.35152	50.9	**91.15**	1.2577	47.2

Table 3.4: Results of OH-3BFBL with all instances of BR dataset

Test	Single Utilisation	Single STDDEV	Mix Utilisation	Mix STDDEV
BRXL1	95.6	1.358	**95.6**	1.460
BRXL 2	94.6	2.277	**95.2**	1.361
BRXL 3	94.1	1.889	**94.7**	1.515
BRXL 4	93.8	1.549	**94.4**	1.433
BRXL 5	93.2	1.480	**94.4**	1.193
BRXL 6	92.6	1.902	**94.2**	1.072
BRXL 7	92.5	1.420	**94.1**	1.006
BRXL 8	92.9	1.261	**94.5**	1.019
BRXL 9	93.4	0.872	**94.7**	0.598
BRXL 10	93.6	0.672	**94.9**	0.558
AVERAGE	93.63	1.4681	**94.67**	1.1214

Table 3.5: Results of OH-3BFBL with all instances of BRXL dataset

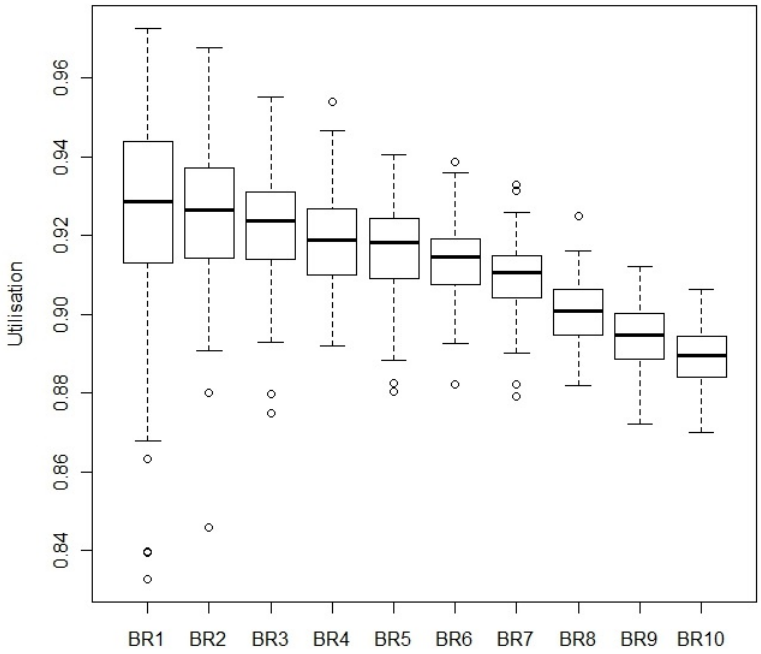


Figure 3.9: Variation of the performance of OH-3BFBL Mix set up from BR1 to BR10

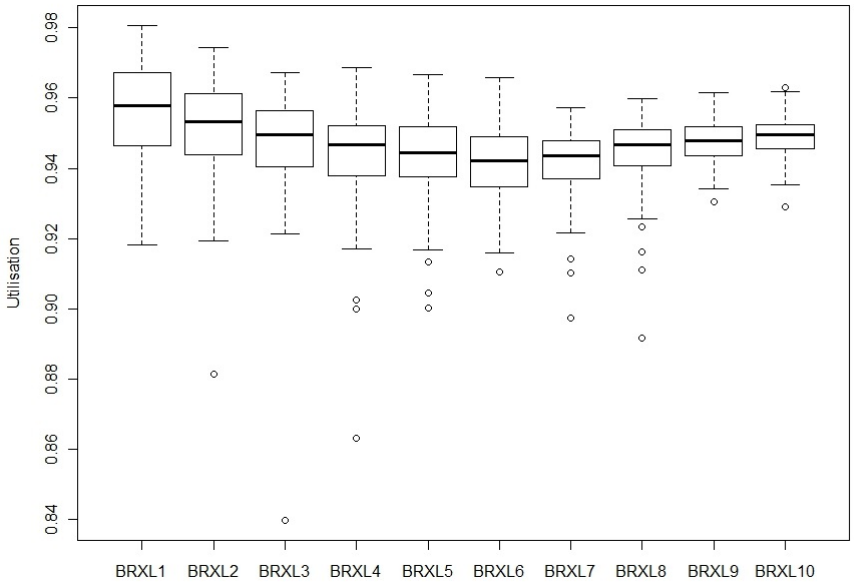


Figure 3.10: Variation of the performance of OH-3BFBL Mix setup from BRXL1 to BRXL10

3.5 Further Modification to OH-3BFBL

In this section, we highlight some observations from the previous experiments and modifications to OH-3BFBL. The first observation is that Group block contains only two box types. However, it is possible to group more than two box types to form a bigger block. In Group block generation, a third block type is not included even if there is no internal loss. The second observation is about an issue found in Extreme Point (Crainic et al. Summer 2008) which is used to present the packing position in 3BF and 3BFBL. The issue is that Extreme Point does not include all possible packing positions. As shown in figure 3.11, the black dots represent the Extreme Point generated for a packed box. However, it is also possible to pack a block in a position represented by a white dot. The original Extreme Point approach identifies the intersection between the projection from the packed block to either the container side or other blocks. Point generation using Extreme Point encourages packing boxes towards the deepest bottom left corner of the container. However, this can result in some wasted space which can be reduced by using the reallocation procedure described in section 3.2.2. Figure 3.12 and figure 3.13 shows other possible locations of the same block at the deepest layer in both 2D and 3D. In figure 3.13 case B represents shifting across the x-axis and case C represents shifting up and across z-axis. In order to address the above issues, we propose a Multi-type Block generation in section 3.5.1 and Layer Point in section 3.5.2.

3.5.1 Multi-type Block

In this section, we propose a Multi-type block to generate blocks with more than just two box types. Similar to the previous Group block generation, there are two types of arrangements: ABOVE and NEXT. A sample of various Multi-type blocks is shown in figure 3.14.

The main idea of a Multi-type block is the combination of two existing blocks together. The combination can be between Simple or Group block to allow more than two box types

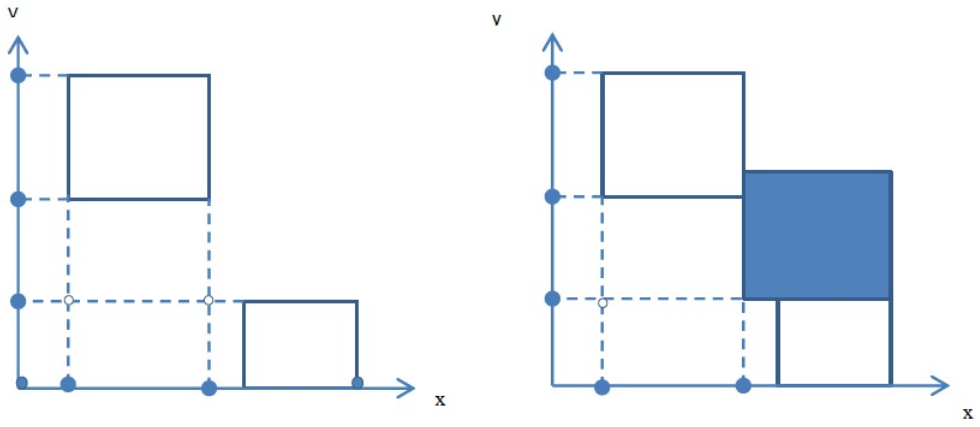


Figure 3.11: A example of Extreme Point (black dots) and Layer Point (white dots)

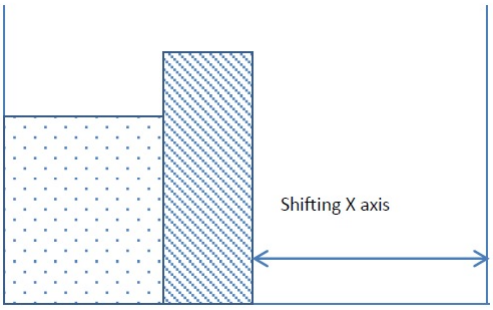


Figure 3.12: Possible Non-Extreme Point position in two-dimensions

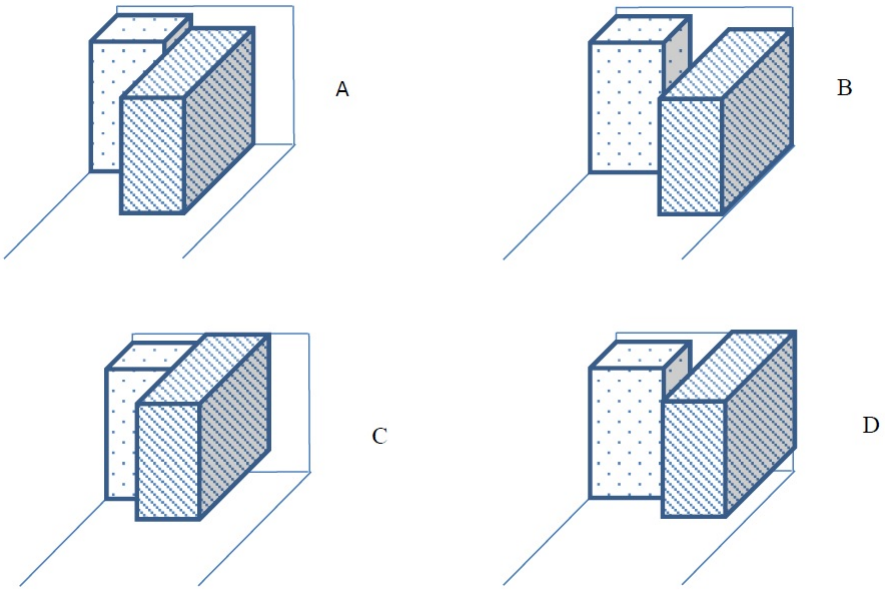


Figure 3.13: Possible Non Extreme Point Position in three dimensions

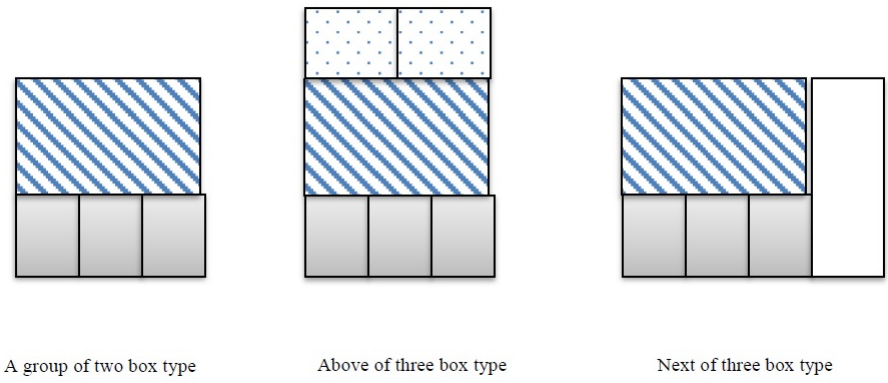


Figure 3.14: A sample of various Multi-type blocks which have more than two box types

in a single combined block. The conditions for a valid combination between two blocks are:

- Volume utilisation has to be greater or equal to α.
- Volume utilisation is calculated as:
  - $\frac{\sum BoxVolumne}{EnvelopeBoxVolume} \geq \alpha$ where Box Volume is the sum of the volume of block A and block B. Envelope Box is the minimum rectangular box that contains the Multi-type block.
- Boxes required to form blocks A and B are a subset of input boxes.
- For "Next" arrangement:
  - blockA.length $\geq$ blockB.length
  - blockA.height $\geq$ blockB.height
  - blockB is placed in point (blockA.width, 0, 0) relative to the position of blockA
- For "Above" arrangement:
  - blockA.length $\geq$ blockB.length
  - blockA.width $\geq$ blockB.width
  - blockB is placed in point (0, 0, blockA.height) relative to the position of blockA

The pseudo code for block generation is shown in Algorithm 3.4. The grouping process starts with the generation of Simple blocks from single box type. Multi-type block generation continues until a number of individual blocks are generated or there are no more blocks available. Process GenerateAboveBlock and GenerateNextBlock will find valid combinations between two generated blocks in BL with ABOVE and NEXT arrangement and return all valid blocks. During block generation it is possible to combine

boxes in different arrangements however the blocks generated can potentially have the same dimension. This is termed as ambiguous blocks. Ambiguous blocks are blocks with the same dimensions, the same number of box types and the same quantity of each box type inside the block. An example of ambiguous block is shown in figure 3.15. During the block generation, only one of the ambiguous blocks is added to the output.

Algorithm 3.4 Multi Block Generation

Input: : set of boxes to pack, B and block number limit, n
Output: : set of block BL
1: $BL \leftarrow \emptyset$
2: $BL \leftarrow BL + GenerateSimpleBlock(B)$
3: **while** $BL.Count < n$ and $nomoreblockcreated$ **do**
4: $BL \leftarrow BL + GenerateAboveBlock(BL)$
5: $BL \leftarrow BL + GenerateNextBlock(BL)$
6: **end whilereturn** BL

3.5.2 Layer Point

Layer Point is proposed to determine points which are not available using Extreme Point. The idea of Layer Point generation is to create horizontal and vertical projections from the block at the layer being considered. The pseudo code for Layer point is shown in Algorithm 3.5. At the beginning of packing, there is no packed box and no layer point. When considering the deepest gap, all packed blocks intersecting with the layer are selected. From the selected block, the horizontal and vertical lines are projected along the y-axis and x-axis on layer Z. The intersection between vertical and horizontal lines are layer points using the getIntersection process in Algorithm 3.5 line 12.

When a block is packed, the deepest surface of a block has 4 corners represented as bottom left (BL), bottom right (BR), top right(TR) and top left (TL) points as shown in figure 3.16. Normal Extreme Point method packs deepest bottom left most point of block – BL point into the candidate position. In Layer Point generation, each of the categories is in correspondence with each corner of the block. Informally, each category

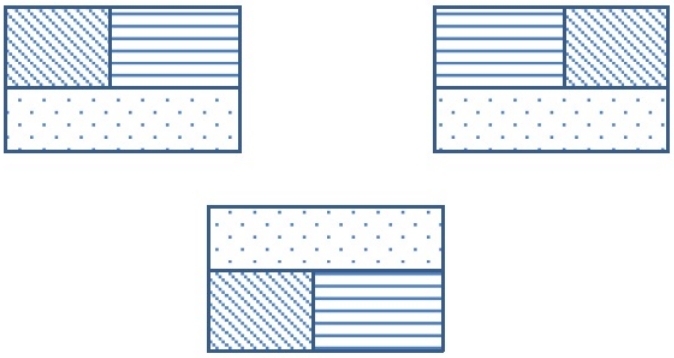


Figure 3.15: Example of blocks with the same dimensions but different arrangement - Ambiguous block

Algorithm 3.5 Generate Layer Point Procedure

Input: : packing plan, P and current layer, Z
Output: : set of layer point, LP

1: $LP \leftarrow \emptyset$
2: $hLine \leftarrow \emptyset$
3: $vLine \leftarrow \emptyset$
4: **for all** block bl and its position p in P **do**
5: **if** bl intersect Z **then**
6: vLine.add(project line from point (p.x, p.y, Z) across y axis
7: vLine.add(project line from point (p.x + bl.xSize(), p.y, Z) across y axis
8: hLine.add(project line from point (p.x, p.y, Z) across x axis
9: hLine.add(project line from point (p.x, p.y+bl. ySize, Z) across x axis
10: **end if**
11: **end for**
12: $LP \leftarrow GetIntersection(vLine, hLine)$
return LP

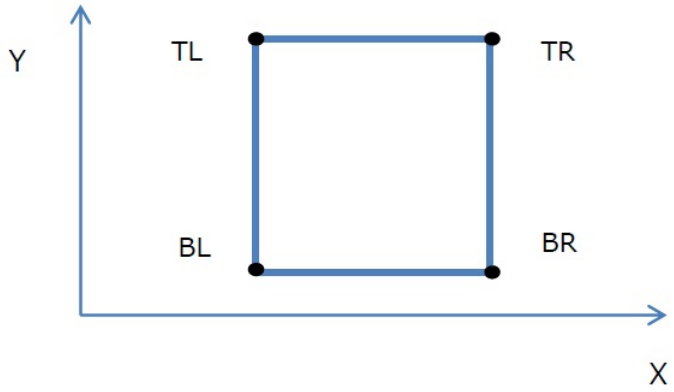


Figure 3.16: An example of multiple layer point

is the bottom left most candidate point when rotating the coordinate around the y axis. For example: top left candidate point is a bottom left candidate point if the container is rotated 90 degrees anti-clockwise around the y axis. When considering a block, all point categories are considered. The block's bottom left corner will be packed in the point from the bottom left most candidate point list. The block's top right corner will be packed in the point from the top right most point list and similar to the others. Instead of generating only bottom left most layer point, there is generation of the layer point for each category. Each candidate point can be generated using the similar method. At the beginning, there is one point (0, 0, 0) in bottom left candidate point. When at least one box is packed, bottom left candidate point contain the extreme point generated by the "Extreme Point" method in Crainic et al. (Summer 2008). Bottom left candidate point also contained in the layer points using generate method in section 3.5.2.

3.6 Further Modification Experiment

3.6.1 Experimental Setup

In order to have a fair comparison, the experiment setting is kept the same as in section 3.4.1, i.e. number of blocks to estimate and run time. However, there is an additional

Test	OH-3BFBL Single Mode (%)	OH-3BFBL Mix mode (%)	OH-3BFBLEX (%)
BR1	92.2	**92.5**	89.6
BR2	91.8	**92.5**	90.8
BR3	91.7	**92.2**	91.3
BR4	91.3	**91.9**	90.9
BR5	91.0	**91.6**	91.1
BR6	90.9	**91.4**	91.1
BR7	90.8	**91.0**	90.8
BR8	90.0	90.1	**90.6**
BR9	89.6	89.4	**90.3**
BR10	89.3	88.9	**90.0**

Table 3.6: Result of OH-3BFBL with Multi-type block generation and layer point

setting: Limit of block generation where n is 10,000. The hardware setting is identical to the previous experiment.

3.6.2 Experimental Results

Table 3.6 shows that Multi-type block generation and Layer Point both performed better in strong heterogeneous cases. However, in the other cases, the modification did not perform well. The reasons are: firstly, in weakly heterogeneous cases, Multi-type block generation creates bigger blocks using more boxes, which results in fewer available boxes at the end of the packing to choose from. This also limits any positive benefits of overhead estimation. A second reason is that overhead estimation has to consider more block and position combinations. It is possible that there are many different positions and block pairs which have no difference to the final estimated result. Figure 3.17 shows an example where there is no difference between all four positions and block combinations.

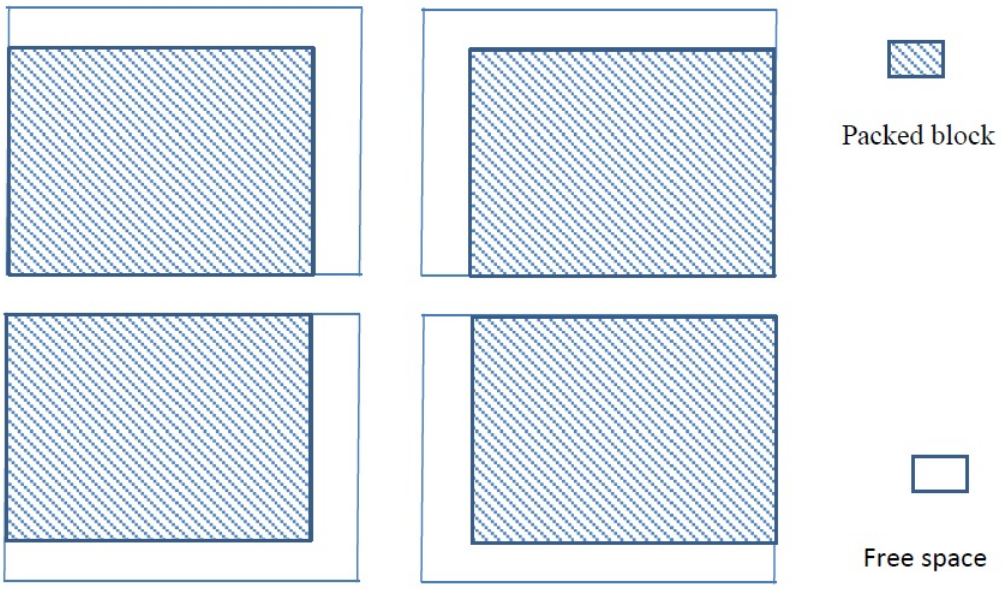


Figure 3.17: Example of different positions and point combinations where free space cannot be re-used

3.7 Conclusion

In this chapter, the three-dimensional strip packing problem without stability constraints was investigated. We introduced a modified packing heuristic 3BFBL with two new processes: Block generation and Overhead estimation used in combination with the best-fit methodology. Block generation has two types of combinations, Simple block and Group block, and is advantageous when combining small boxes into large blocks thereby minimising internal loss inside the block. In order to improve the heuristic result, an overhead estimation procedure was developed and used with 3BFBL. That overhead estimation avoids large blocks which can create wasted space by packing the block which has the best estimated outcome.

The proposed OH-3BFBL shows an improved performance in stronger heterogeneous cases when compared to methods in the literature and can also obtain a better average utilisation. Two setups were presented. The simple setup showed a slightly better per-

formance on instances with a high number of box types and low quantity of each type. However, in data sets with the same number of box types and increased quantity of each type, as in the BRXL data set, the Mix setup shows a better performance than the Single set up.

Further changes were introduced, including Multi-type block and Layer Position, both of which improved the performance of OH-3BFBL in strong heterogeneous instances. Multi-type block allowed the combination of more than two types of boxes together. Layer Point modification allowed more possible packing positions. These changes lead to slightly lower results in weak heterogeneous instances but improved performance in strong heterogeneous cases. For weaker heterogeneous cases, Multi-type block generation allowed more internal loss which was repeated during the packing process and lead to low utilisation. On the other hand, Multi-type block generation was found to be more suitable in strong heterogeneous instances where low box quantity does not allow to repetition of internal loss.

4 Overhead Estimation and Constructive Heuristics For The 3D-SPP with a Stability Constraint

In this chapter, heuristics and overhead estimation for the 3D-SPP with the addition of a stability constraint is investigated. Stability constraint has been mentioned in the literature as one of the key issues in the container loading problem (Bischoff & Ratcliff (1995)). An example of this can be seen in the previous chapter where boxes are allowed to overhang in the air to allow maximum free space. This is obviously not applicable in real life operations. During collaborative work with 3T Logistics Ltd, this constraint has come up on numerous occasions as a compulsory requirement in transport planning. This requirement ensures compact and stable packing suitable for transportation. It is necessary to mention that this stability is not important in every packing case. However in this context and for related work with a business, it is an important aspect in determining the feasibility of an automated packing technique in a business operation. Previous research has been carried out for the 3D-SPP including the stability constraint. From the previous chapter, it is known that the best fit methodology performs well in the 3D-SPP without a stability constraint. However, to the best of our knowledge, there is no study about the best fit performance with a stability constraint. The 3D-SPP with the addition of a stability constraint is defined in section 4.1. In section 4.2, the performance of a range of heuristics for the 3D-SPP with a stability constraint is investigated. Based

on the results of section 4.2, the performance of overhead estimation in the 3D-SPP with a stability constraint was evaluated.

4.1 3D-SPP with Rotation and Stability Constraint

In this chapter, the 3D-SPP with a stability constraint is defined. There are different definitions for a stability constraint. For example, the bottom area of all items or boxes must be supported by either the container or other boxes, which are fully supported. Another definition is a percentage of the bottom area of a box is supported (partially supported). Yet another one is only the centre of gravity of a box is supported. In this work, only the fully supported stability constraint case is investigated. That is, a box is supported when the entire bottom surface is completely in contact with either other boxes or the container. Then, the 3D-SPP with a stability constraint tackled here can be defined as follows:

- Input:
  - A set of rectangular boxes with given dimensions and rotation ability.
  - A container with fixed width and height, but infinite length.
- Output:
  - A packing plan showing the position of each box in the container.
- Objective:
  - Minimise the length of the container required to pack all boxes.
- Constraints:
  - All boxes need to be placed orthogonally (i.e. the edges of the boxes need to be parallel with the container).
  - All boxes must be packed.

  - Boxes cannot overlap.
  - The bottom surface of each box is in full contact with either other boxes or the container (stability constraint).

The evaluation function for the problem is the same as described in chapter 3.

4.2 Investigation into Best Fit, Best Support Heuristics

4.2.1 Block Generation for Stability Constraint

In the previous chapter, block generation with 3BF heuristics was introduced. In block generation, multi-type blocks allow a small internal loss. However, if there is internal loss then it is possible to violate the stability constraint. An example is shown in figure 4.1 where block C is not completely supported when placed on top of a group block containing A and B which has a small internal loss. In this section, the rules for the block generation process are modifed to allow a small inner loss and so the stability constraint can be met. Similar to the grouped block, there are two types of arrangement which can create a block: Above and Next arrangement. Block generation can be informally described as the combining of block A and block B if the following conditions are met:

- Volume utilisation of the combined block is greater than or equal to 98%.
  - The volume utilisation is calculated as $\frac{\sum BoxVolume}{EnvelopeBoxVolume}$
    - Box Volume is the sum of the volume of boxes of block A and block B.
    - Envelope box is the minimum rectangular box that contain all the blocks.
- Boxes required to form block A and block B is a subset of input boxes.
- For "Next" arrangement:
  - blockA.length $\geq$ blockB.length

  - blockA.height $\geq$ blockB.height
  - block B is placed in point (blockA.width, 0, 0) relative to position of block A

- For "Above" arrangement:
  - blockA.length $\geq$ blockB.length
  - blockA.width $\geq$ blockB.width
  - block B is placed in point (0, 0, blockA.height) relative to position of block A
  - block B is fully supported by block A

- Block's top surface utilisation is greater than or equal 98%
  - The utilisation of the top surface of the block is calculated as $\frac{\sum TopSurface}{EnvelopeBoxTopSurface}$
    - TopSurface is the area of all boxes in which the top surface is as high as the envelope box top surface.
    - EnvelopeBoxTopSurface is the area of the top surface of envelope box.

The new condition of top surface utilisation ensures that the combined block can be used as a base for other blocks. This avoids forming any horizontal tower cases during the packing process. Where blocks have a small internal loss and small top surface, subsequent blocks to be placed on top have to have a smaller bottom surface. The limit for volume utilisation and top surface utilisation can be changed. However after rigorous initial experiments, it was found that 98% utilisation is more likely to produce a better result.

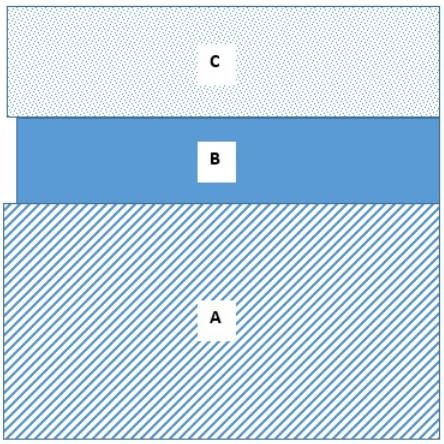


Figure 4.1: Non-supported Block

4.2.2 Best Fit and Best Support Heuristics

In chapter 3, it was shown that the performance of overhead estimation depended on the heuristic used in completing a partial packing plan. For the 3D-SPP, a number of heuristics were evaluated and the 3BFBL heuristic had the best performance. Therefore, a range of heuristics is presented and their performance with an additional stability constraint is evaluated. The best heuristic will be used with overhead estimation to further improve performance. There are two types of heuristic which are considered: Best Fit and Best Supported. The main idea of the best fit heuristic is to prefer blocks which fill most of the gap. The best fit selection is comparable to 3BFBL heuristics which was introduced in chapter 3 when blocks with a larger XZ surfaces is preferred. If there is more than one block that has a similar XZ surface area then secondary selection is applied (i.e. contact area). The best-support heuristic prefers a block with the largest bottom area. This encourages subsequent boxes to be packed on top and increase stability. Where there is more than one block available to pack into the deepest gap, the block with the

largest XY surface is preferred. If there is more than one block that has a similar XY surface area then a secondary selection is applied. The pseudo code for both heuristics shown in Algorithm 4.1.

Algorithm 4.1 Best Fit/ Support Heuristic-3BFS

Input: : container, C and set of boxes to pack B, set of generated block, BL
Output: : packing plan, P with all blocks and its position

```
1: P ← ∅
2: while |B| >0 do
3:     G ← GetLowestGap(C, P)
4:     bestF/S ← selectBestFit/SupportPair(G, BL)
5:     p ← selectPairSecondaryCriteria(bestF/S)
6:     if p is found then
7:         Reallocation(p)
8:         Add p to P
9:         Remove box in p from B
10:    else
11:        Mark all marks in G is invalid for future use
12:    end if
13: end while
```

For each iteration, both heuristics start by selecting the deepest gap in the container. There are two possible cases:

- If there is a valid block to fit in the deepest gap, then all possible pairs of blocks and their position are evaluated. A number of blocks and positions are selected using best-fit or best-support criteria and stored in bestF/S (Algorithm 4.1 line 4). A secondary criteria selection is used to select one block and position in bestF/S. The selected pair will be packed into the container.

- If there is no valid block then the deepest gap will be discarded and the next deepest gap is selected.

In the original 3BF, only the single best fit block was considered. In order to improve the flexibility of best-fit and best-support heuristics, a best fit/support threshold k is introduced. For example, with best fit selection instead of considering one or more valid

blocks with the largest XZ area, N, any blocks with an XZ surface greater than or equal to k*N are considered for secondary selection. A similar rule is applied for the best-support heuristic. The value of k is specified in experimental set up.

The secondary policies used in best-fit and best-support heuristics are four tie breakers from Allen et al. (2011) Maximum Contact, Maximum XZ Surround Score, Maximum Y, Minimum Y and one new tie breaker Maximum XY Surround Score.

- Maximum Contact: The block with the largest area contacting either of the other boxes or the container is selected.
- Maximum XZ Surround Score: The block has the largest number of boxes which have Y co-ordinate equal or less is selected. This selection encourages option which form a flat XZ surface. This creates a bigger gap for future blocks.
- Maximum Y: The block placed in deepest bottom left most position and the furthest point in Y - axis
- Minimum Y: The block placed in deepest bottom left most position and the deepest point in Y - axis
- Maximum XY Surround Score: Block has the largest number of boxes which have deeper XY surface. This selection encourgages forming a flat XY surface. This offers the opportunity for future boxes to be packed on top.

Second criteria selection are used in Algorithm 4.1 line 5.

4.2.3 Experiments with Heuristics

One experiment has been carried out to investigate the performance of the selected heuristics. There are 10 combinations of best fit and best-support heuristics with secondary criteria selection. The algorithms are implemented in Java single thread code. The experiments were run on a PC with AMD 2.0 GHz and 1 G memory. There are two runs

Heuristic	Utilisation (%)
Best Fit - Maximum Contact - 0.5	86.7
Best Support - Maximum Contact - 0.5	86.4
Best Fit - Maximum Contact - 0.8	85.2
Best Fit - Maximum XZ Surround Score - 0.5	85.2
Best Fit - Maximum XY Surround Score - 0.5	84.5
Best Fit - Maximum Y - 0.8	84.5
Best Fit - Maximum XZ Surround Score - 0.8	84.3
Best Fit - Maximum XY Surround Score - 0.8	84.2
Best Support - Maximum XZ Surround Score - 0.5	84.1
Best Support - Maximum Y - 0.5	83.8
Best Support - Maximum XY Surround Score - 0.5	83.7
Best Fit - Maximum Y - 0.5	83.2
Best Support - Maximum Contact - 0.8	82.4
Best Support - Maximum XZ Surround Score - 0.8	82.3
Best Support - Maximum Y - 0.8	82.0
Best Support - Maximum XY Surround Score - 0.8	81.6
Best Fit - Minimum Y - 0.8	81.2
Best Support - Minimum Y - 0.8	77.9
Best Fit - Minimum Y - 0.5	76.2

Table 4.1: Performance of best fit/support heuristics with BR data set

for each heuristic. Each run is performed with a different value for the best fit/support parameter - k. The value of k was set to 50% and 80%. The average overall results for 1000 instances of BR1-BR10 are shown in table 4.1.

Table result in the table 4.1 is ordered by decreasing utilisation. From these result it is important to note the following:

- Best Fit - Maximum Contact - 0.5 heuristic with a best fit threshold - k =50% and Maximum Contact is used as secondary criteria selection has the highest average utilisation.

- The best fit heuristic outperforms the best-support heuristic. One of the reasons for this is that the current block generation only allows a "layer" block where the number of blocks across the y-axis is only 1 and this limits the best-support heuristic.

- The Maximum Contact tie-break policy gives the all best three average results. Therefore, Maximum Contact criteria can be useful for the 3D-SPP with a rotation and stability constraint.
- The majority of the best results were produced when the best fit/support parameter, k=50%.
- Best fit and support heuristics gives a lower average result compared to results from TSACC and TSACC-4P.

4.3 Overhead Estimation with Stability Constraint

As the results from section 4.2.3 suggest, the Best Fit - Maximum Contact heuristic where k = 50% gives the best performance. Therefore this heuristic has been selected for be used with overhead estimation. From now on this heuristic is referred to as OH-3BFMC. The pseudo code for OH-3BFMC is shown in Algorithm 4.2. Similar to OH-3BFBL, OH-3BFMC uses blocks instead of single boxes which was introduced in section 4.2.1.

Different to OH-3BFS instead of the selectBestFit/SupportPair procedure, OH-3BFMC does not use a heuristic to pack a number of blocks after each estimation. This is due to optimised implementation in the Java application which allows a faster estimation. Another key difference is instead of using the OverheadEstimation procedure, OH-3BFMC uses OverheadEstimation3BFMC and this is presented in Algorithm 4.3. At each iteration, OH-3BFMC selects the deepest gap. There are two possible cases:

- If there are valid blocks then all possible pairs of blocks and their positions are evaluated.
- If there is no valid block then the current gap will be discarded and the next deepest gap is selected.

For each iteration, OH-3BFMC packs a block until there are no boxes left or the time limit has been reached and the complete packing plan is returned. The pseudo code for procedure OverheadEstimation3BFMC is shown in Algorithm 4.3. OverheadEstimation3BFMC starts by selecting a number of best fit blocks and their positions, bestFitBL. How the number of blocks and position is selected is described in the experimental set up. Each selected possibility will be used in the current plan and a complete packing plan is produced. Different to the overhead estimation procedure in the previous chapter, 3BFMC has been used to complete the packing plan (Algorithm 4.3 line 7). From the completed packing plan, a block and its position which has the highest utilisation will be selected.

Algorithm 4.2 3BFMC With Overhead Estimation(OH-3BFMC)

Input: : container, C set of boxes to pack B and set of generated block, BL
Output: : packing plan, P with all blocks and its position

```
1: P ← ∅
2: while |B| >0 and time < t do
3:     G ← GetLowestGap(C, P)
4:     p ← OverheadEstimation3BFMC(G, BL, P)
5:     if p is found then
6:         Add p to P
7:         Remove box in p from B
8:     else
9:         Mark all marks in G is invalid for future use
10:    end if
11: end while
```

4.3.1 Experimental

4.3.1.1 Experimental Set Up

3BFMC and OH-3BFMC are implemented using Java single thread code. Experiments are run on a PC with AMD 2.0 GHz and 1 G memory. CrateViewer application is used to visualise the solution (Allen et al. 2011). The parameters for OH-3BFMC are as follows: number of blocks used to estimate is 130; time limit is 294 seconds. To

Algorithm 4.3 OverheadEstimation3BFMC procedure

Input: : A gap, G and set of block, BL
Output: : a block and position, p

```
1: bestFitBL ← GetBestFitBlock(G, BL)
2: bestS ← 0
3: bestP ← nil
4: for all p in bestFitBL do
5:     P' ← Copy(P)
6:     pack p in packing plan P'
7:     score ← CompletePackingUsing3BFMC(P')
8:     if score > s then
9:         bestS ← score
10:        bestP ← p
11:    end if
12: end forreturn bestP
```

investigate the performance of OH-3BFMC, two experiments were carried out. The first experiment compares the performance of OH-3BFMC with published results from the literature. The experimental data set contains the first 10 instances from the BR data set. There are two comparisons of OH-3BFMC. In the first comparison, the result using OH-3BFMC is compared with the best known approaches in the literature TSACC and GACC. This is a direct comparison between OH-3BFMC with non-parallel methods. The second comparison is between the parallel methods GACC-4P, TSACC-4P and OH-3BFMC. The second experiment is an investigation into the performance of OH-3BFMC in a wider range of instances. OH-3BFMC is tested with all 100 instances in each BR sub-data set. The reported result is the average of 100 instances of each BR data set.

4.3.1.2 Experimental Results

BR data set - first 10 instances: The results of OH-3BFMC with the first 10 instances of each BR data set are shown in table 4.2 and table 4.3. Table 4.2 shows a direct comparison between OH-3BFMC with the non-parallel methods TSACC and GACC. Table 4.3 shows a comparison between OH-3BFMC with the parallel methods TSACC-4P and GACC-4P.

Test	TSACC	GACC	OH-3BFMC
BR1	**92.1**	83.8	87.8
BR2	**92.5**	88.1	89.9
BR3	**90.9**	88.7	89.2
BR4	**89.9**	87.8	88.9
BR5	**89.0**	87.7	88.1
BR6	87.2	87.1	**88.7**
BR7	84.9	85.3	**87.4**
BR8	81.5	83.6	**85.8**
BR9	78.6	81.1	**84.3**
BR10	76.9	78.0	**82.8**
AVERAGE	86.35	85.12	**87.27**

Table 4.2: BR data set - first 10 instances - non-parallel method

The results from Table 4.2 are summarised as follows:

- TSACC shows the best result for the weakly heterogeneous cases BR1-BR5. However OH-3BFMC improves the container utilisation for the stronger heterogeneous cases BR6-BR10.
- On average, over all the BR data set, OH-3BFMC demonstrated the highest utilisation.

From Table 4.3, the results can be summarised as follows:

- TSACC-4P has the highest utilisation for the weakly heterogeneous cases BR1 - BR6. GACC-4P yields the best results for BR7-BR8. However, OH-3BFMC gives the highest utilisation for the stronger heterogeneous cases BR9 - BR10.
- On average, over all the BR data set, the performance of OH-3BFMC is comparable to TSACC-4P and even better when compared with GACC-4P. However, TSACC-4P and GACC-4P used 4 parallel computers and total CPU time is much longer than the time limit of OH-3BFMC.

Test	TSACC-4P	GACC-4P	OH-3BFMC
BR1	**92.3**	84.3	87.8
BR2	**93.5**	88.6	89.9
BR3	**92.3**	89	89.2
BR4	**90.8**	88.5	88.9
BR5	**89.9**	88.1	88.1
BR6	**89.2**	88.7	88.7
BR7	87.1	**87.8**	87.4
BR8	83.9	**85.9**	85.8
BR9	80.9	84.3	**84.3**
BR10	79.1	82.1	**82.8**
AVERAGE	**87.90**	86.73	87.27

Table 4.3: BR data set - first 10 instances - parallel method

BR data set - 100 instances: The performance of OH-3BFMC is shown in table 4.4 with utilisation, standard deviation and run time and seen in Figure 4.2. The results can be summarised as follows:

- Utilisation decreases from BR1 - BR10. However the general trend is that standard variation also decreases from BR1 to BR10 with the exception of BR9.
- The average performance OH-3BFMC over 100 instances is better when compared with only the first 10 instances.
- OH-3BFMC shows consistent results with a larger number of instances. The majority of the results can be found within 1.955% of the average.
- The run time of OH-3BFMC is lower than the limit of 294 seconds especially in weakly heterogeneous cases.

4.4 Conclusion

In this chapter, the three-dimensional strip packing problem with the addition of a stability constraint was investigated. The stability constraint was defined as all bottom

Test	Utilisation (%)	STDDEV	Run Time (seconds)
BR1	89.0	3.42	59
BR2	89.2	2.51	62
BR3	89.5	1.98	89
BR4	89.1	1.80	102
BR5	88.7	1.81	129
BR6	88.6	1.48	164
BR7	87.5	1.46	229
BR8	86.1	1.38	290
BR9	84.1	1.87	297
BR10	82.2	1.84	300
AVERAGE	87.40	1.955	172

Table 4.4: BR data set - 100 instances

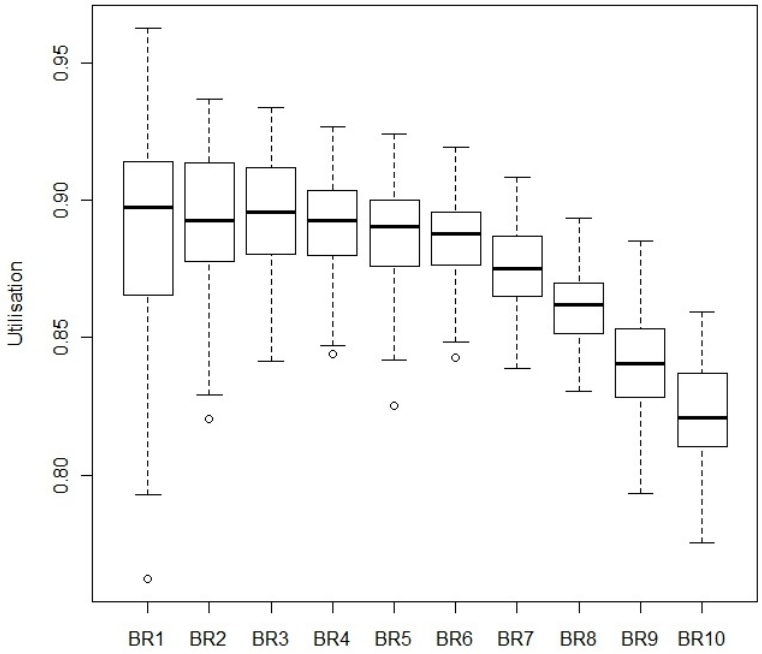


Figure 4.2: OH-3BFMC - BR data set

surfaces of packed boxes must be supported by the container or other boxes. The best-fit heuristic was adapted and best-support heuristics were introduced. The best-heuristic preferred to pack the largest volume first whereas the best-support heuristics preferred a block with the largest bottom surface to encourage stability in further iterations. Different criteria. i.e. Maximum Contact, Minimum Y etc. were adapted and tested as a tie breaker. Different heuristics and tie breakers were combined. The performance of each combination was evaluated to identify the best performance of best fit and maximum contact with stability constraint. The best fit heuristics out performed the selected best-support heuristics and was also compatible with the stability constraint. Out of all the tie breakers Maximum Contact showed consistency and good results in combination with any heuristic. Therefore, overhead estimation was combined with Maximum Contact heuristics (OH-3BFMC). To the best of our knowledge, OH-3BFMC can obtain the best result in strong heterogeneous instances when compared with other well known non-parallel approaches from the literature. OH-3BFMC does not give the best result in weak heterogeneous instances. However, the overall average performance of OH-3BFMC is higher. When compared to the parallel methods, OH-3BFMC shows the highest utilisation for 2 out of 10 BR datasets. When the number of instances is increased, OH-3BFMC shows competitive and consistent utilisation with a lower run time compared to the most well known methods.

5 Heuristics For Pallet Space Equivalent Measurements

This chapter presentes research conducted in collaboration with 3T Logistics Ltd to find an automated approach for a real life packing problem. The problem is described as a Pallet Space Equivalent (PSE) problem. Firstly, the operation of 3T Logistics Ltd and the context of the problem with real life constraints is introduced. This is followed by the definition of the problem in comparison with the three-dimensional strip packing problem. Thirdly, a constructive approach for the problem is presented. Finally, real life data sets and evaluation of the performance of the heuristics are presented.

5.1 Introduction and Operation Overview

3T Logistics Ltd (3T) is a UK-based Fourth Party Logistics (4PL) provider. 3T provides transportation management services to a wide range of clients via their multi-mode transportation management system - SOLO. When 3T Logistics receives customer orders requiring delivery, the orders are processed, consolidated and assigned to a delivery plan. The company then arranges collection and delivery of goods from the customer's site to their consignees via a network of carriers. At the warehouse, products are packed into different types of handling units. 3T often receives the handling unit detail from its customers and uses this information in the transport planning process. The most common method is to palletise items although items can also be stored in bags or boxes. Depend-

ing on the properties of an item, different handling units have different requirements for rotation or stackability. One critical task is to calculate the cost of delivery. There are many mechanisms for calculating delivery costs and it also depends on contracts with carriers. The preferred and most common mechanism is based on the required vehicle footprint. To measure the vehicle footprint, the pallet space is specified as a unit of measurement. Therefore correctly estimating pallet space equivalent is very important not only to 3T but also for customers and carriers. A pallet space is defined as the floor space of a standard pallet. There are different standard pallet dimensions, for example Euro or UK standard pallets. The shipper will typically pack products onto Euro (800 x 1200 mm) or UK (1000 x 1200 mm) standard pallets. An example of a Euro pallet can be seen in Figure 5.1. Handling units are classified as stackable or non-stackable. An example of a stackable pallet is shown in Figure 5.2 and a non-stackable pallet is shown in Figure 5.3.

Figure 5.1: A 800mm x 1200mm Euro standard pPallet

Figure 5.2: An example of a stackable pallet

Figure 5.3: An example of a non-stackable pallet

There are operational requirements which have a significant effect on the day-to-day operations of 3T. Firstly an order for delivery can be changed at the last minute and 3T has to react quickly with high frequency of change. Secondly, actual physical packing is not carried out by the staff at 3T and therefore the packing plan should be "easy" and "good enough" to replicate. Thirdly, customers can have a single order which is larger than the standard vehicle capacity therefore a larger vehicle should be ordered. Vehicle size is not specified until the planning process is completed. Therefore there are no known fixed container dimensions at the point of receiving an order.

Currently, 3T use an in-house method to estimate the pallet space equivalent required for each order. Due to commercial interests, details of the current method cannot be published. The original method used by 3T is to calculate the number of pallets which can be placed across a given width and length. It also makes the assumption that all handling units are either stackable or non-stackable.

There are two main issues with this approach. Firstly, it cannot calculated pallet space for a mix of stackable characters or a mix of different types of handling units. Secondly,

currently there is no method to identify an invalid solution (i.e. the current method output 0.5 PSE for instance which has only one standard pallet). In order to check the performance of the current method, human observers have to be allocated to validate the results. This method is not cost effective and can be time consuming. Based on these issues, there are two objectives for this collaboration. Firstly, define the problem with a better way to measure the performance of the packing method. Secondly, design a quick and easy to implement method to improve estimation.

The rest of the chapter is arranged as follows: Section 5.2 defines the Pallet Space Equivalent (PSE) problem and introduces a lower bound estimation which will be used to measure its performance. Section 5.3 introduces a new method for the PSE problem. Finally, section 5.4 presents experimental and performance analysis of the proposed method.

Actual costs can only be calculated after the delivery has been made. Additional costs can also be incurred due to factors such as change in item quantity, waiting time charge and fuel surcharges.

5.2 Pallet Space Equivalent Problem

The operational requirements of 3T has been investigated and the problem has been defined as Pallet Space Equivalent (PSE) measurement. PSE problem is defined as follows:

- Input:
  - A set of rectangular handling units with given dimensions, quantities, stackability and rotation ability.
  - A container with fixed width and height, but length can be extended as required.
  - Handling unit can be PALLET, BOX or BAG.

  - Each handling unit has one of the following stackability options: BASE, TOP, SAME, NONE.

- Output:
  - A packing plan showing all handling units and theirs positions.
- Objective:
  - To minimise the pallet space equivalent required to pack all handling units.
- Constraints:
  - All handling units must be packed.
  - Handling units need to be placed orthogonally (i.e. edges of boxes need to be parallel with the container).
  - Handling units cannot overlap.
  - Bottom surface of each handling unit has to be fully supported by either other handling units or the container.
  - Handling units with BASE stack ability allow other handling units with TOP ability to be placed on top.
  - Handling units with TOP stack ability can be stacked on top of other base handling units or on the floor of the container.
  - Handling units with NONE stack ability can only be placed on its own (i.e. a hazardous item such as a gas container).
  - Handling units with SAME stack ability can be stacked with the same handling unit type together.
  - PALLET handling units can not be stacked more than 2 on top of each other.
  - BOX, BAG handling units can be stacked more than 2 on top of other BOX and BAG.

- PALLET handling units can only rotate so that the pallet base is always at the bottom.
- PALLET handling units cannot be placed on top of BOX or BAG handling units.

To the best of our knowledge, no similar problem has been investigated in the literature. Following the classification by Wäscher et al. (2007), PSE is a three-dimensional packing problem therefore classification of dimensionality is 3. All handling units will be packed into selected containers therefore the kind of assignment classification is V. There is only one container therefore O is set as an assortment of large object classifications. In summary, the PSE problem belongs to the 3/V/O problem class according to classification by Wäscher et al. (2007). More details about classification is presented in chapter 2.

In comparison to well-known packing problems in the literature, the PSE problem is different to the container loading problem because the container length can be extended to the required length. The PSE problem is similar to the three-dimensional strip packing problem (3D-SPP) described in section 2.2 but has different requirements for utilisation measurement. The PSE problem utilisation is measured by the floor area of the container instead of the length of the container. This is due to operational requirements: 3T's customer only pays for floor space required therefore minimising floor space is the objective. An example to demonstrate the difference between 3D-SPP and the PSE problem is an instance with one PALLET of the same height as the container. Figure 5.4 shows an aerial view of the same packing plan from the top of the container. Figure 5.4a shows the utilisation using the length of the container required to pack the handling unit. In this sample, the utilisation of the strip packing problem is not 100% due to wasted space on the left side. This is different to the PSE problem as shown in Figure 5.4b. The PSE problem utilisation is measured by the floor space of the pallet. In this case, the pallet space equivalent is 1 and utilisation in this case is 100%. PSE includes two factors

mentioned by Bischoff & Ratcliff (1995) which is not studied in previous chapters:

- "Load bearing strength of items": No more than 2 PALLET handling units are stacked together.
- "Handling constraints": No BOX or BAG under a PALLET, no handling units can be packed on top of handling units with no BASE stackability.

(a) Strip packing problem (b) PSE problem

Figure 5.4: Comparison of utilisation between strip packing loading and PSE problem

5.2.1 PSE Utilisation

In the 3D-SPP an optimal container length is calculated. Optimal length and actual container length are used to calculate the utilisation as a percentage. As mentioned in chapter 3, the PSE problem is similar to the 3D-SPP. It is possible to adapt the utilisation measurement of 3D_SPP to measure that for PSE. Instead of container length, an optimal floor space area can be calculated and the utilisation of PSE can be measured based on actual and optimised floor space. Optimal floor space can be denoted as optPSE and the formula for optPSE and utilisation of PSE are shown in Equation 5.1.

Using optPSE has issues with the height and stackability of the handling unit which can result in an inaccurate or infeasible output result. For example, in an instance with

$$
\begin{aligned}
utilisation &= \frac{optPSE}{planPSE} \\
planPSE &= \text{PSE of packing plan} \\
optPSE &= \frac{\sum HandlingUnitVolume}{cH * pW * pL} \\
&\text{where } cH \text{ is container height} \\
&\qquad pW \text{ is standard pallet width (800 mm)} \\
&\qquad pL \text{ is standard pallet length (1200 mm)}
\end{aligned}
$$

Equation 5.1: PSE Utilisation formula based on 3DSPP Adaption

a single standard pallet where height is a half that of the container length, using optPSE will have a best utilisation of 50% and this is the best possible solution. Utilisation is supposed to be 100% and therefore highlights the problem with handling unit height. An example to demonstrate a stackability issue: 2 PALLET handling units where height is half that of the container height. The handling units only have BASE stackability. The best possible packing plan will require 2 pallet spaces or PSE. Using optPSE formula, optimum pallet space is 1 PSE with a utilisation of 50%. This is infeasible and violates the stackability constraint. It is not significant for comparing between different packing methods. Nevertheless, it is critical in order to make business decisions with regards to which method is acceptable for operation. From the above examples, any packing method with can only achieve utilisation of 50% is not a true reflection of the quality and usability of the packing method.

To avoid these issues, a different method for estimating the optimal floor space is proposed and is denoted as estPSE. The pseudo code calculation is shown in Equation 5.2. The main idea of estPSE is to take advantage of the known pallet quantity and their stackability constraints in order to estimate the minimum floor space required. To the best of our knowledge, there is no similar method that takes into account the handling

unit type and stackability to provide a more accurate utilisation measurement. In this evaluation, two assumptions have been made. The first assumption is when PALLET handling units are combined together they will not violate any constraints e.g. too high for the container. The second assumption is that BOX and BAG handling units can be always be packed on top of PALLET handling units. Algorithm 5.1 calculates two estimations, one for pallet (pEst) and one for volume (vEst). The final value estPSE is the higher of the two estimates. Algorithm 5.1 starts by calculating pallet space for PALLET handling units which can only be placed on its own or PALLET handling units which can only be stacked with the same type (Line 6). PALLET handling units with only BASE or TOP stackability are combined together with the assumption that no constraints are violated (Line 7). In the case where there are PALLET handling units with only BASE or TOP stackability left, they will be combined with PALLET handling units which have both BASE and TOP stackability (Line 9). After these combinations, space required for any remaining PALLET handling units is added (Line 14). pEst does not take into account BAG or BOX handling unit type with the assumption that everything else can be placed on top. pEst provides a better estimation for instances where all or the majority of handling unit types are PALLET. For simple cases such as with 1 non stackable PALLET or 2 stackable PALLET handling units then pEst will give the correct 1 PSE for each case. To accommodate instances with only BOX or BAG items, vEst is calculated using the 3D-SPP adaption as presented in Equation 5.1(Line 16). estPSE will be the highest between the pEst and vEst (Line 17). Choosing the higher value between vEst and pEst helps to avoid instances where pallets with stackability cannot be combined. For example, there are 2 large PALLET handling units which are as high as the container, one handling unit has BASE stackability and the other has BOTH stackbility. pEst is calculated as 1 PSE but this is not correct and vEst has a better estimation at 2 PSE.

$$
\begin{aligned}
utilisation &= \frac{estPSE}{planPSE} \\
planPSE &= \text{PSE of packing plan} \\
estPSE &= \text{optimum PSE calculated by Algorithm 5.1}
\end{aligned}
$$

Equation 5.2: PSE utilisation formula with modified optimal PSE calcuation using estPSE

5.3 Heuristics For Pallet Space Equivalent

As mentioned in section 5.2, one of the key objectives for this research is to design a fast, automated method to measure PSE. After discussion with the business, design and development of heuristics for the PSE problem is the preferred option. This does not exclude future work on other approaches such as meta-heuristics and hyper-heuristics.

As mentioned, the PSE problem is classified with and shares many characteristics with 3D-SPP. Therefore, the best-fit methodology from literature was adapted and three modifications were introduced: block generation, candidate point and selection criteria. An overview of the packing heuristic is shown in Algorithm 5.2. The packing process starts after block generation where items are grouped together to create a larger block of handling unit. Details of block generation are presented in section 5.3.1.

At each iteration, the lowest available gap which is presented by candidate points is selected. With the selected gap, blocks are tested to select one combination of blocks and packing position. A combination is valid when no constraints are violated. Selection criteria is introduced in section 5.3.3. After the block has been packed then the candidate points are updated it continues onto the next cycle. If there are no available blocks and positions then selected gaps are marked as invalid for future iterations. The process continues until there are no handling units left to pack.

Algorithm 5.1 *estPSE* calculation

Input: : Set of input handling units HU
Output: : Estimated optimal floor space, estPSE

$s \leftarrow NumberOfPalletOnlyHaveSameStackability(HU)$
$b \leftarrow NumberOfPalletHasBaseOnlyStackability(HU)$
$t \leftarrow NumberOfPalletHasTopOnlyStackability(HU)$
$both \leftarrow NumberOfPalletHasBaseAndTopStackability(HU)$
$none \leftarrow NumberOfPalletHasNoStackability(HU)$
$pEst \leftarrow none + \lceil \frac{s}{2} \rceil$
$combBT \leftarrow \min(b, t)$
$baseOrTop \leftarrow \max(b - combBT, t - combBT)$
$combBTL \leftarrow \min(both, baseOrTop)$
$pEst \leftarrow pEst + comBT + combBTL$
$bothLeft \leftarrow \max(both - combBTL, 0)$
$baseLeft \leftarrow \max(b - combBT - combBTL, 0)$
$topLeft \leftarrow \max(t - combBT - combBTL, 0)$
$pEst \leftarrow pEst + \lceil \frac{baseLeft + topLeft + bothLeft}{2} \rceil$
$volume = \sum(i.Volume), i \in I$
$vEst = \frac{\sum HandlingUnitVolume}{container.Height * standardPalletArea}$
$estPSE \leftarrow \max(pEst, vEst)$

Algorithm 5.2 PSE packing best-fit heuristic algorithms

Input: : container C, set of handling unit to pack B, set of generated blocks BL
Output: : packing plan P, with all blocks and its position defined

$P \leftarrow \emptyset$
while |B| >0 **do**
    Get lowest gaps, G
    Find all valid block, P from BL for G
    Select a combination block and packing position, p from P
    if p is found **then**
        Reallocation(p)
        Remove from B the handling unit in p
    else
        Mark all gaps in G as invalid
    end if
end while

5.3.1 Block Generation

Block generation processes are shown in Algorithm 5.3. The block generation process starts with the generation of a simple block for each handling unit (Lines 5, 6 and 7). Simple blocks will be combined together to generate group blocks (Line 8). Block generation returns both generated simple blocks and group blocks.

Algorithm 5.3 Block generation process to generate simple block for different types of handling unit

Input: : A set of input handling units, HU
Output: : A set of generated block, bl

```
1: procedure BLOCK GENERATION
2:     pallet ← HU.GetPalletHandlingUnit()
3:     box ← HU.GetBoxItemHandlingUnit()
4:     bag ← HU.GetBagItemHandlingUnit()
5:     simplePallet ← GenerateSimpleBlock(pallet)
6:     simpleBox ← GenerateSimpleBlock(box)
7:     simpleBag ← GenerateSimpleBlock(bag)
8:     groupBlock ← GenerateGroupBlock(simplePallet, simpleBox, simpleBag)
9:     bl ← simplePallet + simpleBox + simpleBag + groupBlock
10:    return bl
11: end procedure
```

Simple blocks are generated by combining the same handling unit type into the same block. Simple blocks are generated by the GenerateSimpleBlock process as shown in Algorithm 5.4. GenerateSimpleBlock input is the handling units. For each handling unit, all possible rotations are considered (Line 4). maxX, maxY and maxZ are the maximum number of handling units across the x-, y- and z-axes in a block. maxX and maxZ are calculated as the maximum number of blocks that can be placed across the axis and are also restricted by pre-set parameters (Line 5 and 6). maxY is calculated differently using the standard container length which is set in the experiment (Line 7). xBound, yBound and zBound are the limits of the number of handling units along each axis. This is to avoid a long run time which may occur during block generation. The values of xBound, yBound and zBound are specified in the experimental set up. x, y

and z are the number of handling units across the x-, y- and z-axes of the container. For each combination of x, y, and z, a block is generated and checked for validation (Line 11 and 12). A simple block is valid when no constraint is violated in quantity, dimension, bearing or stackability. For example, a combination of x = 3, y = 2, z = 2 will produce a block such as that shown in Figure 5.5.

Algorithm 5.4 Process to generate simple block

Input: : A set of input handling units, HU
Output: : A set of generated simple block, simpleBlock

```
 1: procedure GenerateSimpleBlock
 2:     simpleBlock ← ∅
 3:     for all Handling unit, h from HU do
 4:         for all possible rotation,r of HU do
 5:             maxX ← min(⌊c.Width/r.Width⌋, xBound)
 6:             maxZ ← min(⌊c.Height/r.Height⌋, zBound)
 7:             maxY ← min(⌊c.standardLength/r.Length⌋, yBound)
 8:             for x = 1 → maxX do
 9:                 for y = 1 → maxY do
10:                     for z = 1 → maxZ do
11:                         block ← GenerateBlock(r, x, y, z)
12:                         if IsValid(b) then
13:                             Add block to simpleBlock
14:                         end if
15:                     end for
16:                 end for
17:             end for
18:         end for
19:     end for
20:     return simpleBlock
21: end procedure
```

After Simple block generation then Group blocks are generated. Group blocks are generated by combining multiple Simple blocks together. Group block also takes into account stackability and bearing constraints between different handling unit types. Handling units of the same type which have the SAME stackability will be grouped together. Handling units of different types can only be combined if they have suitable stackability. For example, a BOX with BASE stackability can be combined with another BOX with

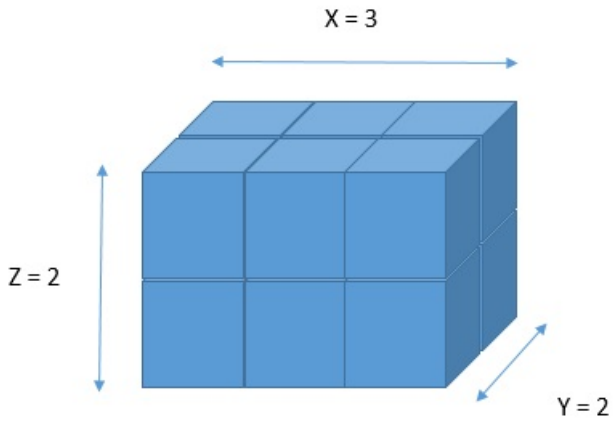


Figure 5.5: A sample of a simple block

TOP stackability. Group block generation is shown in Algorithm 5.5 where the process combines different types of handling units together. GenerateGroupBlock process starts with an initial generated block, grBl with all simple blocks of PALLET (Line 2). For each iteration, the Combine process combines the current generated block with another simple block type in the following order: PALLET, BOX and then BAG (Line 4). This particular order is used to ensure that a BAG will be placed on top of a BOX item. This is not a constraint but a preferred practice in real life. The process continues until a number of blocks are generated or no more new blocks are created. In the first iteration, simple blocks of PALLET handling unit are combined with other PALLET, BOX and BAG handling units. The limit of the number of blocks generated is specified in section 5.4.2.

The Combine procedure, shown in Algorithm 5.6, combines a current block with a Simple block. Each current block is combined with a Simple block using three types of combination NEXT, ABOVE and INFRONT. A NEXT arrangement places a Simple block along the x-axis with current block. An ABOVE arrangement places a Simple block on top of a current block. An INFRONT arrangement places a Simple block in front of

Algorithm 5.5 Process to generate group block from simple block of different handling unit types

Input: : A set of simple block of each handling unit type, simplePallet, simpleBox and simpleBag
Output: : A set of generated block, grBl

```
1: procedure GenerateGroupBlock
2:     grBl ← simplePallet
3:     while max number of block is reached or no more block can generate do
4:         groupPalletBlock ← Combine(grBl, simplePallet))
5:         Add groupPalletBlock to grBl
6:         groupBoxBlock ← Combine(grBl, simpleBox))
7:         Add groupBoxBlock to grBl
8:         groupBagBlock ← Combine(grBl, simpleBag))
9:         Add groupBagBlock to grBl
10:    end while
11:    return grBl
12: end procedure
```

a current block if looking from the extended end of the container. This is different to previous block generation for the strip packing problem. Different to the strip packing problem, a long block does not decrease utilisation due to different evaluation functions. Figure 5.6 shows an example of one Simple block in Figure 5.6a and a Group block in Figure 5.6b. The three possible combinations between generated block and Simple block are shown in Figure 5.7. For each combination, only valid combinations are accepted. A combination is valid when:

- The block is placed on its own on the container floor, no stability constraint is violated.
- Width and height are smaller than the width and height of the container.
- There are enough handling units to form at least one block.
- Volume utilisation of the block is greater than or equal to 98%.

The volume utilisation of a block is measured using the same formula as shown in chapter 4. The value 98% is selected based on observation from rigorous experiments.

For example,a PALLET handling unit which has BASE stackability can be combined with a PALLET handling unit which has TOP stackability. After a combination of PALLET block types are generated, a block contain BOX handling unit can be combined with the PALLET block. It is possible to combine a BOX block with other blocks which contain more than one type of PALLET. This process is similar to the loading process in warehouses where pallets will be packed into the truck first. Then boxes will be placed on top or next to pallets if possible to increase vehicle utilisation. Finally if there are any bags left, loading personnel will normally place these on top of everything else. At the first iteration, the generated Group block contains only two types of handling unit. After each iteration, the number of handling unit types of blocks is increased by one. The generation of Group block continues until a certain number of generated blocks is reached or no more new blocks can be found. There is a limit to the number of group blocks that will be set in an experimental set up.

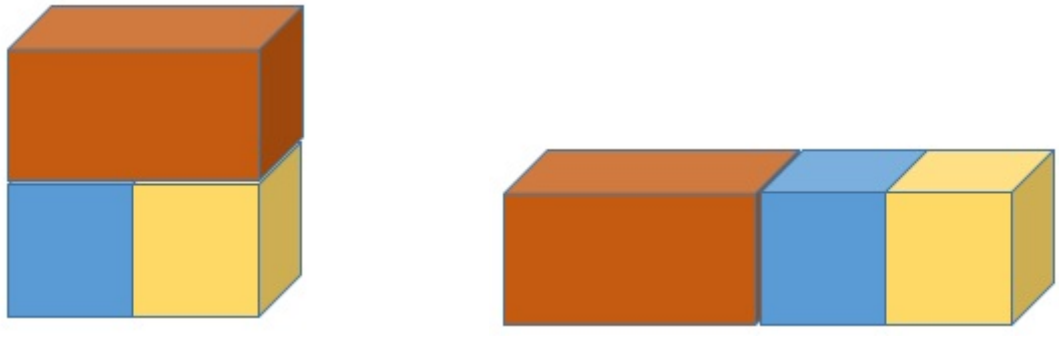
(a) Simple Block (b) Group Block

Figure 5.6: Simple block and Group block

Algorithm 5.6 Process to combine a Group block and a Simple block

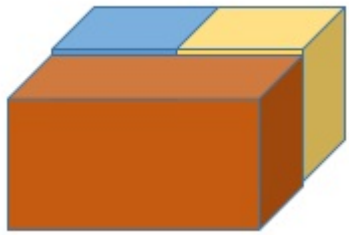
Input: : A set of previously generated blocked,gBLs and a set of simple block, sBL
Output: : A set of generated block, combinedBlock

```
1: procedure COMBINE
2:     combinedBlock ← {}
3:     for all b ∈ gBL do
4:         for all s ∈ sBl do
5:             nextBL ← GenerateNextBlock(b, s)
6:             if isValid(nextBl) then
7:                 Add nextBL to combinedBlock
8:             end if
9:             aboveBL ← GenerateAboveBlock(b, s)
10:            if isValid(aboveBL) then
11:                Add aboveBL to combinedBlock
12:            end if
13:            infrontBL ← GenerateInfrontBlock(b, s)
14:            if isValid(infrontBL) then
15:                Add infrontBL to combinedBlock
16:            end if
17:        end for
18:    end for
19:    return combinedBlock
20: end procedure
```



(a) ABOVE

(b) NEXT



(c) INFRONT

Figure 5.7: Group Block Arrangements

5.3.2 Candidate Point

For each iteration, heuristics start by selecting the lowest gap. The original 3BF heuristic used Extreme Point presentation to represent the gap. In this work, the approach is adapted to take into account additional constraints. After a block has been placed into a position, a collection of points are added to present the gap. A block and its placed position can be denoted as B and P, respectively. The following coordinates will be added to the candidate point list:

1. Point (P.x, P.y, P.z + B.height)
2. Point (P.x+B.Width, P.y, P.z)
3. Intersection between projection from point (P.x, P.y+B.length, P.z) down the z-axis and previous by packed block or container.

4. Intersection between projection from point (P.x, P.y+B.length, P.z) down the x-axis and previous by packed block or container.

An example of candidate points are shown in Figure 5.8. In Figure 5.8, point (1) is the top of the block point, point (2) is the point at the left hand-side of the figure, point (3) is the bottom point and point (4) is the right most point. It is worth noting that the number of new candidate points are significantly less than in the original process in chapter 3. Due to the requirement of the stability constraint, candidate points which offer no support are avoided.

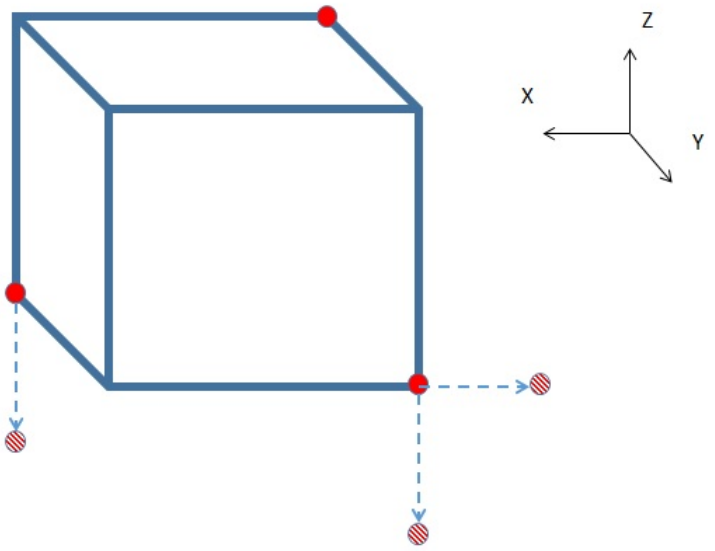


Figure 5.8: Candidate Points

5.3.3 Selection Criteria

From all valid combinations of blocks and packing positions, different criteria and tie breakers are used to select a combination to pack. There are three main criteria:

- Max Volume: block with highest volume is selected. If there are multiple blocks with the same volume then tie breakers are used in the following order:

  - Block with maximum area at the XZ surface
  - Block with maximum height
  - Block with deepest bottom left most position
  - Block with smallest quantity of handling units

- Max Contact: Block with highest contact is selected. Contact area is measured by the formula $ContactArea = ContactAreaWithOtherBlocks*2+ContactAreaWithContainer$ If there are multiple blocks with the same contact area then tie breakers are used in the following order:
  - Block with maximum area at the XZ surface
  - Block with maximum height
  - Block with deepest bottom left most position
  - Block with smallest quantity of handling units

- Max Bottom: the block with the largest bottom (area of XY surface) is selected. If there are multiple blocks with the same contact area then tie breakers are implemented in the following order:
  - Block with maximum height
  - Block with deepest bottom left most position
  - Block with smallest quantity of handling units

5.4 Experimental

In this section, expriments which have been carried out to evaluate the heuristics are presented. First, data sets which have been collected and used as a bench mark are presented in section 5.4.1. The set up of the experiments is outlined in section 5.4.2 and the experimental results are given in section 5.4.3.

Data Set	BR (Box Type)	Estimate PSE	Item Type	Valid (%)	Pallet Type (%)	Other Type (%)
TT1	BR1 (03)	07.96	01.76	69	76	24
TT2	BR2 (05)	12.47	04.45	38	67	33
TT3	BR3 (08)	16.06	06.79	36	74	26
TT4	BR4 (10)	18.60	09.38	29	68	32
TT5	BR5 (12)	18.46	11.50	16	61	39
TT6	BR6 (15)	21.83	13.85	23	65	35
TT7	BR7 (20)	26.24	17.27	15	66	34
TT8	BR8 (30)	22.95	22.00	00	68	32

Table 5.1: TT data set properties and 3T's current method performance overview

5.4.1 Data Set

We have collected 3300 instances from 3T Logistic Ltd's database and converted these into a data set called TT. To the best of our knowledge, no research has been conducted for this problem using real-life data. This real-life data was gathered from various customers during a two-week period of warehouse operation. In order to have a comparison with data sets in the literature, the TT data was separated into different ranges of box types. The range of box types correspond to the BR data set box types which are split into the TT data set and into 8 subsets from TT1 to TT8.

Table 5.1 shows the data set properties and corresponding BR data set. The Estimate PSE column indicates the average estPSE for all instances in each subset. Item Type column is the average number of handling unit types across all instances. There is a difference between the TT and BR data set. The number of box types in each BR data set is the same but the number of handling unit types is within a range in TT dataset. For example, in the instance of BR1 there are 3 different types of box but for the instance of TT1 there are between 1 and 3 types of handling units. The Valid column shows the percentage of current valid calculations using the current 3T method. A calculation is invalid when the resulting pallet space is smaller than the estimate PSE.

Table 5.2 shows the stackability of the PALLET, BOX or BAG handling units. The percentage of handling units which have BASE, TOP, SAME and NONE stackability are reported. It is worth mentioning that NONE stackability handling units are only a small

Data Set	Pallet				Other			
	Base (%)	OnTop (%)	Same (%)	None (%)	Base (%)	Top (%)	Same (%)	None (%)
TT1	38	16	17	37	09	09	09	15
TT2	34	22	22	34	10	14	14	19
TT3	37	20	19	37	13	15	15	11
TT4	33	14	14	34	22	27	27	06
TT5	37	18	18	23	33	37	37	02
TT6	45	25	25	20	28	34	34	01
TT7	46	26	26	19	29	34	34	00
TT8	68	27	27	10	30	32	32	00

Table 5.2: Percentage of TT data set of different handling unit types and stackability

percentage in comparison to the others but they take up more floor space than other stackable handling units.

Table 5.2, indicates underestimating PSE issue of 3T's current method. From TT1 to TT10 the percentage of valid calculations using 3T's current method decreases. This shows that 3T's current method does not perform well when there are more handling unit types. A possible cause is that 3T's current method will assume everything is either stackable or non-stackable. This might not be a problem with a low number of handling unit types. However, there is more likely to be a mixed stackability with a higher number of handling unit types. It is possible to cause significant problems in operation when the carrier collects items which are different than expected. e.g. a delay in delivery or a higher than expected cost. This is especially true when the majority of the items are PALLET types in comparison to other item types (Table 5.1) and about one third of PALLET items cannot be stacked together whereas other items have more relaxed stackability. This reflects the real-life nature of a handling unit when small items of small quantities are not palletised and they are normally placed in a bag or box to be transported.

5.4.2 Experimental Set Up

The proposed method is implemented in C# programming language (.NET Framework 4.0) and single thread code. This is different to chapter 3 and chapter 4 where the methods were implemented using Java. The change in language is due to a compatibility

requirement with 3T's current system. All experiments were run on a virtual server with the following specifications:

- Operation System: Windows Server 2008 R2
- CPU Intel Xeon 2.67 Ghz
- RAM 1G

The experimental parameters were set as: xBound = 25, zBound = 30, yBound = 100, standard container length: 13600 mm, maximum generated block number = 10000. Container standard length was used during block generation to avoid very long blocks being generated. This is to encourage a more realistic outcome but it is not a constraint during the packing process.

5.4.3 Experimental Results

Table 5.3 shows the result of the heuristics with the TT data set. Max Volume, Max Contact and Max Bottom columns show the average utilisation of the corresponding heuristics for each data set. The Max All column shows the average utilisation with the highest utilisation from Max Volume, Max Contact and Max Bottom. Only Max Volume, Only Max Contact and Only Max Bottom columns show the percentage of instances in which only the corresponding heuristics has the highest utilisation.

From table 5.3, it can be seen that Max Volume outperforms the other two heuristics especially in data sets with a small number of box types. However, for instances with a larger number of box types there is not a significant difference between the performance of Max Volume and Max Contact. By combining the highest utilisation from all the heuristics, the overall utilisation is increased by nearly 4%. Max Bottom only has the highest utilisation at 3% for all instances.

Data Set	Estimate PSE	Max Volume (%)	Max Contact (%)	Max Bottom (%)	Max All (%)
TT1	07.96	87.24	65.23	61	90
TT2	12.47	72.25	64.21	57	77
TT3	16.06	76.24	70.32	63	76
TT4	18.60	76.32	70.45	61	78
TT5	18.46	67.14	62.67	47	72
TT6	21.83	62.14	66.70	50	71
TT7	26.24	68.15	64.81	53	73
TT8	22.95	60.17	60.12	47	61
AVERAGE	11.46	71.2	65.5	54.8	74.8

Table 5.3: Performance of heuristics and their best combined results with the TT dataset

It is possible to obtain the same result for different heuristics. Table 5.4 shows the number of instances where only one heuristic has the highest utilisation. Max Volume also shows the greatest number of instances with the best performance compared to the other two heuristics. The Max Contact heuristic also performs best in some instances but not in as many instances as Max Volume. The Max Bottom heuristics has some best results but was not significant compared to the others. This shows that in order to obtain Max All results, as presented in Table 5.3, we only need to combine the Max Volume and Max Contact heuristics. If we can select only one heuristic then Max Volume is likely to give the best utilisation and only loses about 4% from the highest utilisation.

The average run time for each instance is less than 5 seconds for each heuristic and in most cases it is significantly less than 5 seconds. With this result, and following examination by the 3T planning team, the proposed heuristics approach was found to be suitable for real-life applications. Therefore, we integrated the Max Volume heuristic into the 3T system and the results provided by 3T have been very encouraging as they show an increase of around 20% utilisation and the under-estimation issue has been eliminated.

Table 5.5 shows the detailed performance of the Max Volume heuristic. The results of the Max Volume heuristic are separated into two parts, above and below average, which are presented in Table 5.5. For each part, data for PALLET and other handling unit types are presented with the average quantity and average type number. It can be seen that the performance of the Max Volume heuristic is dependent on the proportion of

Data Set	Only Max Volume	Only Max Contact	Only Max Bottom
TT1	21	01	01
TT2	43	05	02
TT3	42	09	03
TT4	41	16	05
TT5	51	28	04
TT6	41	35	06
TT7	55	33	03
TT8	54	53	01
AVERAGE	43.5	22.5	03

Table 5.4: Number of instances where only one heuristic has the highest utilisation

Data Set	Utilisation Lower Average				Utilisation Above Average			
	Pallet Quantity	Other Quantity	Pallet Type	Other Type	Pallet Quantity	Other Quantity	Pallet Type	Other Type
TT1	03.49	06.24	0.86	0.84	13.71	2.05	01.28	0.13
TT2	10.39	16.37	2.16	1.84	18.67	9.67	03.48	0.52
TT3	12.15	28.98	4.01	2.49	25.51	7.12	05.73	0.64
TT4	14.54	65.60	5.38	3.63	27.35	9.18	07.94	1.06
TT5	14.44	83.38	5.87	5.13	30.67	7.30	11.00	0.81
TT6	14.08	93.21	6.85	6.54	52.44	5.11	12.67	0.89
TT7	24.37	128.42	10.21	6.95	63.40	11.20	14.80	1.80
TT8	28.67	113.67	14.33	6.33	74.52	16.23	17.53	2.45

Table 5.5: Summary of Max Volume performance separated into higher and below average instances

PALLET to other item types. Instances with low utilisation have a lower quantity of PALLET than other types of handling unit. It is the opposite to instances with have more PALLET than other types of handling unit. From the bottom 10% utilisation, 59% of the lowest utilisation has no PALLET items and this is due to the estimate PSE calculation. The estimate PSE calculation takes into account the PALLET item's stackability but not with other item types. This is an over-estimate during the estimate PSE calculation for BOX and BAG handling unit items. An example of an instance of low utilisation is one BOX with the same dimensions as a standard pallet and with an estimate PSE of 0.02 which is significantly less than the actual 1 PSE minimum space required.

5.5 Conclusion

In this chapter, a new type of problem which arises from real-life business operations - Pallet Space Equivalent Measurement (PSE) is defined. To the best of our knowledge this is a new problem that has not been reported in the literature. The PSE problem is a variation of the three-dimensional strip packing problem with different utilisation and addition constraints. The properties of the PSE problem were also taken into account to design new utilisation measurements for the PSE problem. This offered improved accuracy in utilisation measurements and allowed for a method of validating the result of a packing method. Due to certain business requirements, Maximum Contact, Maximum Volume and Maximum Bottom heuristic approaches were utilised to tackle the PSE problem. In order to accommodate new constraints and changes in block generation, a number of possible packing point generation and selection criteria were implemented. Data sets were collected and created from real life operations for the PSE problem instead of using generated data sets as in chapter 3 and chapter 4. From the experiments, Maximum Volume and Maximum Contact showed superior results in comparison to the Maximum Bottom heuristic with a limited amount of computation time. The majority of the best results could be obtained by combining only Maximum Volume and Maximum Contact. However, in contrast to the three-dimensional strip packing problem, Max Volume had better utilisation compared to Max Contact. The performance of the heuristic had a strong relationship with the number of PALLET items due to utilisation measurements considering PALLET handling unit types separately to others. Further work can be done to continue to improve the utilisation measurement.

6 Clustering Effect And Planning Quality In Multi-Carrier Transport

Landa-Silva et al. (2011) introduced the single-customer multi-carrier problem (SCMC) with a hybrid heuristics approach (HHLP). In this chapter, the SCMC problem is updated to reflect recent changes in 3T's operations. Different components of HHLP are investigated and modifications to adapt to the new requirements are proposed. The chapter is structured as follows: in section 6.1, new operational requirements to the multi-carrier planning problem presented in Landa-Silva et al. (2011) are addressed. In section 6.2, different clustering algorithms are selected as an alternative for the original DBSCAN in HHLP and different cluster performances in different scenarios are compared. From 3T's operational feedback, a new load building issue was identified resulting from inefficient planning. Details of the issue are introduced in section 6.3. In section 6.4, two approaches are proposed in order to resolve the inefficient planning using evaluation functions and local search operators. Experimentation and evaluation of different clustering is presented in section 6.5 and experimentation and evaluation of modifications to resolve the inefficient planning are given in section 6.6.

6.1 Single-Customer Multi-carrier Planning Problem

Based on the work of Landa-Silva et al. (2011), 3T has implemented HHLP into its French operation. With significant success in France, 3T's strategy is to extend implementation

	Shipment Number	FLT	LTL	Groupage	Straight Distance	Straight Driving Time	Between Distance	Between Driving Time	Booking Windows
LUK	84	1	15	67	153	170	187	200	380
LSP	48	1	19	29	363	348	314	320	250
LFR	12	0	1	11	233	235	251	260	120
Landa-Silva et al. (2011) (OLUK)	74	8	18	48	148	167	173	197	445

Table 6.1: Shipment properties at United Kingdom, Spain and France

of HHLP to the United Kingdom and Spain. However, there are significant differences in the geographical nature and shipment profiles across these different countries. Table 6.1 shows a summary of shipment properties for the customer sites in the United Kingdom (LUK), Spain (LSP) and France (LFR), and also the original UK data properties from Landa-Silva et al. (2011)(OLUK). The UK site has the highest number of shipments and the French site has the lowest number of shipments. Business profiles change, as denomonstrated by the difference between the original UK data from Landa-Silva et al. (2011) and the current UK profile. The number of full truck loads (FTL) in the UK site is significantly lower in the original UK data. FTL shipments are assigned a dedicated vehicle with no other shipments. Due to this arrangement, the FTL plan has no delivery conflicts. Therefore, the current UK data has more possible combinations between shipments and increases the solution space. Another factor is driving distance, driving distance straight from the source to the destination and between each shipment is similar for both the current UK and original UK data. However, booking windows are more restricted for the current UK data. This reflects current trends in the business transition: customer bases in the current and original UK data are similar, but customer order size has become smaller and more frequent and more "on demand". This is a reflection of the current economic situation where companies want to reduce their operational costs, making more demands on manufacturers. This places a greater constraint on the current UK requirement compared to the original scenario.

Figures 6.1, 6.2 and 6.3 show sample distributions for one day at three different customer sites in the United Kingdom (LUK), Spain (LSP) and France (LFR). As shown in Table 6.1, shipment distributions across different sites have different properties. LUK and LSP shipment distributions are generally around big cities: Manchester and London

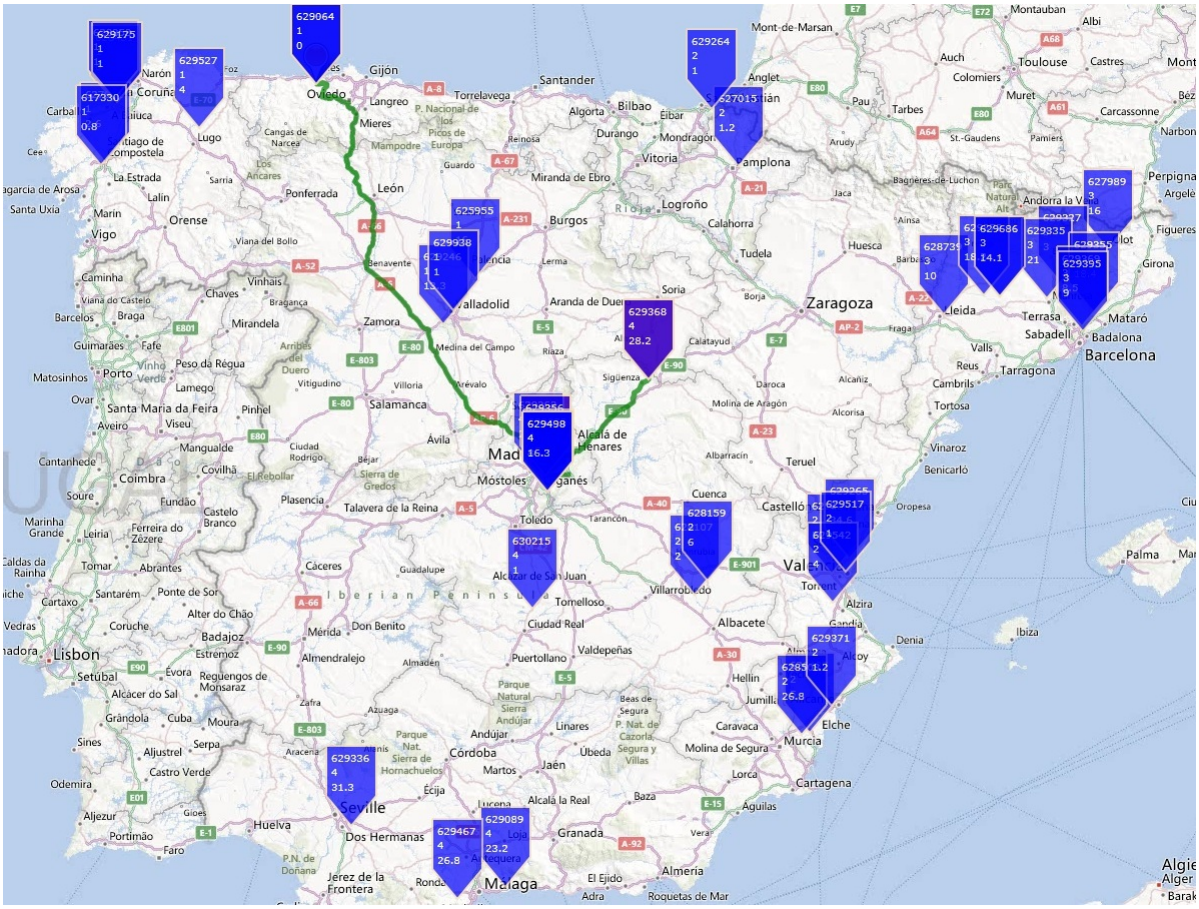


Figure 6.1: LSP shipment distribution

in the UK; Barcelona and Madrid in Spain. For France, due to the small number of shipments, it is more difficult to show this pattern. It is worth mentioning the different positions of warehouses which act as the shipment's source: the UK warehouse is located near the centre of the UK whereas France's and Spain's warehouses are located to the side and near the country's border, respectively. This causes some issues when planning in Spain and France. For example, due to limited delivery volume and longer driving time to other regions of Spain, it is harder to maximise the carrier's utilisation and driver time when planning deliverys for the West and North West of Spain.

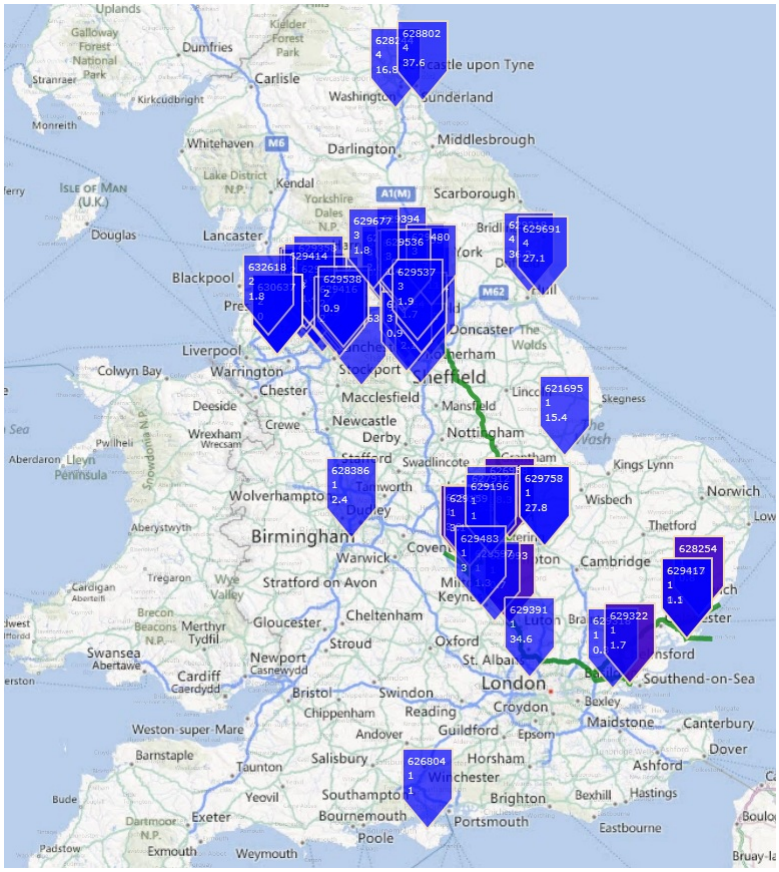


Figure 6.2: LUK shipment distribution

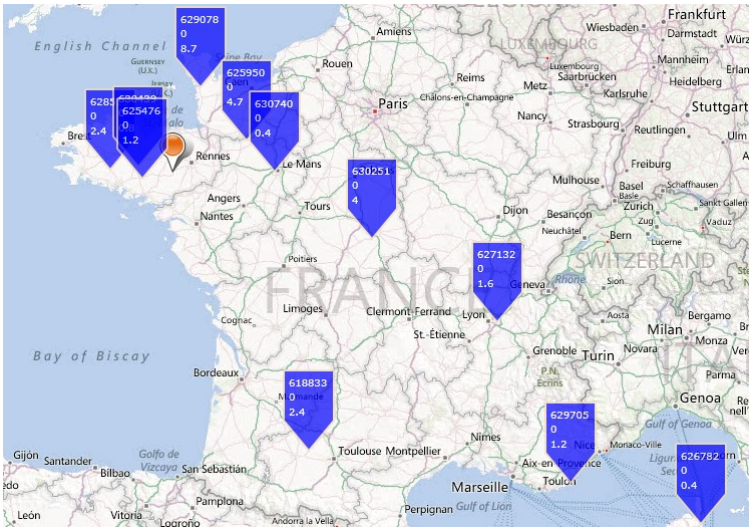
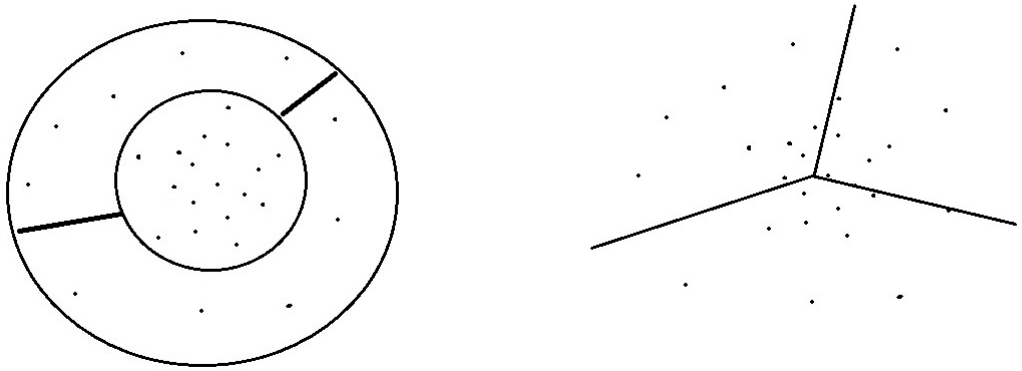


Figure 6.3: LFR shipment distribution

6.2 Evaluation of Different Clustering Algorithms

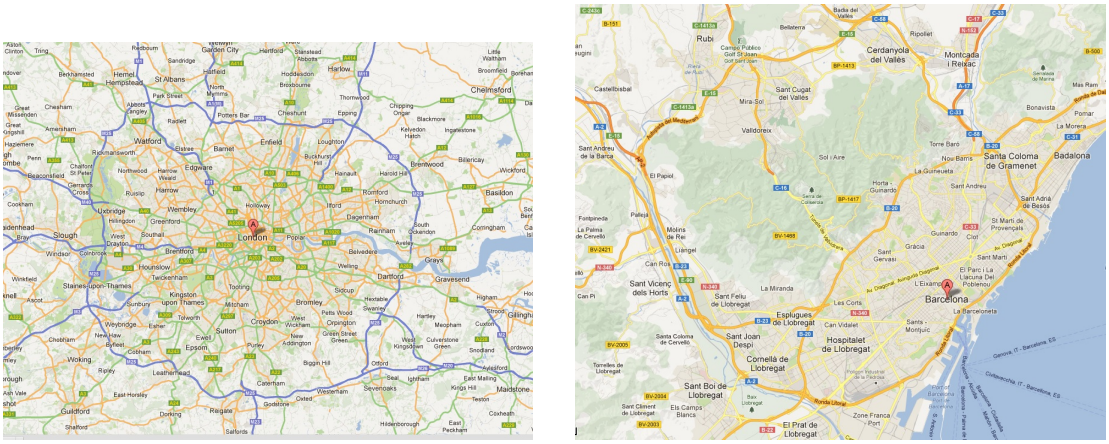
As mentioned in Section 6.1, shipment distributions are normally close to and concentrated near big cities and a large cluster is likely to be created around a big city. Cluster algorithms (DBSM) adapted from the original DBSCAN algorithm (Ester et al. 1996) has been used by Landa-Silva et al. (2011) where larger clusters are partitioned into smaller clusters by reducing the parameter distance of DBSCAN (ε) between shipments. It was a very effective approach for the clustering of destinations of shipments for OLUK and produced suitable transport plans as described in Landa-Silva et al. (2011). However, during implementation in 3T's operation, two issues were identified. The first issue happens when big clusters are partitioned into smaller clusters especially if a cluster exists in a city area. Cluster by DBSM created an unbalanced plan where some routes were concentrated within city centres and other routes went around the city. Figure 6.4a shows a sample of a large cluster that has been separated using DBSM. Initially, all the delivery points belong to a single large cluster, DBSM then reduced the distance limit between shipments and split larger clusters into smaller sub-clusters. It can be observed that the ring road of the city also acts as the boundary for the inner cluster. The closer the shipment to the city centre, the shorter the distance between shipments. Hence, DBSM tends to group inner city shipments into a cluster and groups outside shipments into other clusters. During the initial load building process, outer and inner shipments will be grouped into separate plans. This particular type of arrangement causes concern for carrier operations: inner city plans normally require significantly longer times to complete due to the nature of the traffic. It is also not possible to measure actual driving time in a city not only because it is a longer distance but also because changes can happen rapidly and more frequently. In some cases, it is much more difficult to travel between deliveries and to return from the city and back to the depot which is normally located outside the city and within working hours. Furthermore, using DBSM algorithms to split big clusters, we cannot identify the number of clusters that will be created. For



(a) Cluster boundary generated by DBSM (b) Cluster boundary generated by DBSKM

Figure 6.4: Different clustering results from DBSM and DBSKM

a larger cluster, with a known number of shipments inside the cluster, a number of sub-clusters can normally be pre-defined. For example, a human planner will split a city into regions with a set number of drops per region. Figure 6.5, shows typical examples in London (UK) and Barcelona (Spain). This also highlights one difference between human planners and automated planning. Human planners normally use straight line distance as a guid when creating a plan. Even the most experienced human planner will only have knowledge on actual driving distance for a limited area. In some instances, plans created by human planners can be infeasible when driving and straight line distances are significantly different. Figure 6.6 demonstrates an example of the difference between driving and straight line distance.



(a) London ring road (b) Barcelona ring road

Figure 6.5: A sample of city ring roads

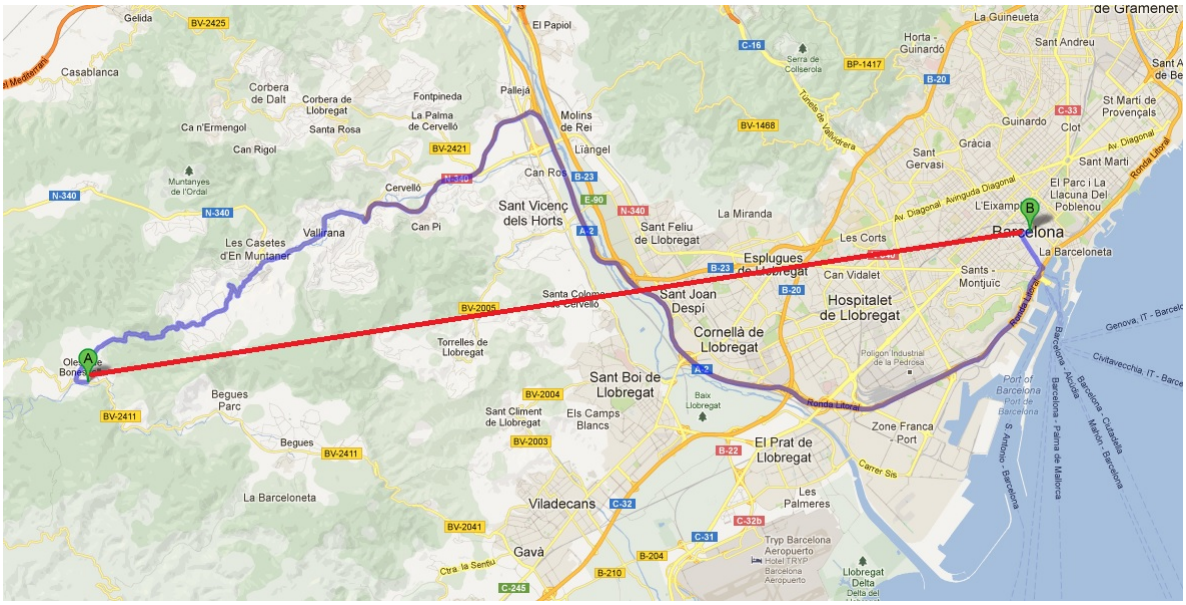


Figure 6.6: Difference between straight line and actual driving distance

To overcome this issue, a combination of DBSCAN and K-Mean is proposed. The idea is to utilise the features of K-Mean to create clusters which have a balance between inner and outer city shipments as shown in Figure 6.4b. Hence, plans are created with delivery occuring both outside and inside the city. Because all delivery points are already relatively close together and in a large cluster, it is possible to identify the number of clusters required as a parameter for K-Mean from the number of drops allowed in each vehicle. In addition, we want to explore different options of clustering algorithms to find out if there is already a better clustering algorithm which can be used for this problem. In general, there is no one clustering solution for all problem domains. Different problem clustering solutions are designed based on different factors in criteria, requirements or assumptions etc. Wide ranging surveys have been published to review different types of clustering advantages and disadvantages. Comprehensive clustering reviews are given by Jain et al. (1999) and Xu & Wunsch (2005). Clustering algorithms can be classified into different categories: partition method, hierarchical method, density-based method, grid-based method and model-based method (Kotsiantis & Pintelas 2004). A combination of clustering algorithms is also a suitable approach. Strehl & Ghosh (2003) proposed different ways of combining different clustering techniques. Based on these surveys, the following cluster algorithm approaches were selected:

- Original DBSCAN (DBSOR): the original implementation of DBSCAN introduced by Ester et al. (1996).
- Multiple DBSCAN (DBSM): the adapted DBSCAN has been used by Landa-Silva et al. (2011).
- SLINK (SLINK): the original implementation introduced by Sibson (1973).
- Expectation–maximization (EM): the implementation was introduced by Dempster et al. (1977).
- K-mean (KMEAN): the original implementation was introduced by Lloyd (1982).

- DBSCAN + K-mean (DBSKM): the proposed combination of DBSCAN and K-Mean.

6.3 Inefficient Measurement

Due to the complexity of real-life operations, it is difficult to measure the quality of the plans created. Different criteria have been selected to measure quality. Landa-Silva et al. (2011) selected vehicle utilisation, delivery violation, cost, driving distance and backward distance. Backward mileage is measured to avoid subsequent delivery points which are closer to the source than previous delivery points. A sample of a plan with backward mileage is shown in Figure 6.7. Direct distance from the last drop (shipment 62543) is closer than the direct distance to that of the previous drop (shipment 624980). Given that there is no time violation, reduced backward mileage helps create delivery plans that are suitable for carrier preference and cost structure. In general, carrier cost structure is normally in proportion to the distance of the furthest delivery. During 3T's operation, it has been identified that some plans created by an automated planning process have been rejected by the carrier. Upon observation, two types of plan were classified that were rejected and not captured by the original criteria. The first type of rejected plan is shown in Figure 6.8. The last drop (shipment 630039) has the longest distance from the source compared to the other previous shipments. This plan was rejected by the carrier due to the very long distance from the Manchester region with only one delivery to the Cambridge region which had four deliveries. The other type of rejected plan is shown in Figure 6.9. This happens when a shipment's collection is inside the same cluster or region as the shipment's destination.

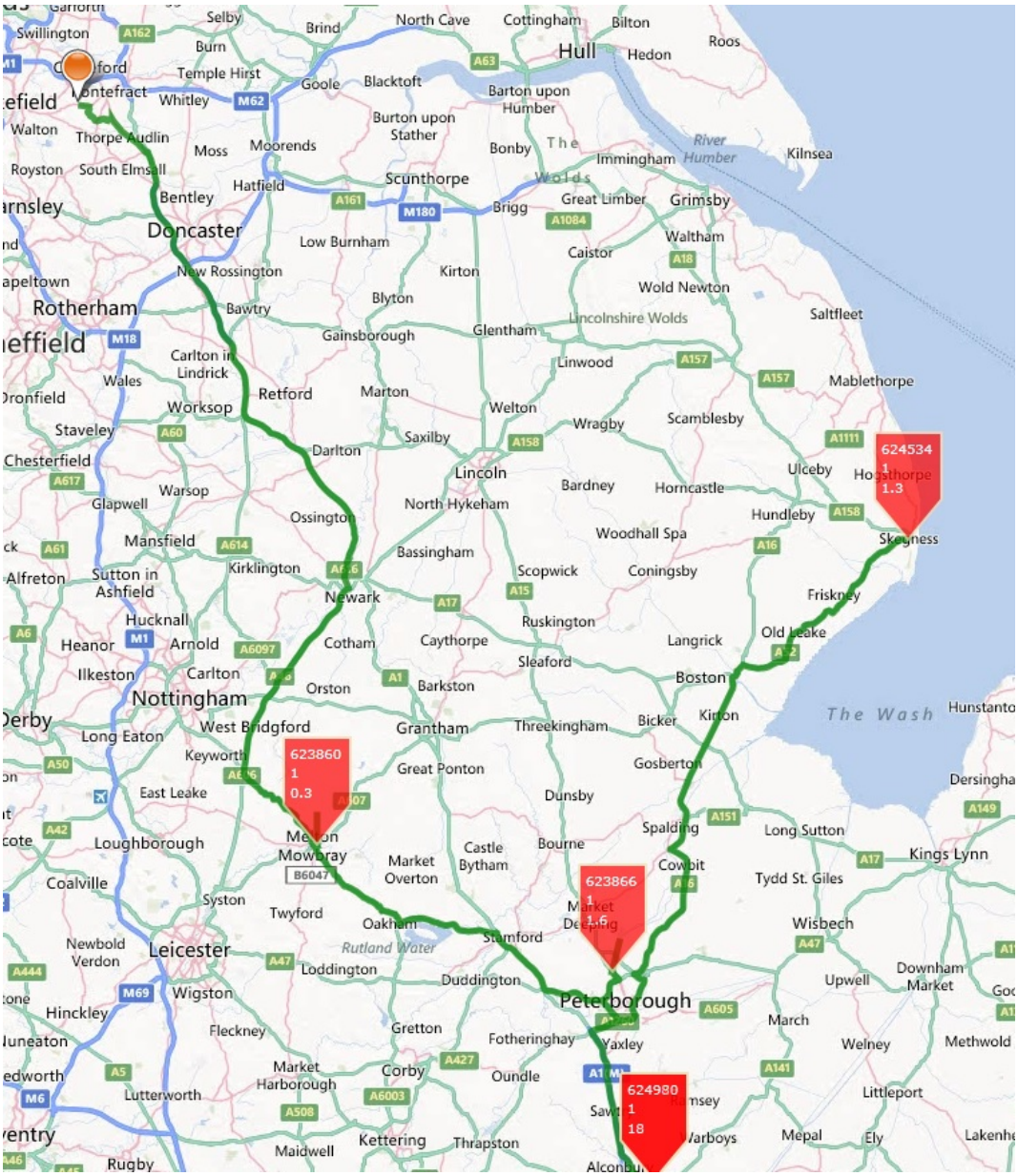


Figure 6.7: A sample of delivery plan contain backward milage

Figure 6.8: Carrier rejected plan as route going through regions in different direction

Figure 6.9: Carrier rejected plan when shipment's souce is in the same region with its destination

In order to detect this problem, we propose a new factor to measure plan quality - inefficient mileage. The formula used for inefficient mileage is defined in Figure 6.10. The main idea for calculating inefficient mileage is to compare the current planned route to a direct route for each of the delivery points. For a perfect plan, a route going though all deliveries to the final delivery will have zero inefficient mileage. In Figure 6.8, the driving route to the last four deliveries is significantly longer compared to the direct distance from the source to the last four deliveries, therefore inefficient mileage will be included for each of the last four drops. Another example is given in Figure 6.9, where the last two deliveries will result in inefficient mileage. Inefficient mileage evaluation depends on the value of the parameters α and β. Smaller values of α encourage a straighter route to the current delivery point from the previous delivery point. Smaller values of β will help straighten the overall route.

$$RI(l) = \begin{cases} true & \frac{RIM(l)}{R(l,d_n)} > \beta \\ false & otherwise \end{cases} \tag{6.1}$$

$$RIM(l) = \sum_{i=1}^{n-1} I(l, d_{i+1}) * (R(l,d) - DIR(d_i)) \tag{6.2}$$

$$I(l,d) = \begin{cases} 1 & \frac{R(l,d)}{DIR(d)} > \alpha \\ 0 & otherwise \end{cases} \tag{6.3}$$

where

- l is the plan to measure
- d is the delivery point of the plan
- n is the number of drops in the load
- DIR is the direct distance from source to destination of drop d
- R is the distance of the load - l from source though each delivery point up to delivery point d
- α and β are parameter thresholds and are specified in the experiment set up.
- RI functions to identify if plan-l is an inefficient plan
- RIM is a function to measure the inefficient mileage of plan - l
- I is a function to measure if a delivery point - d of plan - l creates an inefficient route.

Figure 6.10: Inefficient Plan Formula

6.4 Resolve Inefficient Plan

There are two approaches which have been proposed to resolve the inefficient plan issue. In the first approach, inefficient milages measurements within the evaluation function are incorporated and this is denoted as EVALFUNC. After an initial solution is created, we make changes to the solution until there is no conflict left. The idea is that an inefficient plan is likely to create conflict in delivery time because of the longer drive. Because of commercial sensitivity, it is not possible to report the details of the evaluation function, however, a simplified version of the evaluation function is shown in Equation 6.4. None of the other factors are weighted more than the inefficient score. The inefficient score is calculated using Equation 6.5, RIM and R are defined in section 6.3.

$$Eval(p) = 150 * ConflictScore + 100 * InefficientScore + OtherScore \tag{6.4}$$

$$InefficientScore = \frac{RIM(l)}{R(l, d_n)} \tag{6.5}$$

The second approach to resolve the inefficient plan issue is to use selection heuristics. This approach is denoted as HEURISTICS. There are three components for each heuristic: target election, candidate selection and operator. Each heuristic starts by select target and candidate shipments. After selection, heuristics apply operators with select target and candidates shipment in order to generate new solution. Inefficient shipment is measured by the RIM function as described in section 6.3. Two target selections, two operators and three candidate selections were chosen:

Target selections:

- Most inefficient shipment: shipments which have the highest inefficient mileage are selected
- Most conflicted shipment: shipments which have the highest conflict time are selected. Conflict time is measured by the difference between arrival time and re-

quired delivery time of shipment.

Operators:

- Reassign: move the target shipment to all possible positions in the candidate plan. The best position will be selected.
- Swap: swap the target shipment to all possible shipments in the candidate plan. The best pair will be selected.

Candidate selection:

- Same cluster: plans which contain at least one shipment from the same cluster with the target shipment are selected.
- Different cluster: plan which contain all shipments from different cluster with the target shipment are selected.

There are a number of differences to operator in Landa-Silva et al. (2011), candidate selection returns plans instead of shipments and all possible positions in the candidate plan are evaluated. Also different cluster candidate functions have no distance limit. New plan candidate functions are only applicable with the Reassign operator. It is designed to targets delivery points with restrictions and is normally for delivery windows outside of working hours. If a delivery plan is created and combined with shipments in both normal working hours and outside working hours, then the driver working hours will be long and a large proportion of the waiting time will result in additional cost. This will reduce the quality of the plan and is also likely to be rejected by the carrier. New heuristics are included to improve the planning process as in Algorithm 6.1. The original improved load building process given in Landa-Silva et al. (2011) is terminated when there are no delivery window conflicts remaining. Improved plans process and prioritise delivery window conflicts before attempts are made to improve the solution by eliminating inefficient plans. A combination of heuristics, in order to resolve delivery

window conflicts and inefficient plans, are shown in Algorithms 6.2 and 6.3. It is possible when resolving delivery window conflict that inefficient plans can also be resolved at the same time. However, this depends on operational requirements and inefficient plans can be accepted as a valid solution. Inefficient plans can also be considered as a soft constraint and in real life operations an inefficient plan can be accepted. For example, a plan can be classified as an inefficient plan but it will be more cost effective to pay the carrier additional costs than to create a separate plan for a single shipment.

Algorithm 6.1 Improve plan process after initial solution to resolve constraint violation

```
1: procedure IMPROVE PLAN PROCESS
2:     i ← 0
3:     while Has Conflict || i > MaxInteration do
4:         if IsBookingTimeWindowsConflict then
5:             Resolve Delivery Windows Conflict
6:         else
7:             Resolve Inefficient Plan
8:         end if
9:         i ←i +1
10:    end while
11: end procedure
```

Algorithm 6.2 Resolve delivery windows conflict process

```
1: procedure RESOLVE DELIVERY WINDOWS CONFLICT
2:     if !CanImproveByReassignMostConflictToSameCluster then
3:     else if !CanImproveBySwapMostConflictToSameCluster then
4:     else if !CanImproveByReassignMostConflictToDifferentCluster then
5:     else if !CanImproveBySwapMostConflictToDifferentCluster then
6:     else if !CanImproveByReassignToNewPlan then
7:     end if
8: end procedure
```

Algorithm 6.3 Resolve inefficient plan process

```
1: procedure RESOLVE INEFFICIENT
2:     if !CanImproveByReassignMostInefficientToSameCluster then
3:     else if !CanImproveBySwapMostInefficientToSameCluster then
4:     else if !CanImproveByReassignMostInefficientToDifferentCluster then
5:     else if !CanImproveBySwapMostInefficientToDifferentCluster then
6:     end if
7: end procedure
```

6.5 Evaluation of Clustering Algorithms Experiments

6.5.1 Experimental Set Up

Data sets were collected from the live database of 3T for three different warehouses in three different countries: Featherstone - United Kingdom (LUK), Pravia - Spain (LSP) and Pontivy - France (LFR). There were 2 experimental set ups which were carried out with different datasets: standard and relax. Standard data is the original data from 3T's database used in Landa-Silva et al. (2011). Relax data set contain the same data as Standard dataset however delivery point's booking windows is increased to full working hours (09:00 - 16:30). For each set up, two results were reported: the first result with an initial solution only and the second result including improvement processes from the initial solution.

Parameters were set as follows: $\alpha = 1.5$, $\beta = 0.5$ and maximum interation is 200. Each clustering algorithm was tested with at least 4 sets of parameters and the best results are reported. For each set up, shipment data of 15 days were randomly selected and the average result reported.

6.5.2 Experimental Results

6.5.2.1 Standard Data

Results of initialisation for standard tests are shown in Tables 6.2, 6.3 and 6.4. At LUK, which has the highest of number of shipments, the DBSCAN variation clustering

Cluster Algorithms	Total Loads	Vehicle Fill	Delivery Time Violation	Plan Mileage	Working Time	Total Cost	Inefficient Plan
DBSOR	**8.7**	**93.8**	12.75	270	638	**3515**	6.2
DBSM	10.7	83.3	6.96	166	387	3802	2.3
SLINK	11.3	87.7	8.67	216	411	4109	3.9
KMEAN	11.2	91.1	12.17	239	461	4345	5.3
EM	10.3	84.4	7.50	199	371	3889	3.0
DBSKM	10.5	85.5	**6.42**	**160**	**392**	3647	**1.8**

Table 6.2: Results of different clustering algorithms in LUK at initial solutions using standard data

Cluster Algorithms	Total Loads	Vehicle Fill	Delivery Time Violation	Plan Mileage	Working Time	Total Cost	Inefficient Plan
DBSOR	**13.3**	**92.7**	9.0	580	617	4094	3.2
DBSM	14.0	84.6	8.5	484	637	3315	0.5
SLINK	15.2	85.4	6.2	**455**	**405**	**2736**	**0.3**
KMEAN	13.8	92.6	9.4	526	474	4031	2.8
EM	13.6	92.4	**3.5**	535	481	4732	2.0
DBSKM	13.8	84.8	8.0	493	623	3647	0.4

Table 6.3: Results of different clustering algorithms in LPS at initial solutions using standard data

algorithm outperforms the other clustering algorithms. DBSKM has the lowest cost, violation and working time and DBSOR has the highest vehicle utilisation. However, in LSP and LFR, other types of clustering techniques give better results. In LSP, SLINK has the lowest driving distance, driving time, cost and inefficient plan. EM produces plans with the lowest violation. Similar to LUK, DBSOR has the lowest load build with the highest utilisation however more delivery time violations and inefficient plans are generated. There is no significant difference in the performances of DBSOR, DBSM and DBSKM. This is due to the distribution and quantity of shipments and only small clusters are formed. In LFR, no plan is created for all cluster algorithms. This is because the load building heuristic by Landa-Silva et al. (2011) only creates plans when there are FTL or LTL. However, as shown in Table 6.1, there are fewer FTL and LTL per day due to changes in business demands.

Tables 6.5, 6.6 and 6.7 summarise the final results of the optimisation with standard data. The improve plan process reduced the number of conflicts therefore there was no delivery window violation in the final result. The main trend from the initial to the final results is a cost increase. In order to resolve violation, shipments with very strict booking slots (i.e. 08:30 to 09:00) have to be delivered separately and results in a

Cluster Algorithms	Total Loads	Vehicle Fill	Delivery Time Violation	Plan Mileage	Working Time	Total Cost	Inefficient Plan
DBSOR	0.4	28.6	**1.5**	384	604	330	0.4
DBSM	0.4	28.6	**1.5**	384	604	330	0.4
SLINK	**0.8**	29.6	**0.8**	**187**	235	**302**	**0.0**
KMEAN	0.7	38.4	1.7	278	353	387	0.3
EM	0.7	**41.4**	2	217	330	330	**0.0**
DBSKM	0.4	28.6	**1.5**	384	604	330	0.4

Table 6.4: Results of different clustering algorithms in LFR at initial solutions using standard data

significant cost increase. In LUK, the result from DBSKM has the lowest cost and the lowest cost increment from the initial state. From Table 6.2, DBSKM, DBSM and EM have a low delivery time violation. However, improved plan processes work better with the DBSKM cluster therefore the cost increase in DBSKM is the smallest. DBSOR has the highest vehicle utilisation and cost in the initial state, but it also has the highest delivery time violation. The final result using DBSOR has the highest cost with lower vehicle utilisation. This indicates less violation from the initial solution and is likely to give a better cost in the final result. The quality of initial load building can also be seen when comparing changes in total loads and plan mileages. From Tables 6.2 to 6.5, total loads and plan mileages increase due to resolving violation, except for DBSOR where the number of loads increased but the plan mileage decreased. The clusters using DBSOR are larger therefore there are fewer numbers of clusters for reassignment or to swap operators. In this case, conflict shipments are assigned to a new plan. When considering an inefficient plan, DBSKM and DBSM have the best results compared to the other algorithms. Since a business focussed solution is required, total cost is the most important factor in the final result. DBSKM gives the best result with the lowest cost and it also has the lowest inefficient plan which is more acceptable for the human planner.

In LSP, a similar correlation occurs between initial delivery time violation and total cost. With the lowest violation in the initial solution, EM produces plans with the lowest cost after resolved delivery time violation. SLINK, KMEAN and EM have a high utilisation which is different from LUK. LSP has fewer shipments than LUK therefore

Cluster Algorithms	Total Loads	Vehicle Fill	Delivery Time Violation	Plan Mileage	Working Time	Total Cost	Inefficient Plan
DBSOR	13.1	79.5	0	257	**475**	4758	7.5
DBSM	13.9	73.1	0	**194**	586	4542	4.8
SLINK	12.7	80.5	0	228	526	4635	5.7
KMEAN	13.7	79.6	0	272	536	4726	7.9
EM	12.3	**84.4**	0	299	571	4389	4.0
DBSKM	**12.2**	80.7	**0**	198	517	**3904**	**3.9**

Table 6.5: Results of different clustering algorithms in LUK at final solution using standard data

it is easier to be grouped in KMEAN and EM. However, the performance of KMEAN and EM are dependent on parameter selection. If the reported result is an average result with different parameters then KMEAN and EM are no better than the others. DBSOR, DBSM and DBSKM have similar results because of the shipment distribution, as seen in Figure 6.1. There is only a high density of shipments around the Barcelona region and overall fewer shipments compared to LUK. Therefore there is no significant difference between DBSOR, DBSKM and DBSM. For LSP, the number of inefficient plans is very low across the different approaches. This is due to the distance from the source of shipment to the destination which is much longer than in LUK. Therefore carriers can only make a smaller number of drops. The inefficient plan measurement (Figure 6.10) gives the first drop direction as the main direction. If there are a low number of drops then it is not easy to identify if the route is acceptable unless there is a significant change in the direction of the route. However, because of the geographic position of LSP which is to one side of Spain, it is very unlikely to have loads going via different directions.

In LFR, the final result is very similar to the initial state as only a very small number of plans are created with small conflicts. With a small number of shipments and shipments which are far apart from each other, new plans are normally created to resolve the conflict. From discussions with human planners, the current load building method is not suitable for LFR for a data set with a small number of shipments.

Cluster Algorithms	Total Loads	Vehicle Fill	Delivery Time Violation	Plan Mileage	Working Time	Total Cost	Inefficient Plan
DBSOR	16.6	76.8	0	552	694	5663	3.7
DBSM	16.4	77.5	0	**418**	**552**	5813	1.8
SLINK	**15.8**	78.7	0	520	674	5242	2.4
KMEAN	16.4	77.9	0	547	743	5549	4.2
EM	15.3	**79.1**	0	564	684	**5131**	3.7
DBSKM	16.1	77.0	**0**	448	652	5341	**1.4**

Table 6.6: Results of different clustering algorithms in LSP at final solution using standard data

Cluster Algorithms	Total Loads	Vehicle Fill	Delivery Time Violation	Plan Mileage	Working Time	Total Cost	Inefficient Plan
DBSOR	0.4	15.6	0	108	364	350	**0.0**
DBSM	0.4	12.6	0	78	**201**	370	**0.0**
SLINK	0.8	**15.7**	0	187	365	**332**	0.2
KMEAN	0.7	11.0	0	278	505	397	0.1
EM	0.7	13.4	0	97	411	335	0.1
DBSKM	0.4	11.6	0	**61**	215	346	**0.0**

Table 6.7: Results of different clustering algorithms in LFR at final solution using standard data

6.5.2.2 Relax Data

Table 6.8, 6.9 and 6.10 show the results of the same set of data with extended delivery windows. Standard experiments show a correlation between delivery time violation and final cost. The less conflict in the initial state, the closer the initial cost is to the final cost. This experiment investigated the effect of delivery windows on cost and the selection of clustering algorithms. This is critical to business operation as delivery windows are contracted with the customer. However, if the effect of extended delivery windows is significant for a business then a decision can be made to re-negotiate the contract with the customer. Compared to previous experiments, changes occur in all three sites. Firstly, delivery time violation is decreased, as predicted. Secondly, vehicle utilisation is increased because larger sub-points are formed during sub-point generation. More about sub-point generation can be found in Landa-Silva et al. (2011). Thirdly, increased delivery windows reduce time constraints and allow a shipment in one region to be treated as a single drop in the optimisation process. The distance of a plan, working time and inefficient plan are decreased or at least similar compared to those in the standard experiment. This is partly because of sub-point generation improvement. Moreover, shipments cannot be

Cluster Algorithms	Total Loads	Vehicle Fill	Delivery Time Violation	Plan Mileage	Working Time	Total Cost	Inefficient Plan
DBSOR	**10.9**	**94.1**	3.83	269	472	4579	5.9
DBSM	11.2	84.1	0.7	**169**	387	3934	2.8
SLINK	11.4	88.8	3.3	224	404	4327	4.2
KMEAN	11.3	92.7	2.7	246	460	4425	6.0
EM	11.5	95.6	2.3	269	387	4059	5.2
DBSKM	11.1	85.8	**0.2**	180	**371**	**3849**	**2.3**

Table 6.8: Results of different clustering algorithms in LUK at initial solutions using relax windows data

Cluster Algorithms	Total Loads	Vehicle Fill	Delivery Time Violation	Plan Mileage	Working Time	Total Cost	Inefficient Plan
DBSOR	13.8	**91.5**	3.1	586	692	4932	3.1
DBSM	**12.8**	85.8	0.3	**391**	**465**	4235	**0.4**
SLINK	15.1	79.6	**0.2**	452	499	**3630**	**0.4**
KMEAN	13.7	90.6	1.3	528	640	5056	2.3
EM	14.0	89.3	1.3	522	628	5432	2.0
DBSKM	12.9	87.0	0.4	402	469	4236	0.5

Table 6.9: Results of different clustering algorithms in LSP at initial solutions using relax windows data

grouped in sub-points but can be delivered with shipments in the same region.

Compared to the initial solution of the standard data set, DBSOR still has the lowest number of loads created with the highest vehicle utilisation. DBSKM has the lowest violation in the initial state. Inefficient plan in the initial state has a slight increase because of relaxed booking windows. This allows a carrier to have more time to deliver to more destinations. It is important to note that the result for initial DBSKM has a lower cost than the final state of the standard booking windows with a lower delivery time violation. Feedback from business planners suggests the results of the DBSKM initial state can be used as one possible solution without any further modification. One factor not shown in the table is run time where the average run time of the optimisation output to the initial state is 5 minutes and the time for full optimisation is 10 minutes.

Cluster Algorithms	Total Loads	Vehicle Fill	Delivery Time Violation	Plan Mileage	Working Time	Total Cost	Inefficient Plan
DBSOR	0.4	29.4	**0**	393	604	330	0.4
DBSM	0.4	29.4	**0**	393	604	330	0.4
SLINK	**0.8**	30.2	**0**	**297**	345	**302**	**0.0**
KMEAN	0.7	38.4	0	297	367	387	0.3
EM	0.7	**41.4**	0	217	330	330	**0.0**
DBSKM	0.4	29.4	**0**	393	604	330	0.4

Table 6.10: Results of different clustering algorithms in LFR at initial solutions using relax windows data

Cluster Algorithms	Total Loads	Vehicle Fill	Delivery Time Violation	Plan Mileage	Working Time	Total Cost	Inefficient Plan
DBSOR	11.4	90.5	0	271	478	4907	6.8
DBSM	**10.9**	84.1	0	**173**	401	4332	2.9
SLINK	11.7	86.8	0	231	419	4327	4.5
KMEAN	12.3	86.1	0	239	434	4425	6.7
EM	12.2	**92.2**	0	188	**365**	4059	5.5
DBSKM	10.9	85.5	**0**	161	384	**3787**	**2.6**

Table 6.11: Results of different clustering algorithms in LUK at final solution using relax windows data

Cluster Algorithms	Total Loads	Vehicle Fill	Delivery Time Violation	Plan Mileage	Working Time	Total Cost	Inefficient Plan
DBSOR	14.8	**85.4**	0	586	573	5877	3.1
DBSM	**12.8**	85.8	0	**391**	**392**	4924	**0.4**
SLINK	15.1	79.5	**0**	452	453	**4505**	**0.4**
KMEAN	14.3	86.2	0	528	542	4689	3.0
EM	14.7	85.2	0	522	527	4703	2.3
DBSKM	13.0	87.0	0	402	402	4832	0.5

Table 6.12: Results of different clustering algorithms in LSP at final solution using relax windows data

The majority of the run time is spent in distance query time between all delivery points and pricing from different carriers. However, it is important in business to have a balance between the solution quality and an acceptable run time. For a business with a known order then run time is not a critical factor, but in particular logistic models this is not the case. For both LSP and LFR, SLINK gives the lowest costs with EM and KMEAN having similar results. At LFR when the booking time window is extended, there is no conflict at the initial state.

6.6 Resolve Inefficient Plan Experiment

6.6.1 Experimental Set Up

Similar to the experiment of the clustering performance, there are 2 experimental set ups: Standard data and Relax. The data set was collected from 3T's live database for the United Kingdom (LUK). For each set up, evaluation integration and inefficient heuristic approaches were evaluated. Only the final results are reported because the initial states of the two approaches were identical. The clustering algorithm DBSKM was selected for

this experiment based on its performance from section 6.5.2. The maximum iteration for each run was set to 200.

6.6.2 Experimental Results

Table 6.13 shows the results for evaluation integration and combination heuristics for the standard set of data. The evaluation function approach does have an postive effect on reducing the number of inefficient plans when compared with the original improve plan process. Combine heuristics approach has the best performance in creating efficient plans. It has a slight increase in cost in both evaluation and the heuristic approach. However, the effect of inefficient plan reduction has significant business impact than cost increase. With the relax data, fewer plans are created because shipments can be grouped together to increase vehicle utilisation. The evaluation function approach generates more inefficient plans than the heuristic approach. Relax windows gives more options to load vehicles. This creates longer routes and a higher chance of inefficient plans. However, the evaluation function does not have the capability to swap or reassign shipments. The heuristic approach not only takes advantage of the opportunity to maintain vehicle utilisation but also reduces the number of inefficient plans. After all experiments, the output was demonstrated to 3T planners to evaluate the practicality of the solution. By introducing inefficient measure and combine DBSCAN+KMEAN clustering, the final result is feasible and the acceptances are significant improved. The DBSCAN+KMEAN combination is now the preferred clustering algorithm. It is also important to point out that without taking into account any knowledge of a carrier's costing structure, the proposed planning process can produce plans with no delivery window conflicts and near-zero inefficient plans at a lower cost. Inefficient plans were also investigated and identified as normally having a higher cost compared to the other plans. This shows that costings are dependent on driving mileage and fuel price. A transport plan with a straight route is likely to have a lower carrier cost.

Approach	Total Loads	Vehicle Fill	Delivery Time Violation	Plan Mileage	Working Time	Total Cost	Inefficient Plan
EVALFUNC	14.5	77.8	0	155	450	3218	1.3
HEURISTICS	14.5	77.9	0	149	423	3229	0.8
EVALFUNC (Relax Windows)	12.4	88.5	0	175	418	3003	2.0
HEURISTICS (Relax Windows)	12.9	85.1	0	164	394	3106	0.3

Table 6.13: LUK - Final Results - Resolve Inefficient Plan

6.7 Conclusion

In this chapter, the Single-Customer Multi-Carrier planning problem was updated to include a business requirement. Different clustering algorithms and their performance and compatibility with current HHLP approaches in different scenarios were investigated. Inefficient planning issues from the application of HHLP in a business were identified. A formula to identify inefficient plans was designed as well as different approaches to eliminate inefficient plans. From the experiments, the density based clustering algorithms DBSCAN and its variations were found to be suitable with a large number of shipments cases(LUK, LSP). However, distribution based (EM), centroid based (KMEAN) and connectivity based (SLINK) clustering were more suitable with a smaller number of shipments cases. With standard data, the original DBSCAN clustering algorithm gave the best utilisation and lowest cost. However this led to creation of a large number of inefficient plans and affected the feasibility of transforming the automated plan into actual delivery plans. DBSKM offered a better carrier plan with only a small increase in cost.

When the data set's constraint was relaxed, DBSKM reacted positively and its performance significantly improved with the lowest cost. A gap was also identified in the current method where data containing a large number of small shipments resulted in HHLP becoming ineffective. Two approaches to eliminate inefficient plans were proposed: integrated into evaluation function and heuristic specifically targeting the issue plan. The heuristic approach was more efficient in resolving the inefficient plans compared to the integrated evaluation function. The heuristic approach also offered an excellent effect on making the delivery plan more feasible leading to a lower cost of delivery. This reflects

a relationship between the carrier cost and the efficiency of the plan.

7 Conclusions and Future Work

7.1 Conclusions

As computational power has become more accessible, the demand for automated optimisation in transportation logistics has increased significantly. One of the most common transportation logistic optimisation problems is the maximisation of vehicle utilisation. Therefore, three-dimensional cutting and packing problems have received increased attention in recent years as they arise when seeking to optimise vehicle utilisation. When three-dimensional cutting and packing problems are compared to one- or two-dimensional packing problems, a number of areas which require improvement can be identified.

A range of algorithmic methods and frameworks have been developed for different classes of three-dimensional strip packing problems under different constraints. In the research presented in this PhD thesis, modifications to the 3BF framework were proposed and then extended with a "look-ahead" approach. Firstly, block generation processes (Single Mode and Mix Mode) were proposed in order to combine suitable boxes and create larger blocks which were suitable for a best-fit methodology. Secondly, a procedure of position re-allocation was presented, to reduce potential space lost. Thirdly, an overhead estimation approach was implemented with a modified best-fit heuristic (OH-3BFBL).

The OH-3BFBL heuristic showed an improved performance, with stronger heterogeneous instances, when compared to the performance shown in standard data sets. Increasing the number of box types improved the result of Single Mode, however, Mix Mode still had a better average utilisation across all instances. For instances containing

a large quantity of each box type, OH-3BFBL demonstrated a significant improvement compared to the best known approaches with the same computational run time. To improve the performance of OH-3BFBL, recursive block generation and improved processes for candidate point generation were implemented in OH-3BFBLEX. This modification allowed OH-3BFBLEX to explore more possible packing positions and therefore provide enhanced utilisation in strong heterogeneous cases. However, a negative impact was observed in weak heterogeneous cases due to ambiguous blocks and their positions.

Following our initial research into 3D-SPP, the 3D-SPP problem with a stability constraint was considered next. There are a number of different criteria available for stability constraints and the fully supported stability was selected for this research. Adjustments to the block generation process were made in order to maintain the stability constraint. An adapted best-fit methodology and an additional best-support heuristic were considered and implemented. The performance of previous 3BF heuristics and best-support heuristics with stability requirements were evaluated. From initial experiments, Maximum Contact criterion was able to produce a good result with best-fit or best-support heuristics. Maximum Contact was selected to be the main criterion to select blocks and combine with overhead estimation (OH-3BFMC). The OH-3BFMC approach showed an improved result in stronger heterogeneous instances and gave a best overall result when compared to best known single thread approaches. When compared to multi-threaded implementation, OH-3BFMC still resulted in higher utilisation for instances with the largest number of box types and the average performance was not far off the best known result.

Following the collaboration with 3T Logistics Ltd, this research was extended to a real life Pallet Space Equivalent problem (PSE) as a variation of the 3D-SPP. To the best of our knowledge, no previous work has been carried out for the 3D-SPP using data from real-world cases and live operations. With support from 3T, the operational packing process and its relationship with 3T's businesses requirements was observed. From that

observation, the PSE problem and utilisation evaluation was defined. The PSE problem models packing of an order with a combination of PALLET, BOX or BAG items. Each item has rotation, quantity and a stackability constraint and all items need to be placed. However, in constrast to the utilisation in 3D-SPP using container length, the utilisation in the PSE problem is only concerned with the floor space required for all items to be packed. Due to the stability and stackability constraint, a new method to evaluate PSE utilisation was proposed where the stackability of pallets was considered to estimate the optimum floor space required. New ways to evaluate utilisation helped to provide a better picture of the performances of the packing methods and was critical for deciding which method would be good or suitable enough for live operations. A range of heuristics were adapted from 3D-SPP heuristics and evaluated for the PSE problem. Maximum Volume and Maximum Contact heuristics gave the best results across all instances. This is a similar trend compared to the work with the 3D-SPP problem where the best-fit heuristic was more likely to produce good results compared to best-support heuristics. The experiment also showed a correlation between the proportion of pallet item type and utilisation. The utilisation evaluation was more effective when there were more pallet items.

Finally, 4PL transport planning with new operational requirements was studied. A single customer multi-carrier planning problem was re-visited and the problem description was updated to reflect new business requirements. One important component of the current solution method is clustering. DBSCAN clustering algorithms were adapted and gave good initial solutions. Different clustering algorithms from the literature were reviewed and a hybrid of clustering techniques was proposed. Experiments indicated that for different instances the clustering algorithm selection can affect the planning result. Advantages and disadvantages of different clustering algorithms for different customer scenarios were investigated and analysed. For instance, with high number of shipments, density-based clustering algorithms perform better. Centroid based-algorithms are more

suitable for instances with a small number of shipments across a larger area. With large numbers of shipments, DBSOR normally offered good vehicle utilisation but it also gave the highest number of inefficient plans. DBSKM offered a lower vehicle utilisation and there were fewer inefficient plans created so more feasible delivery plans were achieved. One of the main constraints of the planning problem is that delivery windows have to be met. The effect of extending small delivery windows was investigated and experiments showed that a more flexible delivery window has a positive effect in reducing transport costs. This finding offered critical information for 3T and its customers. The customer can then identify potential savings in negotiation with its customer delivery windows. After obtaining feedback from what happens in live operations, a new attribute of plan measurement was introduced - inefficient plan measurement and local search operators were used to eliminate inefficient plans from the final solution. Inefficient plans were described by operational staff as a vehicle route going backwards or with a very sharp bend. In order to resolve inefficient plans, two approaches were proposed: integrated inefficient plan in evaluation function and inefficient specific local search operator. The specific local search operator achieved a better performance compared to other approaches. The overall result presented a significant increase in plan quality especially plan acceptance and all modifications are now implemented in the current system being used at 3T Logistics Ltd.

With this detailed approach and experiment, the reader can easily have an out-of-the-box method for the three-dimensional strip packing. With additional constraints, the overhead estimation approach can offer a quick and simple-to-implement option to improve the solution. For audiences who are interested in 4PL transport planning, the clustering algorithms review can give some guidance as to which cluster technique is suitable depending on individual requirements. It also gives information about new factors to measure and possible solutions to improve planning quality.

7.2 Future Work

In this section, a number of research areas that were beyond the scope of this thesis are identified. They involve practical requirements and hence may be of interest for future research.

- Chapter 3: block generation and overhead estimation provide a simple framework for the three-dimensional strip packing problem. A different block generation strategy which incorporates information about the boxes (i.e. when the number of boxes is large then build two layers of box instead of single layer) would be useful. Overhead estimation can be extended to reduce computational effort by avoiding ambiguous packing where different blocks with similar dimensions have the same final result.

- Chapter 4: currently, block generation employs a conservative approach. However, it is possible to have a further study on the effect of internal loss inside the generated blocks. In practice, it is possible to pack an item which is not fully supported. For example, a large foam box cannot be packed underneath other items, however it is perfectly acceptable to have this foam box overhanging other items within reasonable practical conditions. Therefore, a study on non-fully supported stability would have practical relevance.

- Chapter 5: current utilisation evaluation takes into account the stackability of pallet type handling unit. Further improvement could be made to include other types of handling units and their stackability. For example, BOX or BAG items which cannot be combined with any other item or itself can be used to estimate floor space. Heuristic approaches have been introduced in this thesis. Due to business requirements, overhead estimation was not included in the scope of this project. However, it is possible to apply overhead estimation to improve the results. Another possible option is to integrate a meta-heuristic into the packing process.

- Chapter 6: the current planning approach is based on the assumption that shipments are ready and can be collected at any time. Due to requirements from the customer, a shipment collection time can also have time windows. For example, a shipment can be collected at anytime but some can only be collected after 12:00 due to manufacture time. If all shipments are collected at 12:00 then some shipments cannot be delivered on time. One simple solution is to have a dedicated vehicle which is very expensive for late collected shipments. A study in automated transport planning with multiple time windows could be carried out to accommodate this new requirement. Another practical requirement raised by a customer is having a maximum number of vehicles that can be loaded at the same time. For example, a warehouse has a limited number of bays for loading trucks. If the number of plans to collect at one point in time is more than the number of bays then it is not practical. The current approach did not take this factor into account. Also, due to the complexity of real-world live operationss it is possible to add additional constraints to the problem in the future. Research into the application of a wide range of meta-heuristics and hybrid appraoches for transport planning would have great potential for future research and development.

References

Agbegha, G. Y., Ballou, R. H. & Mathur, K. (1998), 'Optimizing auto-carrier loading', *Transportation Science* **32**(2), 174–188.

Agrawal, R., Gehrke, J., Gunopulos, D. & Raghavan, P. (1998), 'Automatic subspace clustering of high dimensional data for data mining applications', *SIGMOD Rec.* **27**(2), 94–105.

AliReza, S. & Mehdi, Y. (2010), 'Lsp, 3pl, llp, 4pl which one come in useful for outsourcing cycle![online]'.
URL: *http://www.ieee.org/documents/lsp_ 3pl.pdf*

Allen, S., Burke, E. & Kendall, G. (2011), 'A hybrid placement strategy for the three-dimensional strip packing problem', *European Journal of Operational Research* **209**(3), 219 – 227.

Ankerst, M., Breunig, M. M., peter Kriegel, H. & Sander, J. (1999), Optics: Ordering points to identify the clustering structure, ACM Press, pp. 49–60.

Araya, I., Neveu, B. & Riff, M.-C. (2008), An efficient hyperheuristic for strip-packing problems, *in* C. Cotta, M. Sevaux & K. Sörensen, eds, 'Adaptive and Multilevel Metaheuristics', Vol. 136 of *Studies in Computational Intelligence*, Springer Berlin Heidelberg, pp. 61–76.

Baker, B. S., Coffman Jr, E. G. & Rivest, R. L. (1980), 'Orthogonal packings in two dimensions', *SIAM Journal on Computing* **9**(4), 846–855.

Baker, E. & Schaffer, J. (1986), 'Solution improvement heuristics for the vehicle routing and scheduling problem with time window constraints.', *American Journal of Mathematical and Management Sciences* **6**(3), 261–300.

Balakrishnan, N. (1993), 'Simple heuristics for the vehicle routeing problem with soft time windows', *Journal of the Operational Research Society* **44**(3), 279–287.

Bansal, N., Han, X., Iwama, K., Sviridenko, M. & Zhang, G. (2007), Harmonic algorithm for 3-dimensional strip packing problem, *in* 'Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms', Society for Industrial and Applied Mathematics, pp. 1197–1206.

Beaumont, L. (2004), Key performance indicators for the pallet distribution network sector, Technical report, The Logistics Business Ltd.

Bedeman, M. & Gattorna, J. L. (2003), 'Third-and fourth-party logistics service providers', *Gower Handbook of Supply Chain Management* **5**, 469–485.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I. & Laporte, G. (2007), 'Static pickup and delivery problems: aÂ classification scheme and survey', *TOP* **15**(1), 1–31.

Berger, J. & Barkaoui, M. (2003), 'A route-directed hybrid genetic approach for the vehicle routing problem with time windows', *INFOR* **41**, 179–194.

Berkhin, P. (2006), A survey of clustering data mining techniques, *in* 'Grouping multidimensional data', Springer, pp. 25–71.

Bischoff, E. E. & Marriott, M. D. (1990), 'A comparative evaluation of heuristics for container loading', *European Journal of Operational Research* **44**(2), 267–276.

Bischoff, E. E. & Ratcliff, M. S. W. (1995), 'Issues in the development of approaches to container loading', *Omega* **23**(4), 377–390.

Bortfeldt, A. (2006), 'A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces', *European Journal of Operational Research* **172**(3), 814 – 837.

Bortfeldt, A. & Gehring, H. (1998), 'Ein tabu search-verfahren für containerbeladeprobleme mit schwach heterogenem kistenvorrat', *OR Spectrum* **20**(4), 237–250.

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

Bortfeldt, A., Gehring, H. & Mack, D. (2003), 'A parallel tabu search algorithm for solving the container loading problem', *Parallel Computing* **29**(5), 641–662.

Bortfeldt, A. & Jungmann, S. (2012), 'A tree search algorithm for solving the multi-dimensional strip packing problem with guillotine cutting constraint', *Annals of Operations Research* **196**, 53–71.

Bortfeldt, A. & Mack, D. (2007), 'A heuristic for the three-dimensional strip packing problem', *European Journal of Operational Research* **183**(3), 1267 – 1279.

Bortfeldt, Andreas; Gehring, H. (1999), Two metaheuristics for strip packing problems, *in* 'Proceedings of the 5th International Conference of the Decision Sciences Institute', Vol. 2, pp. 1153–1156.

Bouthillier, A. L. & Crainic, T. G. (2005), 'A cooperative parallel meta-heuristic for the vehicle routing problem with time windows', *Computers & Operations Research* **32**(7), 1685 – 1708.

Bräysy, O. (2001), Local Search and Variable Neighborhood Search Algorithms for the Vehicle Routing Problem with Time Windows, PhD thesis, Universitas Wasaensis.

Bräysy, O. (2003*a*), 'Fast local searches for the vehicle routing problem with time windows', *INFORM* **41**, 179–194.

Bräysy, O. (2003*b*), 'A reactive variable neighborhood search for the vehicle-routing problem with time windows', *INFORMS Journal on Computing* **15**(4), 347–368.

Bräysy, O. & Gendreau, M. (2001*a*), 'Metaheuristics for the vehicle routing problem with time windows', *Report STF42 A* **1025**.

Bräysy, O. & Gendreau, M. (2001*b*), 'Route construction and local search algorithms for the vehicle routing problem with time windows', *Sintef Report, STF42 A* **1024**.

Bräysy, O. & Gendreau, M. (2005*a*), 'Vehicle routing problem with time windows, part i: Route construction and local search algorithms', *Transportation Science* **39**(1), 104–118.

Bräysy, O. & Gendreau, M. (2005*b*), 'Vehicle routing problem with time windows, part ii: Metaheuristics', *Transportation Science* **39**(1), 119–139.

Bremermann, H. J. (1962), 'Optimization through evolution and recombination', *Self-organizing systems* pp. 93–106.

Brown, G. G. & Ronen, D. (1997), 'Consolidation of customer orders into truckloads at a large manufacturer', *J Oper Res Soc* **48**(8), 779–785.
URL: *http://dx.doi.org/10.1057/palgrave.jors.2600430*

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E. & Qu, R. (2010), 'Hyper-heuristics: A survey of the state of the art', *Computer Science Tech. Rep. NOTTCS-TR-SUB-0906241418–2747, University of Nottingham* .

Burke, E. K. & Kendall, G. (2005), *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, Springer.

Burke, E. K., Kendall, G. & Whitwell, G. (2004), 'A new placement heuristic for the orthogonal stock-cutting problem', *Oper. Res.* **52**(4), 655–671.

Burke, E. K., Petrovic, S. & Qu, R. (2006), 'Case-based heuristic selection for timetabling problems', *Journal of Scheduling* **9**(2), 115–132.

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P. & Schulenburg, S. (2003), 'Hyper-heuristics: An emerging direction in modern search technology', *Handbook of meta-heuristics* pp. 457–474.

Caputo, A., Fratocchi, L. & Pelagagge, P. (2006), 'A genetic approach for freight transportation planning', *Industrial Management & Data Systems* **106**(5), 719–738.

Caseau, Y. & Laburthe, F. (1999), 'Heuristics for large constrained vehicle routing problems', *Journal of Heuristics* **5**(3), 281–303.

Chazelle, B. (1983), 'The bottomn-left bin-packing heuristic: An efficient implementation', *IEEE Trans. Comput.* **32**(8), 697–707.

Chu, C.-W. (2005), 'A heuristic algorithm for the truckload and less-than-truckload problem', *European Journal of Operational Research* **165**(3), 657 – 667.

Clarke, G. & Wright, J. (1964), 'Scheduling of vehicles from a central depot to a number of delivery points', *Operations research* **12**(4), 568–581.

Coffman, Jr., E., Garey, M. & Johnson, D. (1978), 'An application of bin-packing to multiprocessor scheduling', *SIAM Journal on Computing* **7**(1), 1–17.

Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M. M. & Soumis, F. (2002), 'Vrp with time windows', *The vehicle routing problem* **9**, 157–193.

Cordone, R. & Calvo, R. W. (2001), 'A heuristic for the vehicle routing problem with time windows', *Journal of Heuristics* **7**(2), 107–129.

Cormen, T. H., Stein, C., Rivest, R. L. & Leiserson, C. E. (2001), *Introduction to Algorithms*, 2nd edn, McGraw-Hill Higher Education.

Crainic, T. G., Perboli, G. & Tadei, R. (Summer 2008), 'Extreme point-based heuristics for three-dimensional bin packing', *INFORMS Journal on Computing* **20**(3), 368–384.

Croes, G. A. (1958), 'A method for solving traveling-salesman problems', *Operations Research* **6**(6), 791–812.

David Simchi-Levi, Xin Chen, J. B. (2004), *Theory, Algorithms, and Applications for Logistics and Supply Chain Management*, Springer.

Davies, A. P. & Bischoff, E. E. (1999), 'Weight distribution considerations in container loading', *European Journal of Operational Research* **114**(3), 509–527.

Defays, D. (1977), 'An efficient algorithm for a complete link method', *The Computer Journal* **20**(4), 364–366.

Dempster, A. P., Laird, N. M. & Rubin, D. B. (1977), 'Maximum likelihood from incomplete data via the EM algorithm', *Journal of the Royal Statistical Society: Series B* **39**, 1–38.

Desrosiers, J., Dumas, Y., Solomon, M. M. & Soumis, F. (1995), Chapter 2 time constrained routing and scheduling, *in* C. M. M.O. Ball, T.L. Magnanti & G. Nemhauser, eds, 'Network Routing', Vol. 8 of *Handbooks in Operations Research and Management Science*, Elsevier, pp. 35 – 139.

DFT, D. O. T. (2011), 'Road freight statistics'.

Dullaert, W. (2000), 'Impact of relative rout length on the choice of time insertion criteria for insertion heuristics for the vehicle routing problem with time windows'.

Dullaert, W. & Bräysy, O. (2003), 'Routing relatively few customers per route', *Top* **11**(2), 325–336.

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

Eilon, S. & Christofides, N. (1971), 'The loading problem', *Management Science* **17**(5), 259–268.

Erera, A. L., Hewitt, M., Savelsbergh, M. W. & Zhang, Y. (2013), 'Creating schedules and computing operating costs for {LTL} load plans', *Computers & Operations Research* **40**(3), 691 – 702. Transport Scheduling.

Ester, M., peter Kriegel, H., S, J. & Xu, X. (1996), A density-based algorithm for discovering clusters in large spatial databases with noise, AAAI Press, pp. 226–231.

Everitt, B., Landau, S., Leese, M. & Stahl, D. (2011), *Cluster Analysis*, Wiley series in probability and statistics, 5 edn, Wiley.

Foisy, C. & Potvin, J.-Y. (1993), 'Implementing an insertion heuristic for vehicle routing on parallel hardware', *Computers & OR* **20**(7), 737–745.

Fraser, A. S. (1960), 'Simulation of genetic systems by automatic digital computers vi. epistasis', *Australian Journal of Biological Sciences* **13**(2), 150–162.

Freight Transport Association, P. U. (2012), *The Logistics Report 2012*, Freight Transport Association. http://www.fta.co.uk/_galleries/downloads/email_news/logistics_report2012.pdf.

Garrido, P. & Castro, C. (2009), Stable solving of cvrps using hyperheuristics, *in* 'Proceedings of the 11th Annual conference on Genetic and evolutionary computation', ACM, pp. 255–262.

Garrido, P. & Riff, M. C. (2010), 'Dvrp: a hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic', *Journal of Heuristics* **16**(6), 795–834.

Gehring, H. & Bortfeldt, A. (1997), 'A genetic algorithm for solving the container loading problem', *International Transactions in Operational Research* **4**(5-6), 401–418.

Gendreau, M., Hertz, A. & Laporte, G. (1992), 'New insertion and postoptimization procedures for the traveling salesman problem', *Oper. Res.* **40**(6), 1086–1094.

Gendreau, M., Laporte, G. & Séguin, R. (1996), 'Stochastic vehicle routing', *European Journal of Operational Research* **88**(1), 3 – 12.

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

Gillett, B. E. & Miller, L. R. (March/April 1974), 'A heuristic algorithm for the vehicle-dispatch problem', *Operations Research* **22**(2), 340–349.

Glover, F. (1992), 'New ejection chain and alternating path methods for traveling salesman problems', *Computer Science and Operations Research* **449**.

Glover, F. (1996), 'Ejection chains, reference structures and alternating path methods for traveling salesman problems', *Discrete Applied Mathematics* **65**(1), 223–253.

Glover, F. & McMillan, C. (1986), 'The general employee scheduling problem. an integration of ms and ai', *Computers & operations research* **13**(5), 563–573.

Goldberg, D. E. & Holland, J. H. (1988), 'Genetic algorithms and machine learning', *Machine Learning* **3**(2), 95–99.

Golden, B. L., Raghavan, S. & Wasil, E. A. (2008), *The Vehicle Routing Problem: Latest Advances and New Challenges: latest advances and new challenges*, Vol. 43, Springer.

Guha, S., Rastogi, R. & Shim, K. (2000), 'Rock: A robust clustering algorithm for categorical attributes', *Information systems* **25**(5), 345–366.

Guha, S., Rastogi, R. & Shim, K. (2001), 'Cure: an efficient clustering algorithm for large databases', *Information Systems* **26**(1), 35 – 58.

Günther, H.-O. & Seiler, T. (2009), 'Operative transportation planning in consumer goods supply chains', *Flexible Services and Manufacturing Journal* **21**(1-2), 51–74. **URL:** *http://dx.doi.org/10.1007/s10696-010-9060-5*

Hartigan, J. A. & Wong, M. A. (1979), 'Algorithm AS 136: A k-means clustering algorithm', *Applied Statistics* **28**(1), 100–108.

Ho, S. C. & Haugland, D. (2002), 'A tabu search heuristic for the vehicle routing problem with time windows and split deliveries', *Computers and Operations Research* **31**, 1947–1964.

Hopper, E. (2000), *Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods*, University of Wales. Cardiff.

Hopper, E. & Turton, B. (2001), 'A review of the application of meta-heuristic algorithms to 2d strip packing problems', *Artificial Intelligence Review* **16**(4), 257–300.

Huang, W. & He, K. (2009), 'A caving degree approach for the single container loading problem', *European Journal of Operational Research* **196**(1), 93–101.

Ibaraki, T., Imahori, S., Kubo, M., Masuda, T., Uno, T. & Yagiura, M. (2002), 'Effective local search algorithms for routing and scheduling problems with general time window constraints', *TRANSPORTATION SCIENCE* **39**, 206–232.

Imran, A., Salhi, S. & Wassan, N. A. (2009), 'A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem', *European Journal of Operational Research* **197**(2), 509 – 518.

Ioannou, G., Kritikos, M. & Prastacos, G. (2001), 'A greedy look-ahead heuristic for the vehicle routing problem with time windows', *J Oper Res Soc* **52**(5), 523–537.

Jain, A. K. & Dubes, R. C. (1988), *Algorithms for clustering data*, Prentice-Hall, Inc.

Jain, A. K., Murty, M. N. & Flynn, P. J. (1999), 'Data clustering: A review', *ACM Comput. Surv.* **31**(3), 264–323.

Jansen, K. & Solis-Oba, R. (2006), An asymptotic approximation algorithm for 3d-strip packing, *in* 'Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm', SODA '06, ACM, New York, NY, USA, pp. 143–152.

John Mangan, Chandra Lalwani, T. B. (2008), *Global Logistics and Supply Chain Management*, John Wiley & Sons.

Karabulut, K. & İnceoğlu, M. (2005), 'A hybrid genetic algorithm for packing in 3d with deepest bottom left with fill method', *Advances in Information Systems* pp. 441–450.

Karp, R. (1972), Reducibility among combinatorial problems, *in* R. Miller, J. Thatcher & J. Bohlinger, eds, 'Complexity of Computer Computations', The IBM Research Symposia Series, Springer US, pp. 85–103.

Karypis, G., Han, E.-H. & Kumar, V. (1999), 'Chameleon: hierarchical clustering using dynamic modeling', *Computer* **32**(8), 68 –75.

Kaufman, L. R. & Rousseeuw, P. (1990), 'Finding groups in data: An introduction to cluster analysis'.

Kaufman, L. & Rousseeuw, P. (1987), *Clustering by Means of Medoids*, Reports of the Faculty of Mathematics and Informatics, Faculty of Mathematics and Informatics.

Kirkpatrick, S., Vecchi, M. et al. (1983), 'Optimization by simmulated annealing', *science* **220**(4598), 671–680.

Ko, M., Tiwari, A. & Mehnen, J. (2010), 'A review of soft computing applications in supply chain management', *Applied Soft Computing* **10**(3), 661 – 674.
URL: *http://www.sciencedirect.com/science/article/pii/S1568494609001641*

Kotsiantis, S. B. & Pintelas, P. E. (2004), 'Recent advances in clustering: A brief survey', *WSEAS Transactions on Information Science and Applications* **1**, 73–81.

Ladner, R. E. (1975), 'On the structure of polynomial time reducibility', *J. ACM* **22**(1), 155–171.

Lambert, D. M., GarcÃa-Dastugue, S. J. & Croxton, K. L. (2005), 'An evaluation of process-oriented supply chain management frameworks', *Journal of Business Logistics* **26**(1), 25–51.

Landa-Silva, D., Wang, Y., Donovan, P. & Kendall, G. (2011), Hybrid heuristic for multi-carrier transportation plans, *in* 'Proceedings of the 9th Metaheuristics International Conference (MIC 2011)', pp. 221–229.

Laporte, G. (1992), 'The vehicle routing problem: An overview of exact and approximate algorithms', *European Journal of Operational Research* **59**(3), 345 – 358.

Laporte, G., Gendreau, M., Potvin, J.-Y. & Semet, F. (2000), 'Classical and modern heuristics for the vehicle routing problem', *International Transactions in Operational Research* **7**(4-5), 285–300.

Li, K. & Cheng, K. (1990), 'On three-dimensional packing', *SIAM Journal on Computing* **19**(5), 847–867.

Li, K. & Cheng, K. H. (1992), 'Heuristic algorithms for on-line packing in three dimensions', *Journal of Algorithms* **13**(4), 589 – 605.

Lin, S. (1965), 'Computer solutions of the traveling-salesman problem', *BSTJ* **44**, 2245–2269.

Lin, S. & Kernighan, B. W. (1973), 'An effective heuristic algorithm for the traveling-salesman problem', *Operations Research* **21**(2), 498–516.

Lloyd, S. (1982), 'Least squares quantization in pcm', *Information Theory, IEEE Transactions on* **28**(2), 129 – 137.

McLachlan, G. J. & Krishnan, T. (2007), *The EM algorithm and extensions*, Vol. 382, Wiley-Interscience.

Miyazawa, F. K. & Wakabayashi, Y. (2007), 'Two- and three-dimensional parametric packing', *Computers and Operations Research* .

Miyazawa, F. & Wakabayashi, Y. (2009), 'Three-dimensional packings with rotations.', *Computers & Operations Research* **36**(10), 2801–2815.

Or, I. (1976), *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*, Xerox University Microfilms.

Osman, I. H. (1993), 'Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem', *Annals of Operations Research* **41**(4), 421–451.

Papadimitriou, C. H. & Steiglitz, K. (1982), 'Combinatorial optimization: algorithms and complexity. 1982', *Prenticice Hall* .

Patricia J. Daugherty, A. E. E. & Gustin., C. M. (1996), 'Integrated logistics: achieving logistics performance improvements.', *Supply Chain Management: An International Journal* **1**, 25–33.

Pham, N. (2011), A Univariate Marginal Distribution Algorithm-based Hyper-heuristic for Three-Dimensional Strip Packing Problems,, PhD thesis, The University Of Nottingham.

Pieterse, V. & Black, P. E. (2005), 'Dictionary of algorithms and data structures [online]'. **URL:** *http://www.nist.gov/dads/HTML/greedyalgo.html*

Pisinger, D. (2002), 'Heuristics for the container loading problem', *European Journal of Operational Research* **141**(2), 382–392.

Potvin, J.-Y. & Rousseau, J.-M. (1995), 'An exchange heuristic for routeing problems with time windows', *Journal of the Operational Research Society* pp. 1433–1446.

Prosser, P. & Shaw, P. (1996), 'Study of greedy search with multiple improvement heuristics for vehicle routing problems'.

Ren, Y., Dessouky, M. & Ordonezf, F. (2010), 'The multi-shift vehicle routing problem with overtime', *Computers & Operations Research* **37**(11), 1987 – 1998.

Ross, P., Marín-Blázquez, J., Schulenburg, S. & Hart, E. (2003), Learning a procedure that can solve hard bin-packing problems: A new ga-based approach to hyper-heuristics, *in* 'Genetic and Evolutionary Computation GECCO 2003', Springer, pp. 215–215.

Ross, P., Schulenburg, S., Marín-Blázquez, J. G. & Hart, E. (2002), Hyper-heuristics: learning to combine simple heuristics in bin-packing problems, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference', Vol. 2002, Morgan Kaufmann Publishers Inc., pp. 942–948.

Sander, J., Ester, M., Kriegel, H.-P. & Xu, X. (1998), 'Density-based clustering in spatial databases: The algorithm gdbscan and its applications', *Data Mining and Knowledge Discovery* **2**, 169–194.

Sato, M., Sato, Y. & Jain, L. (2002), *Fuzzy clustering models and applications*, number 9 *in* 'Studies in Fuzziness and Soft Computing', Physica-Verlag HD.

Sibson, R. (1973), 'Slink: An optimally efficient algorithm for the single-link cluster method', *The Computer Journal* **16**(1), 30–34.

Solomon, M. M. (1987), 'Algorithms for the vehicle routing and scheduling problems with time window constraints', *Operations Research* **35**, 254–265.

Strehl, A. & Ghosh, J. (2003), 'Cluster ensembles — a knowledge reuse framework for combining multiple partitions', *J. Mach. Learn. Res.* **3**, 583–617.

Taillard, Ã., Badeau, P., Gendreau, M., Guertin, F. & Potvin, J.-Y. (1997), 'A tabu search heuristic for the vehicle routing problem with soft time windows', *Transportation Science* **31**(2), 170–186.

Tan, K., Lee, L., Zhu, Q. & Ou, K. (2001), 'Heuristic methods for vehicle routing problem with time windows', *Artificial Intelligence in Engineering* **15**(3), 281 – 295.

Tang, J., Pan, Z., Fung, R. Y. & Lau, H. (2009), 'Vehicle routing problem with fuzzy time windows', *Fuzzy Sets and Systems* **160**(5), 683 – 695.

Tas, D., Dellaert, N., van Woensel, T. & de Kok, T. (2013), 'Vehicle routing problem with stochastic travel times including soft time windows and service costs', *Computers & Operations Research* **40**(1), 214 – 224.

Tay, J. C. & Ho, N. B. (2008), 'Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems', *Computers & Industrial Engineering* **54**(3), 453–473.

Team, E. (2012), Uk transport and logistics,sector outlook third quarter 2012, Technical report, Client Capital Management, Corporate Banking, Barclays Bank PLC.

Teodorovic, D. (1999), 'Fuzzy logic systems for transportation engineering: the state of the art', *Transportation Research Part A: Policy and Practice* **33**(5), 337 – 364.

Terashima-Marín, H., Farías Zárate, C., Ross, P. & Valenzuela-Rendón, M. (2006), A ga-based method to produce generalized hyper-heuristics for the 2d-regular cutting stock problem, *in* 'Proceedings of the 8th annual conference on Genetic and evolutionary computation', ACM, pp. 591–598.

Terashima-Marin, H., Farias Zarate, C., Ross, P. & Valenzuela-Rendon, M. (2007), Comparing two models to generate hyper-heuristics for the 2d-regular bin-packing problem, *in* 'Proceedings of the 9th annual conference on Genetic and evolutionary computation', ACM, pp. 2182–2189.

Terashima-Marín, H., Ortiz-Bayliss, J., Ross, P. & Valenzuela-Rendón, M. (2008), Hyper-heuristics for the dynamic variable ordering in constraint satisfaction problems, *in* 'Proceedings of the 10th annual conference on Genetic and evolutionary computation', ACM, pp. 571–578.

Thompson, P. M. & Orlin, J. B. (1989), The theory of cyclic transfers. Working paper, Operations Research Center, MIT, Cambridge, MA.

Thompson, P. M. & Psaraftis, H. N. (1993), 'Cyclic transfer algorithm for multivehicle routing and scheduling problems', *Operations Research* **41**(5), 935–946.

Tsao, Y.-C. & Lu, J.-C. (2012), 'A supply chain network design considering transportation cost discounts', *Transportation Research Part E: Logistics and Transportation Review* **48**(2), 401 – 414.

Vázquez-Rodríguez, J. A. & Petrovic, S. (2010), 'A new dispatching rule based genetic algorithm for the multi-objective job shop problem', *Journal of Heuristics* **16**(6), 771–793.

Voorhees, E. M. (1986), 'Implementing agglomerative hierarchic clustering algorithms for use in document retrieval', *Information Processing & Management* **22**(6), 465–476.

Wang, W., Yang, J. & Muntz, R. R. (1997), Sting: A statistical information grid approach to spatial data mining, *in* 'Proceedings of the 23rd International Conference on Very Large Data Bases', VLDB '97, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 186–195.

Wäscher, G., Haußner, H. & Schumann, H. (2007), 'An improved typology of cutting and packing problems', *European Journal of Operational Research* **183**(3), 1109–1130.

Wei, L., Oon, W.-C., Zhu, W. & Lim, A. (2012), 'A reference length approach for the 3d strip packing problem', *European Journal of Operational Research* **220**(1), 37 – 47.

Xu, R. & Wunsch, D., I. (2005), 'Survey of clustering algorithms', *IEEE Transactions on Neural Networks* **16**(3), 645–678.

Xu, X., Ester, M., Kriegel, H.-P. & Sander, J. (1998), A distribution-based clustering algorithm for mining in large spatial databases, *in* 'Data Engineering, 1998. Proceedings., 14th International Conference on', pp. 324–331.