Multi-Objective Optimisation of the Generalized Bin Packing Problem

MSc in Advanced Analytics - University of Cape Town

2024

Andrea Plumbley

PLMAND002

Supervisor: Dr Rosephine Georgina Rakotonirainy

Minor dissertation presented in partial fulfilment of the requirements for the degree of Master of Science in Advanced Analytics

Plagiarism Declaration

I know the meaning of plagiarism and declare that all of the work in the dissertation, save for that which is properly acknowledged, is my own.

A. Plumbley, 2025

Abstract

This project aimed to investigate multi-objective optimisation of the generalized bin packing problem, which involves the allocation of compulsory and non-compulsory items into a set of bins. The items have characteristics such as weight, width, height, and due date, while the bins have characteristics such as capacity and cost. The main objective of this problem is to minimize cost which usually corresponds to minimizing the number of bins used. However, in many real-world applications there may be multiple objectives that are trying to be met, and these may be competing such as item due dates and load balancing objectives. Classical methods for solving such problems involve combining the objectives into a single objective or converting some of the objectives into constraints with associated goals. Both approaches require one to have prior knowledge of the decision-makers' preferences in terms of a trade-off between the different objectives which are often difficult to obtain. In this work, a multi-objective evolutionary model is proposed to tackle the generalized bin packing problem. The proposed approach optimises the problem across multiple objectives, allowing decision-makers to make a trade-off between solutions presented as a Pareto front. Two objective combinations were considered: cost and item lateness, and cost and load imbalance. The developed model was tested on one- and two-dimensional problem instances, demonstrating its ability to minimize objectives and provide a set of conflicting solutions in certain cases. The results also highlighted potential limitations of the algorithm, such as premature convergence and a lack of solution diversity. Potential reasons for these limitations and recommendations for future research to improve the current algorithm are discussed. This work contributes to the limited literature on multi-objective optimisation of the generalized bin packing problem, providing a multi-objective evolutionary algorithm for the problem, while also highlighting some of the problems encountered when performing multi-objective optimisation.

Acknowledgements

I would like to acknowledge the support and assistance of my supervisor, Dr Rosephine Georgina Rakotonirainy. Thank you for your guidance and consistent encouragement throughout this project. You have been a great source of support and it has been a pleasure working with and learning from you.

I would also like to acknowledge the support of my family and friends. Thank you for showing interest in my work, for asking me questions and for all your encouragement throughout the year.

Lastly, I would like to thank the many other individuals with whom I had conversations about this work; your good questions and different perspectives provided valuable contributions that helped shape this project.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Logistics and distribution activities make up a large part of many businesses, with the logistics industry being worth an estimated 8.4 trillion euros in 2021 (Placek 2023). A very common problem seen in logistics, distribution, transportation and planning is the packing problem. Packing problems are optimisation problems that require a number of items to be packed into containers while minimizing some objective subject to a number of constraints. A two-dimensional representation of a packing problem is shown in Figure 1 where items in (a) need to be allocated to bins in (b).

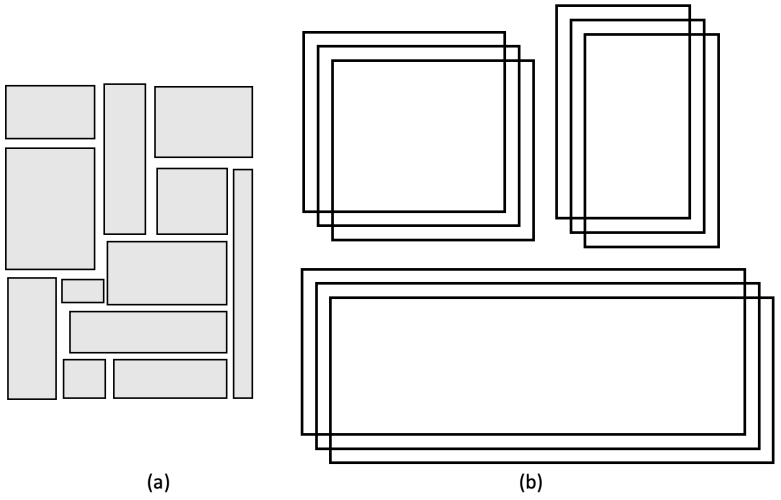


Figure 1: An example of a two-dimensional version of a bin packing problem. (a) represents the set of small items and (b) represents the bins into which items must be packed.

Packing allocation decisions are often on a large scale and optimal solutions are not trivial or easy to find. In many cases the decision on how to best allocate items into different bins, needs to be made in a relatively short amount of time. For example, courier companies receive items from suppliers and the turnaround time for delivery is often short. It is therefore necessary to have models and algorithms developed that are able to handle large problems and give optimal or near optimal solutions in reasonable time frames.

1.1 Problem statement

The classic bin packing problem aims to load a set of items into a set of bins while minimizing total cost (Lodi, *et al.* 1999). In this work a particular type of bin packing problem known as the generalized bin packing problem (GBPP) is investigated. As the name suggests, it is a generalization of the classic bin packing problem and includes a set of compulsory and non-compulsory items along with a set of bins (Baldi, *et al.* 2012). The aim of the GBPP is to allocate all the compulsory items along with some non-compulsory items into the bins in such a way as to minimize costs. The non-compulsory items give additional profit if they are able to be packed and no cost is associated with them if they are not allocated to a bin. The cost that is being minimized in the GBPP is therefore the difference between the cost of the bins used and the profit of the non-compulsory items that are allocated.

A GBPP can be one-, two-, or three-dimensional in nature and this dimensionality refers to the item and bin capacities. In a one-dimensional (1D) bin packing problem, the items have a single dimension: volume or weight. Similarly, the bins will have a single capacity dimension such as volume or weight capacity. In this case, the sum of the weight or volume of the items packed into a specific bin must not exceed the bin's weight or volume capacity. In a two-dimensional (2D) GBPP, items and bins have two parameters or attributes: length and width, length and height or width and height. In such a case the sum of the lengths of the items in a bin must not exceed the bin length and the sum of the item widths must not exceed the bin width, or whichever attributes are being used. There are however specific requirements to calculate this as items can be stacked on top of one another and rotated. A three-dimensional problem follows on from the 2D problem, where length, width and height are all simultaneously taken into account. The sum of the item heights, widths and lengths must not exceed the bin height, width, and length, respectively. The focus of this work is on the 1D and 2D variants of the GBPP.

1.2 Motivation of the study

The basic version of the GBPP is to load compulsory and non-compulsory items into a set of bins constrained by volume or weight attributes (Baldi, *et al.* 2012). Since its first appearance, different extensions of the problem have been proposed. Particularly, Baldi, *et al.* (2019) consider a model for the single objective GBPP which includes bin-dependent item profits, where the profit gained depends on which bin (in their example transport company) is used. A number of heuristics were explored to solve this problem. In 2020, Gradisar & Glavan investigated a model with variability in bins and items where bins and items were not identical and different bin sizes existed. A year later, Crainic, *et al.* (2021) explored an extension of the basic version of the GBPP by considering time constraints, adding due dates to the items and introducing multi-period problems. Their model includes item attributes such as the earliest and latest delivery date as well as travel times. Adding such extensions make the problems more complex but makes them more realistic and applicable to real world scenarios.

Another way in which the complexity can be increased is through the dimensionality of the problem. As previously discussed, the GBPP can be modeled in 1D or higher dimension depending on the practical problem under consideration. As the dimensionality increases there is an increased number of constraints which need to be taken into account, and this introduces complexity particularly in terms of how to represent the problem and the heuristics used to allocate items into bins.

Though these extensions have been investigated, the multi-objective GBPP (MOGBPP) has seldom been studied in the literature despite its practical application. In particular, the MOGBPP describes the fundamental role played by the trade-off between shipping costs and item profits, which arises in all transportation settings. To the best knowledge of the author, only Liu, *et al.* (2008) have addressed the 2D MOGBPP with multiple bin sizes in the literature. Kaabi, *et al.* (2018) and Erbayrak, *et al.* (2021) consider the 3D multiple bin size MOGBPP. Other work and extensions focus specifically on single bin size packing problems or cutting problems. This work, therefore, attempts to fill in that gap by investigating the 1D and 2D MOGBPP with multiple bin sizes and three conflicting objectives: cost, load imbalance, and time constraints.

In terms of solution techniques, a recent survey by Ali, *et al.* (2022) indicated that most of the algorithms put forward in the literature consider single-objective problems with only a few studies considering multi-objective cutting and packing problems. A number of classical methods exist to solve multi-objective problems with two common approaches being to either combine

the multiple objectives into a single objective or to select one objective to keep as the single objective and convert the other objectives into constraints. The appeal of these methods for solving multi-objective problems is that existing methods for solving single objective problems can then easily be applied. However, the problem with these classical methods is that they require prior information from the decision-maker about the objectives in order to weight the objectives when combining them or to select bounds on the new constraints. This is not ideal as it requires the decision-maker to have some tolerance for the different objectives beforehand and this is not always known. An ideal solution for multi-objective problems would be to present a decision-maker with a set of solutions. When a set of solutions to such a problem is found, the solution set can be presented as a Pareto front which a decision-maker can use to make a trade-off between objectives and gain a visual representation of the trade-offs that could be made. Being able to obtain solutions in this way does not require one to have prior knowledge of the decision-makers' preferences.

The focus of this project will be to consider multi-objective methods to solve the MOGBPP that give such a solution set. Multi-objective evolutionary algorithms (MOEAs) are heuristic methods that have shown much promise for multi-objective optimisation problems (de Armas, *et al.* 2010). These algorithms simulate the process of natural evolution and make use of a population which iteratively creates generations of candidate solutions that move toward the optimal solution (Zitzler 1999). These algorithms are particularly useful because they are able to handle large scale and difficult optimisation problems, which is often the case in the real world. These algorithms are also ideal as they are able to keep the multiple objectives separate and thus present a set of solutions to the problem which can be used to make trade-offs. This research will unpack the algorithms that have already been used in similar projects as well as develop and test algorithms specifically for the MOGBPP. By doing this, this research will also contribute toward the gap in the literature in terms of MOGBPP solution techniques.

The GBPP is an important problem because of its large application in the area of logistics, but in addition because it generalizes a number of classical bin packing and knapsack problems (Baldi & Bruglieri 2017). This generalization is of an advantage because it means that the same solution techniques can be used for a large number of problems. These problems may vary in some aspects but if a problem can be modeled as a GBPP then GBPP solution techniques can be used to optimise that specific problem. Optimisation of such problems helps companies to make better decisions and plan for their distribution activities more efficiently.

1.3 Aims and objectives

The aim of this project is to develop a mathematical model to represent a GBPP with multiple objectives and to conduct a computational study to evaluate the efficiency of the proposed model with respect to suitable test instances. One and two-dimensional MOGBPP are considered, with three objective functions including cost, time, and load imbalance. The research focuses on outlining methods that can be used to optimise the MOGBPP, considering the various constraints of the problem, and discussing how well these methods performed. In particular, the focus is on the application of evolutionary algorithms, specifically the genetic algorithm. In the context of multi-objective optimisation, the genetic algorithm is able to solve multi-objective problems and produce a set of Pareto optimal solutions in one run due to its ability to simultaneously search different regions of the solution space (Rajappa 2012). The efficiency of the proposed model and algorithm will be evaluated with respect to the solution quality and computational time. This is important as many logistics and distribution decisions need to be made on-the-go, within a

short time frame.

The above-mentioned aims are accomplished by pursuing the following six objectives:

1. To conduct a review of the relevant literature.
2. To develop a mathematical model for the MOGBPP.
3. To identify algorithms to solve the MOGBPP model developed in fulfillment of objective 2.
4. To implement the algorithm identified in objective 3.
5. To assess the performance of the developed algorithm in fulfillment of objective 3 and 4 in terms of computational requirements as well as quality of solutions.
6. To recommend future research directions. This includes methods to use to solve real-world MOGBPP problems.

By developing a multi-objective model along with suitable algorithms and achieving the objectives that have been set out, this research aims to contribute towards the area of MOGBPPs. This contribution aims to fill a gap in the current literature and provide recommendations for real-world multi-objective optimisation problems.

1.4 Thesis structure

The remainder of this thesis is structured as follows. Chapter 2 contains a brief review of the relevant literature to the MOGBPP. Chapter 3 outlines the methodology adopted in this work, highlighting the mathematical model for the problem and outlining the different algorithms that will be applied. Chapter 4 outlines the test instances and implementation of the algorithm. The computational results are presented in Chapter 5 which are then discussed in Chapter 6. Chapter 7 draws conclusions on the research and highlights the contributions of the work as well as recommendations for future research.

2 Literature Review

This section aims to outline research on the GBPP, specifically the MOGBPP, as well as to discuss existing solution approaches that could be used to solve the problem. First an overview of the GBPP along with its variations and extensions will be given. Following this, existing methods used to tackle multi-objective problems are discussed, specifically comparing classical methods with evolutionary algorithms. Finally existing work on the MOGBPP will be discussed, unpacking the work that has already been done in this area and highlighting gaps in the literature.

2.1 Overview of the GBPP and its variants

The GBPP as discussed in the introductory chapter consists of loading a set of compulsory and non-compulsory items into a set of bins, subject to constraints, in order to optimise some objective. This packing problem falls into the broader context of cutting and packing (C&P) problems within operations research. C&P problems are prominent in many real-world applications, particularly logistics, manufacturing and design. Cutting problems refer to small items that need to be cut from a larger sheet while packing problems refer to fitting small items into a larger bin. Thus, while the applications may look different the actual problem is very similar; namely, fitting smaller objects into larger ones. There are various types of C&P problems and Wäscher, *et al.* (2007) came up with a well-known typology of the problem, identifying six different kinds of C&P problems, as shown in Table 1.

Table 1: Types of C&P problems according to the typology of Wäscher, *et al.* (2007).

	Assortment of Small Items	Output Maximization			Input Minimization		
		Identical	Weakly heterogeneous	Strongly heterogeneous	Arbitrary	Weakly heterogeneous	Strongly heterogeneous
		Identical item placement problem	**Placement problem**	**Knapsack problem**	**Open Dimension Problem**	**Cutting Stock Problem**	**Bin Packing Problem**
Assortment of Large Objects	Identical	-	MILOPP	MIKP	-	SSSCSP	SBSBPP
	Weakly heterogeneous	-	-	-	-	MSSCSP	MBSBPP
	Strongly heterogeneous	-	MHLOPP	MHKP	-	RCSP	RBPP
	One large object	IIPP	SLOPP	SKP	ODP	-	-

The GBPP is an input minimization problem with strongly heterogeneous items where the bins can be either identical, weakly or strongly heterogeneous. Depending on the heterogeneity of the bins, Wäscher, *et al.* (2007) identify three types of bin packing problems as seen in Table 1. The Single Bin Size Bin Packing Problem (SBSBPP) has identical bins, and a real-world example would be loading boxes into trucks, where the trucks are all of the same size or capacity. In the case of weakly heterogeneous bins, a few bins are identical and the problem is called the Multiple Bin Size Bin Packing Problem (MBSBPP). An example of this kind of problem would be a courier company deciding which boxes to pack different items into with a set of five standard box sizes. Finally in the case of strong heterogeneity where few of the bins are identical, the problem is called the Residual Bin Packing Problem (RBPP). The problem being considered in this research is the MBSBPP where the bins are weakly heterogeneous.

The majority of research on the GBPP focuses on the single objective case and its extension to accommodate real-world scenarios. Crainic, *et al.* (2021) explore a multi-period bin packing model and also consider item-to-bin assignment costs. They included a temporal aspect to

their model to account for the fact that the availability of items and bins may vary over time. Additionally, by incorporating a time component, they modeled how the cost of assigning an item to a bin can fluctuate, as in cases where a shipping company may increase container usage fees during periods of high demand. To model the GBPP including multiple periods, Crainic, *et al.* (2021) make use of attributes of items such as availability time, earliest and latest delivery time and item-to-spot-market-bin assignment costs which change over time. They consider a number of heuristics to optimise the problem, and they consider the single objective of minimizing cost. In addition to considering a multi-period model, Gradisar & Glavan (2020) incorporate due date attributes to items in their framework. This means items need to be packed in order to meet their due date for delivery and their model aimed to minimize the lateness of items.

Gradisar & Glavan (2020) also considered the grouping of items or bins. They considered the GBPP with variable sized bins and formulated a model where items can be assigned to only one single group of bins to accommodate the grouping constraints. This grouping constraint means that items and bins are grouped into families and the constraint requires that all items from one family of items must be packed into bins that belong to the same bin family. Items and bins within the same families do not need to be identical to one another. The specific application area considered is steel-processing products where a group of orders (family of items) for one product need to be produced from the same batch or raw material (family of bins), where a batch of raw material consists of several material stocks (bins) which have a few common characteristics (Gradisar & Glavan 2020).

Baldi, *et al.* (2019) extend the standard GBPP by including bin-dependent item profits. In their problem formulation, the aim is to minimize a single objective which is the cost of using the bins that are selected minus the profit obtained from the items that are packed. The item profits, however, are dependent on which bin they are allocated to. An application of such a problem is a courier's choice of transport company, in which one must minimize the overall cost taking into account both the transportation costs and the service quality. They consider two heuristic and two metaheuristic approaches to solve the problem.

Having outlined the GBPP in the broader context of C&P problems and introduced some of its extensions, the multi-objective setting will now be discussed.

2.2 Multi-objective C&P problems

While there are very few studies on the MOGBPP, a number of studies exist on multi-objective modeling for other types of C&P problems. These works will be documented in this section, highlighting the types of problems and objectives considered.

2.2.1 Multi-objective cutting problems

Tiwari & Chakraborti (2006) consider a multi-objective 2D cutting problem. The problem that is considered involves the cutting of different shaped rectangles into a sheet of paper. The objectives are to minimize the length of the sheet of paper that is used and minimize the number of cuts made. The problem is therefore finding the best way to layout the rectangles on a sheet with fixed width and variable length so as to minimize the number of cuts that a machine needs to make and to minimize the length of paper used in order to use as few resources as possible. Both guillotine and non-guillotine cutting techniques are considered as two variations to the problem. De Armas, *et al.* (2010) extended the work of Tiwari & Chakraborti (2006) by also considering a multi-objective 2D cutting problem. The difference in their research is that they make use of a

slightly different MOEA algorithm particularly with differences in the selection and replacement steps of the algorithm.

Zheng, *et al.* (2012) also consider a 2D cutting problem and the objectives they consider are minimizing wasted sheet material and minimizing the total processing time. They propose a hybrid heuristic method for solving the 2D steel coil cutting problem. The proposed hybrid heuristic has three stages: initialization, optimise and match, and execute cutting process. These steps consist of clustering, ordering, striping and integer programming methods and in this way combine a set of techniques into a hybrid method for solving the problem.

Gomez & Terashima-Marin (2012) also consider the use of hyper-heuristic methods for multi-objective cutting problems, specifically making use of the NSGA-II MOEA algorithm. The objectives that are considered are minimizing the number of sheets used and the time required to perform the placement of the pieces onto the sheet. The problem also considered solving for irregular items that need to be cut from the sheet. To test their hyper-heuristic method, Gomez & Terashima-Marin (2012) generated 18 different types of problems, with 30 instances of each problem on which to run their algorithms. The proposed method produced efficient results after using NSGA-II in a training phase, and the results of the method applied to the test set solved the problem efficiently taking into account the trade-off between both objectives.

2.2.2 Multi-objective packing problems

Most of the literature on multi-objective C&P problems is specifically considering the bin packing problem, with the two main objectives outside of cost that are studied being the minimization of load imbalance and time-based objectives. The majority of work models the cost objective as minimizing the number of bins used, as in general the more bins that are used the more costly the solution to the packing problem is.

Liu, *et al.* (2008), Fernandez, *et al.* (2013), Trivella & Pisinger (2016) and Kaabi, *et al.* (2018) all consider the second objective of minimizing load imbalance. This objective aims to get the overall centre of gravity of the bins as close as possible to the desired centre of gravity. Trivella & Pisinger (2016) and Kaabi, *et al.* (2018) consider the 3D packing problem trying to minimize load imbalance while Liu, *et al.* (2008) and Fernandez, *et al.* (2013) consider the 2D variant of the problem.

An additional objective that has been considered is to incorporate time into the second objective, along with minimizing the number of bins used. Naderi & Yazdani (2014) and Gomez & Terashima-Marin (2018) incorporate time in their model by having the second objective of minimizing the time required to complete the task. These studies attempt to reduce the time it takes to make the allocation of items to bins, which is important due to the short time frame in which packing decisions often need to be made as discussed previously. Bennell, *et al.* (2013), Arbib & Marinelli (2017) and Arbib, *et al.* (2021) incorporate time into the second objective by minimizing the maximum lateness of items according to a pre-defined schedule. In such a case, each item is given a due date and the aim is to minimize the lateness of items with respect to their due dates.

A few studies have been done considering other types of objectives. Geiger (2008) and Khanafer, *et al.* (2012) both consider the objective of minimizing conflict between items in bin packing problems. Minimizing the conflict between items is accomplished by minimizing the average heterogeneity of the bins (Geiger 2008). Other objectives that have been considered as the

additional objective are: minimizing the cost of the packing process (Sathe, *et al.* 2009) and maximizing the bin space utilisation (Leung, *et al.* 2008).

Three studies have been conducted considering three objectives for the bin packing problem. El Yaagoubi, *et al.* (2022) considered a barge convoy container stowage planning problem. The three objectives consider shifting movement, convoy stability and the number of barges used. Erbayrak, *et al.* (2021) introduced the concept of a family unit to their packing model and the three objectives considered where minimizing the number of bins and the load imbalance as well as maximizing the family unit ratio. Maximizing the family unit ratio encourages the packing of a family of products together. The final paper that considered three objectives was the model proposed by Dahmani, *et al.* (2014) which focuses on a 2D packing problem and aims to minimize the number of bins used, the maximum length of a bin and the load imbalance.

2.3 Multi-objective methods

The multi-objective setting allows for a problem to be optimised in such a way that all objectives are taken into account at once, and the overall problem is tackled as a whole. This provides a better approach than solving a problem according to each objective separately and then trying to decide on which solution is best to use, as this often results in very imbalanced solutions in terms of the different objective values. The aim of multi-objective models is to find a good balance of all the objectives present in a particular problem. In many cases the multiple objectives may be conflicting or competing but must be optimised simultaneously (de Armas, *et al.* 2010). It is often impossible to find a single optimal solution that is better than all others, or dominates all others. This means that a trade-off needs to be made between the objectives.

There are two main approaches that are used in the literature to solve multi-objective cutting and packing problems. The first approach is to transform the problem into a single objective problem and then apply classical solution methods to the single objective optimisation problem. The second approach is the use MOEA algorithms which try to find a set of well-distributed non-dominated solutions. Instead of a single solution, a set of solutions is found using this method and this allows a decision maker to make a trade-off between the different objectives (de Armas, *et al.* 2010). This solution set contains only non-dominated solutions[1]. The two approaches to solving multi-objective problems are now discussed in further detail.

2.3.1 Classical solution approaches

One common approach to problems with multiple objectives is to combine the objectives into a single objective (de Armas, *et al.* 2010). This approach, however, requires the decision maker to have prior knowledge of the problem and decide on the trade-off between the different objectives beforehand. Most research overcomes this issue by designing a mathematical model which contains the multiple objectives in order that the solutions can be presented as a set and then a trade-off can be made. Solutions to bi-objective optimisation problems can be presented as Pareto fronts as done by de Armas, *et al.* (2010) and Tiwari & Chakraborti (2006) in their multi-objective cutting problem as well as by Liu, *et al.* (2008) considering a particular method to a bi-objective bin packing problem. These Pareto fronts are useful as they can give a decision-maker a visual representation of the solution set and the trade-off between the objectives. In addition, one can compare Pareto fronts resulting from different methods or algorithms to one

[1] A solution x is non-dominated if there does not exist a solution y that performs better in at least one objective and equivalently well as x in the remaining objectives.

another to identify if one method results in better solutions compared to another. By keeping the multiple objectives separate and not combining them, it results in greater solution diversity than when objectives are combined (de Armas *et al.* 2010).

Another approach to solving multi-objective problems is goal programming, where one or more objectives are treated as goals and the deviations from these goals are minimized, by minimizing a sum of the deviations. This approach allows multiple objectives to be addressed at once and one does not have to convert them into hard constraints. However, a challenge with this approach is that appropriate target levels for goals need to be specified and if they are not chosen appropriately it can result in no solution being found (Zitzler 1999). This means the decision-maker must know and provide suitable ranges for each of the objectives and this can be challenging to obtain for complex problems or problems that change over time.

These two classical approaches are often used because methods, such as exact methods or metaheuristics, used for single objective optimisation problems can then be easily used to solve the problem without needing to make any changes to the existing methods. Exact methods include dynamic programming, branch-and-bound, and linear and integer programming methods among others (Zitzler 1999). These exact methods are, however, often impractical for many real-world scenarios because they cannot handle large problems and require a large amount of computational time in order to run. If these methods are run on the problem multiple times and the weights on the combined objectives or goals in the constraints are changed, then a set of solutions can be found which can be used to make trade-offs. However, doing multiple runs can be costly in terms of computational time and resources. It is therefore beneficial to make use of other methods of solving optimisation problems that can handle all of the objectives simultaneously and present a set of solutions that form a trade-off in a single run. Such methods would not require a large amount of prior information about the problem or the decision-makers' preferences.

Exact algorithms are based on mathematical programming approaches and are guaranteed to identify an optimal solution or show that no feasible solution exists. Heuristics, on the other hand, refer to any set of rules that may generate good feasible solutions to the optimisation problem but without any proof of convergence to optimality. These methods provide satisfactory but not necessarily optimal solutions. Metaheuristics refers to the combining of different heuristics into one algorithm and many advanced types of such algorithms have been proposed such as simulated annealing, genetic algorithms and ant colony optimisation. (Meta)Heuristic methods therefore provide one with a trade-off between computational time and solution quality, taking less time and resources than an exact method but the solution not necessarily being the precise optimal solution.

2.3.2 Multi-objective evolutionary algorithms

The main approach to solving multi-objective optimisation problems is the use of evolutionary algorithms. The use of evolutionary algorithms, namely MOEAs, is because classical methods are not ideal as discussed above. Evolutionary algorithms have been shown to be useful for finding solutions to large and difficult optimisation problems and have been successfully used in many real-world applications (de Armas, *et al.* 2010).

Evolutionary algorithms are population-based heuristic methods, and they aim to simulate the process of natural evolution (Zitzler 1999). Evolutionary algorithms include the genetic algorithm (Holland 1975), particle swarm optimisation (Kennedy & Eberhart 1995) and ant

colony optimisation (Dorigo 1992) among others. The basic principle is to move a population of solutions towards good regions of the search space, towards the optimal solution.

The basic MOEA algorithm framework is depicted in Figure 2. A population of solutions is created as the initial step. Following this, the algorithm works iteratively with four steps: selection, crossover, mutation, and replacement (Deb 2008). These steps are applied to the current population to create a new population, the next generation of solutions. In order to perform the selection step, fitness scores need to be given to each candidate solution. Fitness scores are obtained using an evaluation of how that particular solution does at meeting the objectives of the problem. The fitness score can also be used to indicate whether a solution is not feasible by giving it a very low score. The selection process chooses the best candidate solutions in a population based on the fitness scores they are assigned. Once candidate solutions have been selected, a crossover operator is used to create a new generation of solutions. Crossover refers to taking two parent genes, and selecting parts of each parent to create a 'child' solution. The mutation step further perturbs the new population of solutions based on a mutation probability. Finally, the replacement step selects the best candidate solutions from the old and new sets of solutions and makes the new population of solutions. The entire process is repeated until a termination criteria is reached (Deb 2008). The multi-objective version of these algorithms, MOEAs, have been shown to behave better than blind search strategies for multi-objective problems (Zitzler 1999).

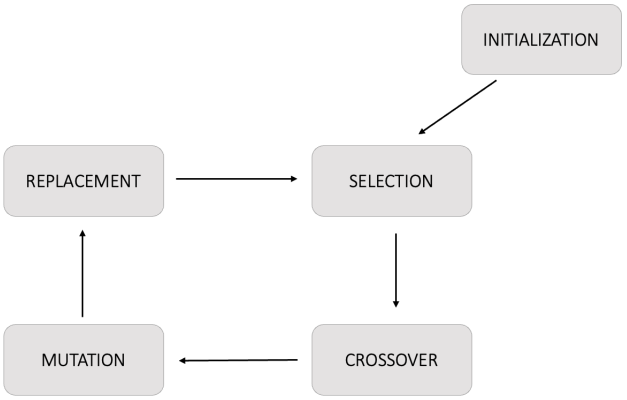


Figure 2: Basic MOEA algorithm framework consisting of four steps: selection, crossover, mutation, and replacement.

Four classes of MOEAs have been utilised in the context of C&P problems (Zhou, *et al.* (2011), Huang, *et al.* (2019)) The difference between these methods lies in the way the four operators outlined previously are handled. The first MOEA class of algorithms are known as domination-based MOEAs. The main differential characteristics for such algorithms occur within the selection step. Selection is applied based on the Pareto dominance principle, where dominant solutions are chosen over dominated solutions. This selection based on dominance is done in order to achieve good convergence of the algorithm. Examples of such domination-based algorithms are NSGA-II (Deb, *et al.* 2002) and SPEA2 (Zitzler, *et al.* 2001). The second class of algorithms are called indicator-based MOEAs. These algorithms make use of performance indicators to guide the search to the optimal solution. This algorithm approach again focuses on the selection step of the algorithm, assignment of performance indicators to each solution effects how they are

selected for crossover. An example of an algorithm in the indicator-based class is the indicator-based evolutionary algorithm (IBEA) (Zitzler, *et al.* 2004). Decomposition based MOEAs are the third class of evolutionary algorithms identified by Huang, *et al.* (2019). The algorithms split the optimisation problem into multiple smaller sub-problems. These sub-problems are then solved simultaneously by performing evolutionary operations among neighbouring sub-problems. Co-evolutionary genetic algorithms are in the decomposition-based class (Zhou, *et al.* 2011). Each sub-problem is optimised by using information from its neighbouring sub-problems, by performing genetic operators to the solutions of neighbouring sub-problems. There is usually a memory component to the algorithm, where the best solution so far is stored for each sub-problem. The final class of algorithms identified are hybrid-MOEAs which combine aspects of different algorithms into one method, using advantages of each algorithm to help with complex problems. An example is combining local and global search methods to improve the speed of convergence and accuracy, as done by Lara, *et al.* (2009). Other hybrid approaches combine different algorithms to obtain better diversity, as done by Elhossini, *et al.* (2010) who implement a hybrid EA-PSO algorithm for a multi-objective optimisation problem.

Zitzler (1999) highlights two major issues that must be addressed when using an evolutionary algorithm. The first is how to implement fitness assignment and selection in such a way as to move the search towards the Pareto-optimal set. This refers to how the objective evaluation is carried out and how individual solutions are chosen based on how well they meet the objectives. The second problem that needs to be addressed is how to ensure that the population remains diverse. Maintaining a diverse population is important to ensure premature convergence does not occur and to obtain a well distributed non-dominated set of solutions.

Having outlined the GBPP and introduced multi-objective optimisation as well as evolutionary algorithms, applications of MOEA algorithms specifically to C&P problems are now presented.

2.3.3 Application of MOEA in C&P problems

Tiwari & Chakraborti (2006) use an optimisation strategy that is a modified version of NSGA-II[2]. The algorithm sorts through a population according to their rank, where solutions are ranked based on their dominance with non-dominated solutions being assigned a better rank. When individual solutions have the same rank, a crowding measure is calculated, this measure gives an indication of the density of the solutions in that particular space. Solutions with higher crowding distances are preferred because they help maintain diversity. The highest ranked solutions are chosen to create a new population. Tiwari & Chakraborti (2006) include elitism in the algorithm to ensure good solutions remain in the population and they use a partially mapped crossover technique.

Tiwari & Chakraborti (2006) tested their evolutionary algorithm along with their method of evaluation and mutation on five test cases. They compared this method using differing numbers of generations. In general, it was observed that as one increased the number of generations from 100 to 1000 to 10000, the Pareto-front became more optimal, decreasing in terms of both objectives. No further improvement was found in the Pareto-front after about 10000 generations and so they concluded that the Pareto-front for 10000 generations was the near optimal solution.

De Armas, *et al.* (2010) tested three existing optimisation algorithms on the multi-objective 2D cutting problem. The first was NSGA-II (Deb, *et al.* 2002). The second optimisation method

[2]Deb, *et al.* (2002) introduced the NSGA-II algorithm which is a non-dominated sorting-based MOEA algorithm.

tested was SPEA2 (Zitzler, *et al.* 2001). This algorithm makes use of a population and an archive. The fitness scores used in the algorithm are assigned using the sum of the raw fitness score plus a density estimation. At each iteration of the algorithm, the non-dominated individuals are used from the original population and the archive of solutions is used to update the archive, following certain criteria for the number of solutions present. Following this the usual procedure of selection, crossover and mutation are performed on the archive to create a new population of offspring. The final algorithm tested was IBEA (Zitzler & Kunzli 2004). This algorithm allows the optimisation goal to be defined in terms of a quality indicator, which is used to calculate the fitness score. It also performs binary tournament selection and does the selection process by iteratively removing the worst individual from the population and updating the fitness values of the remaining individuals. De Armas, *et al.* (2010) tested these algorithms by repeating each test case for each MOEA algorithm 30 times and average values were considered. In order to compare the results of the three algorithms, hypervolume and binary multiplicative $\epsilon-$indicator metrics were used. The hypervolume refers to the size of the objective space that is dominated[3].

From the tests conducted, de Armas, *et al.* (2010) found that the NSGA-II and SPEA2 algorithms performed similarly, and the results showed that they were a slight improvement on the IBEA algorithm, considering the hypervolume and binary multiplicative $\epsilon-$indicator. In addition, it was found that it requires less computational time and effort than the other two algorithms. De Armas, *et al.* (2010) found that their implementation of NSGA-II was an improvement on that of Tiwari & Chakraborti (2006) because they got better results with a fewer number of generations used.

The algorithm put forward by Liu, *et al.* (2008) is a Particle Swarm Optimisation (PSO)-based algorithm which makes use of some evolutionary algorithm concepts such as mutation. PSO is based on swarming theory where all particles in a neighborhood depend on the discoveries and past experiences of their neighbours (Liu, *et al.* 2008). The MOEPSO algorithm is marked by the fact that the changes in genes (which are referred to as particles) are directed by personal best or global best solutions in each instance. The particles (or genes) have a memory of their best position so far and this is called its personal best (pbest). The global best (gbest) is the particle out of the whole population with the best fitness score. The steps of the algorithm are as follows: build an initial population, evaluate this population based on the objective function, archive the particles (used for finding personal best and global best later), perform the PSO operator, perform the mutation operation, implement the bottom left fill heuristic. If a stopping criteria has been met then end the algorithm or else go back to the evaluation step (Liu, *et al.* 2008).

Liu, *et al.* (2008) use an order-based variable length particle structure to represent the solution for bin packing problems. Each particle includes the number of bins used in the solution and the order in which the items are packed into the bin. They choose the variable length representation because it is versatile and allows their selected algorithm to manipulate the permutation of items in each bin without affecting the other bins. The particle is made up of several strings of distinct integers, each string represents a bin, and an integer represents an item. They make use of the bottom left fill (BLF) heuristic to generate the variable length solutions. The BLF heuristic adheres strictly to the bottom left (BL) condition, which implies that an item cannot be moved further down or to the left (Jakobs 1996). This heuristic method works by keeping a list of

[3]This metric evaluates the algorithm by considering the diversity and spread of the points and how close they are to the Pareto front (Guerreiro, *et al.* 2021). It is preferable to obtain higher hypervolume scores as this indicates that the solution found covers a larger proportion of the Pareto front. The binary multiplicative $\epsilon-$indicator is another metric that can be used to compare algorithms and smaller values of this indicator are desirable.

insertion points which get updated as new items are packed. These insertion points, that are kept track of, are used in order to identify if any intersection exist when new items are packed into a bin and in this way they control where items can be placed in a bin.

Liu, *et al.* (2008) ran a computational experiment to examine the performance of their MOEPSO algorithm. They considered six different classes of problems which were associated with different widths and heights of the objects and bins. As the evolution progressed, *i.e.* as the iterations of the algorithm increased, the number of non-dominated solutions increased. Liu, *et al.* (2008) compared their algorithm with and without the use of the PSO operator. The use of the PSO operator was found to bring a slight improvement to the algorithm, in terms of the spread of the Pareto-front and the quality of the solutions. The size of the improvement gained using the PSO operator depended on the class that was being tested, however in general the use of the PSO operator which allows for information sharing between particles, helped improve the quality of the solutions obtained. Their work also showed that the mutation operators used in the MOEPSO algorithm allowed the algorithm to achieve better convergence and diversity which increases the algorithm's ability to explore the solution space.

Bennell, *et al.* (2013) propose a multiple crossover genetic algorithm (MXGA) for the 2D bin packing problem with due dates. The representation of the problem is a gene of length n (the number of items), where each value indicates which bin it is packed into. The Best Fit Bin heuristic is used in decoding from the gene representation into a packing layout. The MXGA uses probabilistic binary tournament selection as the selection step of the algorithm. This means two candidate solutions are chosen, and one is selected over another based on a selection probability. The main differential for this algorithm is its crossover step. MXGA, as the name suggests, uses a multicrossover algorithm. The crossover process is repeated t times to produce $2t$ temporary offspring. Probabilistic binary tournament selection is then used on these $2t$ offspring to select a single candidate offspring solution. This crossover procedure is applied to parents with a particular crossover probability, p_c. Bennell, *et al.* (2013) introduce a swap operator to produce two offspring that are different from their parents by selecting a swap point and swap substrings of parents. This swap operator is introduced in the algorithm to increase diversity in the search space. Mutation occurs to each candidate solution with a probability p_m, as for a usual evolutionary algorithm. Fitness is calculated based on the two objectives being explored by Bennell, *et al.* (2013), namely number of bins used and maximum lateness of items. Finally, the MXGA uses elitism in the replacement step. This means that parents and offspring must compete for a spot in the next generation. Parent and offspring candidate solutions are combined into one population of size $2n$ and are ordered according to their fitness, with the half with the highest fitness scores being selected to be in the next generation. The final step of the MXGA algorithm is a filtration to ensure no solutions are identical to one another. If there are identical solutions, they are replaced with randomly generated solutions to avoid premature convergence.

After running a computational experiment on a number of different classes of problems, Bennell, *et al.* (2013) found that their multicrossover genetic algorithm with single point crossover exhibited better performance compared to the single crossover genetic algorithm and the Unified Tabu Search (UTS) algorithm which were used as benchmarks to test against. This better performance was for the 2D BPP when due dates were not considered and only the number of bins and bin utilization was considered. It was found the the relative quality of the results was not clear cut for the 2DBPP when due dates were added to the problem where the MXGA had mixed success when compared to the UTS.

Fernandez, *et al.* (2013) proposed a multi-objective memetic algorithm (MOMA) to optimise a

2D bin packing packing trying to minimize load imbalance and the number of bins used. The memetic algorithm is a population based meta-heuristic method. A population of solutions is initialized using two strategies: random insertion of items and Bottom Left Level (BLL) which is modified version of the BLF heuristic. The BLL heuristic uses a parameter 'level' to establish the maximum height of bin where items can be packed, and this helps with the optimisation of the load imbalance objective.

In terms of the implementation, Fernandez, *et al.* (2013), used a real-valued representation of the problem where the problem was represented in matrix form with columns for the object number, the height, width and weight of object as well as the bottom left and upper right position of each object. This method differs from the usual approach of having a gene representation in a single string. The proposed algorithm has four mutation operators which mutate the candidate solutions, a crossover operator and makes use of two local search optimisers. All of these are specifically designed to improve the distribution of the items in the bins. The proposed MOMA makes use of an archive instead of using roulette-based selection. The archive of non-dominated solutions is continuously updated to store those non-dominated solutions found in the search process. The size of the archive is managed by a niching strategy to ensure that the best non-dominated solutions are kept and the size does not exceed the maximum size. An additional algorithm that they proposed was a parallel version of the MOMA where the population is divided into sub populations and then the sub populations evolve, where migration between sub populations can happen at various points.

Fernandez, *et al.* (2013) ran computational experiments using a population of 200. They found that most of the non-dominated solutions obtained using their proposed MOMA for the 2DBPP dominated the solutions found when using a MOEPSO method, a hybrid PSO-based evolutionary algorithm developed by Wang, *et al.* (2010). It was also found that the MOMA obtained better non-dominated fronts in terms of coverage and hyper-volume metrics compared to the MOEPSO algorithm. In terms of the parallel MOMA, good performance was found and this technique improved the quality of the non-dominated fronts compared to both the proposed sequential MOMA and the MOEPSO algorithm.

2.4 Packing heuristics for the 2D problem

The 2D problem needs to consider the layout of the items in the cutting or packing problem. There are a number of different packing heuristic methods that exist that aim to pack a set of items as efficiently, in terms of space, as possible. These methods will be briefly outlined. The method adopted in this research is outlined in more detail in the methodology section.

A well-known packing heuristic is the BL algorithm, introduced by Baker, *et al.* in 1980. This method aims to pack items into the lowest and leftmost possible position. The BL-condition which has to be met in this heuristic method, means that no items can be shifted further down or to the left of the bin. Another 2D packing heuristic method is the heuristic recursive algorithm introduced by Zhang, *et al.* (2006). This algorithm adopts a divide-and-conquer strategy and decomposes the original packing problems into sub-problems and then combines the solutions of the sub-problems to create a final packing solution. The algorithm works by recursively partitioning the problem and forming bounded and unbounded regions which have different packing strategies.

The best-fit (BF) algorithm, proposed by Burke, *et al.* (2004) differs from the previous two algorithms as it does not go through a sorted list of items to be packed but rather searches

the list of unpacked items for the best-suited item to be placed in the current available position. Like the other algorithms, the BF algorithm looks in the leftmost lowest position to pack the next item and decides on the most suitable item based on item width. The constructive heuristic (CH) proposed by Leung, *et al.* (2011) is an extension of the best-fit algorithm. It makes use of a skyline which is made up of segments and the algorithm attempts to fill the lowest left most segment first. Items are scored according to a scoring rule which considers the dimensions of the item, along with the segment and the height of its neighbours. The item with the highest score is packed and if no items can fit in the segment, the segment is raised to the height of its lowest neighbour. According to the literature, the CH algorithm outperforms the other methods on average (Rakotonirainy 2018) and is thus adopted in this research.

This literature review has aimed to outline the GBPP and its application areas as well as highlight the relevant existing work in the area of multi-objective C&P problems. From a review of the literature, it is clear that there are still gaps in terms of the multi-objective GBPP. This research will aim to contribute towards some of these areas. Specifically, this research will add an additional study in the area of C&P problems, assessing the use of MOEAs for solving these problems. This research will consider both the 1D and 2D variants of the problem. The test instances to be used will include bins of various sizes contributing to the gap for the variable bin size problem. In addition, this research will highlight where some of the challenges lie when it comes to optimising the MOGBPP.

3 Methodology

The methodology used in this research for solving the MOGBPP is now presented. First, the mathematical formulation of the GBPP is given. Following this the multi-objective extension is presented, together with the problem constraints and specific packing heuristic. The non-dominated MOEA algorithm adopted in this research is then outlined. In particular, a linear relaxation of the problem was investigated in order to generate initial solutions and lower bounds for the problem.

3.1 MOGBPP Notation

As stated in Section 2.1, the GBPP is concerned with allocating a finite set of items I into a finite set of bins J. Each item i is characterized by weight (*resp.* width and height for the 2D case), and profit while each bin is characterized similarly by capacity (*resp.* width and height for the 2D case) and cost. The set I is split into two subsets, compulsory items I^C and non-compulsory items I^{NC}. There are different types of bins where T denotes the set of bin types. Bins belonging to the same bin type $t \in T$ have the same dimensions and cost. For each bin, $j \in J$, its type is denoted by $\sigma(j) \in T$, this function indicates what type of bin it is. A maximum number of U_t bins can be used for each bin type $t \in T$. In the case that the number of bins of type t is unlimited then $U_t = |I|$ because each bin must contain at least one item. The MOGBPP focuses on allocating items to bins in such a way that minimizes or maximizes all specified objectives, depending on their nature.

The mathematical formulation of the MOGBPP makes use of the notation presented in Table 2. Details of the model, including the objective functions and constraints, are provided in the following sections.

3.1.1 Objective Functions

Three different objective functions are considered in this work, namely the total cost, load imbalance, and maximum lateness of packed items.

Objective 1: Minimize Cost

The main objective considered in this work is cost minimization, which relates to reducing the number of bins used in the packing process. Minimizing cost is relevant in practical applications in logistics, inventory management, and resource allocation where stakeholders are interested in keeping costs as low as possible in order to increase profits. The overall cost is the difference between the cost of the bins used and the profits obtained by the loaded non-compulsory items.

The cost objective is given by:

$$\sum_{j \in \tilde{J}} C_j - \sum_{i \in \tilde{I}^{NC}} p_i + \sum_{i \in I^{NC}} p_i,$$

where $\tilde{J}$ refers to the set of bins used in solution and $\tilde{I}^{NC}$ the set of non-compulsory items packed in solution. The last term, the sum of profits of all items is added as a constant and ensures that the objective remains non-negative. This simplifies to:

$$\sum_{j \in \tilde{J}} C_j + \sum_{i \in I^{NC} \setminus \tilde{I}^{NC}} p_i,$$

which is the sum of the bin costs along with the profit of the unloaded non-compulsory items.

Table 2: Notation used in mathematical formulation of the MOGBPP as presented by Baldi, *et al.* (2012).

Variable/Set/Parameter	**Description**
I	set of items
$n = \|I\|$	number of items
$I^C \subseteq I$	set of compulsory items
$I^{NC} \subseteq I$	set of non-compulsory items ($I^C \cup I^{NC} = I$, and $I^C \cap I^{NC} = \emptyset$)
$n_c = \|I^C\|$	number of compulsory items
J	set of bins
$m = \|J\|$	number of bins
T	set of bin types
$\sigma : J \to T$	function indicating which bin type each bin is, *e.g.* $\sigma(j) = t$ if bin $j \in J$ is of type $t \in T$.
p_i	profit generated by each item when allocated
d_i	due date of item $i \in I$
w_i	width of item $i \in I$
h_i	height of item $i \in I$
γ_i	weight of item $i \in I$
a_i	area of item $i \in I$
W_t	width of bin of type $t \in T$
H_t	height of bin of type $t \in T$
C_t	cost of a bin of type $t \in T$
Γ_t	weight capacity of a bin of type $t \in T$
A_t	area of bin of type $t \in T$
U_t	maximum number of bins to be used of type $t \in T$
$U \leq \sum_{t \in T} U_t$	maximum number of bins to be used
$\tilde{J} \subseteq J$	set of bins used in solution
$\tilde{I} \subseteq I$	set of items packed in solution
x_{ij}	binary decision variable that indicates whether item i is packed into bin j ($x_{ij} = 1$) or not ($x_{ij} = 0$)
y_j	binary decision variable that indicates whether bin j is used ($y_j = 1$) or not ($y_j = 0$)
Z_i	binary decision variable that indicates if non-compulsory item i is packed ($Z_i = 1$) or not ($Z_i = 0$)

Objective 2: Minimize Load Imbalance

The additional objective considered in this work is the minimization of load imbalance. Load imbalance is defined as the deviation of the center of mass of the loaded items from a desired center of gravity (Fernandez, *et al.* 2013, Kaabi, *et al.* 2018, Liu, *et al.* 2008, and Trivella & Pisinger 2016). The load imbalance objective is only applicable to the 2D variant of the problem as it requires the geometric layout of the items in terms of their actual coordinate positions within the bins. The load imbalance formula is given by:

$$b = \frac{\sum_{j=1}^{m} \sqrt{(\lambda_{x,j} - \lambda d_{x,j})^2 + (\lambda_{y,j} - \lambda d_{y,j})^2}}{m},$$

where

$$\lambda_{x,j} = \frac{\sum_{i=1}^{n} x_{ij} \lambda_{x,i} \gamma_i}{\sum_{i=1}^{n} x_{ij} \gamma_i},$$

and

$$\lambda_{y,j} = \frac{\sum_{i=1}^{n} x_{ij} \lambda_{y,i} \gamma_i}{\sum_{i=1}^{n} x_{ij} \gamma_i}.$$

$(\lambda_{x,i}, \lambda_{y,i})$ represents the x-y coordinates of the centre of gravity of item i, $(\lambda_{x,j}, \lambda_{y,j})$ represents the x-y coordinates of the centre of gravity of bin j, and $(\lambda d_{x,j}, \lambda d_{y,j})$ represents the x-y coordinates of the desired centre of gravity of bin j.

The desired centre of gravity of a bin is defined as the middle point of the width of the bin and at the bottom side of the bin, as follows

$$(\lambda d_{x,j}, \lambda d_{y,j}) = (\frac{W}{2}, 0) \ \forall j \in 1, 2, ..., m.$$

As discussed in the literature review this objective is important in real world applications to ensure stability of bins and avoid unsafe scenarios. It is important to note at this point that in order to calculate the load imbalance for a particular bin, one needs to have the center of gravity of each of the items in the bin which means one needs to know exactly where it is placed. This will be important when considering the constraints of the problem; when checking if items fit in a particular bin, the position of each item must be determined and stored to be able to calculate the load imbalance objective.

Objective 3: Minimize Item Lateness

In many application areas of bin packing problems, time is an important factor. For example, courier and shipping companies have to schedule when items are sent and factor in how long they will take in order to make deliveries before due dates. This time component can be added to the MOGBPP model to make it more applicable in real-world scenarios and add a scheduling component to the problem.

The temporal characteristic is included in the MOGBPP model as an additional objective. It can be framed in two ways: Either in terms of due dates by minimizing the maximum lateness of items according to a pre-defined schedule (Bennell, *et al.* 2013, Arbib & Marinelli 2017 and Arbib, *et al.* 2021), or in terms of minimizing the time required to complete the task of packing (Naderi & Yazdani 2014 and Gomez & Terashima-Marin 2018). In this research, the former approach is considered, where each item has an assigned due date, and the objective consists of minimizing the lateness of the items. The maximum lateness of an item is defined as the largest deviation from its due date, d_i.

Two assumptions are considered in the model, taking into consideration practical application, as suggested by Arbib, *et al.* (2021). The first assumption is that there is a bin processing time, τ, a positive integer which is the time required to fill a bin with the assigned items. The value of τ is known, constant and independent of the items packed. This implies that bins are completed and delivered at a constant rate with an interval that cannot be smaller than the actual time required to pack the bins. The second assumption is that the completion time of any item i in the j-th bin is $K_i = j\tau$. This assumption implies that all the packed items in a bin are released simultaneously once the bin is packed. Based on these assumptions, the maximum lateness of an item is defined as

$$L_{max} = \max_{\{i \in I\}} \{L_i, 0\},$$

where $L_i = K_i - d_i$ is the lateness of item i.

3.1.2 Constraints

The **first constraint** is that all the compulsory items must be packed and items must be assigned to only one bin. This can be achieved by enforcing the following constraint in the model

$$\sum_j x_{ij} = 1 \text{ , for } i \in I^C.$$

For the non-compulsory items the constraint is modified slightly as these items do not have to be allocated, however if they are allocated they must only be allocated to one bin. The constraint is given as:

$$\sum_j x_{ij} \leq 1 \text{ , for } i \in I^{NC}.$$

The **second constraint** is that the number of each type of bin used must not exceed the available number of bins of that type. It is defined as

$$\sum_{j:\sigma(j)=t} y_j \leq U_t, \forall\ t.$$

The **third constraint** of the problem is that the items allocated to a specific bin must fit in that bin. In a 1D problem, this means that the sum of the weights of the items allocated to a bin does not exceed the bin's capacity, as follows

$$\sum_i \gamma_i x_{ij} \leq \Gamma_j y_j \ , \forall \ j.$$

In a 2D setting this constraint means that for some packing configuration in bin j, say F_j with width w_{F_j} and height h_{F_j}, the following inequalities must be satisfied

$$h_{F_j} \leq H_{\sigma(j)} \ , \forall \ j,$$

$$w_{F_j} \leq W_{\sigma(j)} \ , \forall \ j.$$

3.2 Constructive Packing Heuristic

As mentioned in Section 2.4, the CH algorithm is adopted in this research to generate the packing layout from the 2D problem. The CH makes use of a skyline approach which represents the contour of already packed items, and it can be expressed as a sequence of horizontal segments. The CH treats all items as compulsory when packing. The non-compulsory items are accounted for by the gene representation of the MOEA discussed in Section 3.3.1.

The initial skyline for a bin is a single segment representing the bottom of the bin. Each segment s_j is characterized by its x and y coordinates, its width, and the left and right-hand heights as shown in Figure 3.

	x	y	ω	h_1	h_2
s_1	0	2	3	4	2
s_2	3	4	3	0	0
s_3	6	1	2	3	2
s_4	8	3	2	0	3

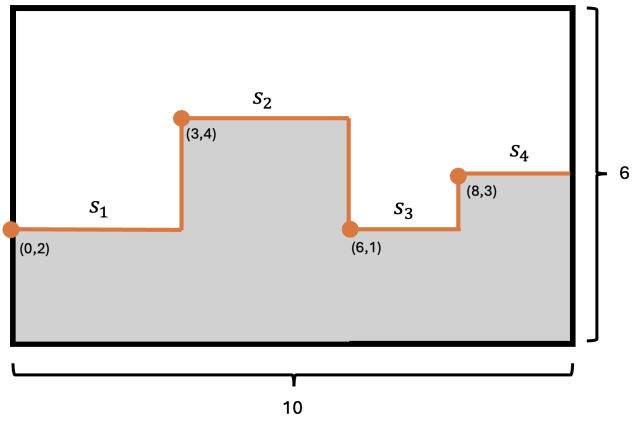


Figure 3: Skyline matrix information along with visual representation for four segments. Orange lines show skyline, and dots show left hand coordinates. The left-hand wall of the first segment and the right-hand wall of the last segment are equal to the difference between the bin height and the y-coordinate of the segment.

The algorithm starts by ordering the list of items according to non-increasing height, and where items have equal heights according to decreasing width. The skyline is initialized to be a single segment that represents the bottom of the bin. While there are unpacked items, the lowest and left most segment is selected. A check is done to see which items fit into the segment by comparing their width with the width of the segment. Items that could fit in the segment are assigned scores according to the scoring rule illustrated in Table 3. This scoring is based on the dimensions of the item as well as the dimensions of the segment. The item with the highest score is then chosen. If no items are found to fit into the segment, the skyline is updated by merging the selected segment with its lowest neighbour. This process is repeated until all items are

packed or until $h > H$, (the height of the bin). The pseudocode representation of the algorithm is provided in Algorithm 1.

Table 3: Scoring rule for determining the best item to fit into an available skyline segment. Parameters h_1, h_2 and w refer to the height of the left wall, height of the right wall and width of the segment respectively. This scoring rule is used by Leung, *et al.* 2011.

If	Conditions	Score
$h_1 \geq h_2$	$\omega =$ item.width and $h_1 =$ item.height	4
	$\omega =$ item.width and $h_1 <$ item.height	3
	$\omega =$ item.width and $h_1 >$ item.height	2
	$\omega >$ item.width and $h_1 =$ item.height	1
	$\omega >$ item.width and $h_1 \neq$ item.height	0
$h_1 < h_2$	$\omega =$ item.width and $h_2 =$ item.height	4
	$\omega =$ item.width and $h_2 <$ item.height	3
	$\omega =$ item.width and $h_2 >$ item.height	2
	$\omega >$ item.width and $h_2 =$ item.height	1
	$\omega >$ item.width and $h_2 \neq$ item.height	0

The CH algorithm returns the positions of the items in the bin as well as an indicator as to whether all the items in a set of allocated items fit into that bin. This algorithm is beneficial for checking whether the constraints are violated but in addition, it gives the position of the items which then allows the load imbalance for a particular bin to be calculated. The load imbalance calculations requires the centre of gravity of an item, which in the 2D case is the midpoint of the item. Because the CH returns the bottom left corner of each item, the midpoint can be calculated using the items dimensions and therefore the load imbalance calculation can be performed. This means that this CH must be done, and the constraint check completed before the load imbalance objective can be calculated for a particular solution.

3.3 The MOEA algorithm

The MOEA algorithm considered in this work is based on NSGA-II proposed by Deb, *et al.* (2002). This algorithm is used as it is able to maintain a better spread of solutions and converge to a better non-dominated front in many cases when compared to other elitist algorithms. In addition to its diversity maintaining mechanisms, the NSGA-II algorithm is more efficient than some other approaches with an overall complexity of $O(MN^2)$ (Deb, *et al.* 2002). The steps and components of the adopted MOEA algorithm are outlined in this section with a large focus on the non-dominated sorting that needs to be done before selection is performed.

There are a number of parameters that need to be selected for the MOEA algorithm and the different steps of the algorithm require different parameters and variables. The parameters used in this work are specified in the sections that follow. The results obtained may be sensitive to these parameters however the extent of parameter tuning was limited by the scope of the project. The specific parameters were selected based on best practices and guidelines established in the literature.

Algorithm 1 Constructive Heuristic (CH)

```
1: Initialize the skyline;
2: set h = 0
3: set numPackedItems = 0
4: while there are unpacked items do
5:     if h >= H (bin height) then
6:         return h, numPackedItems, list of (x, y) coordinates of packed items
7:     else
8:         find the lowest and left-most segment s of the skyline;
9:         if there is an item that fits into s then
10:            for each unpacked item i do
11:                s_i = score(i, h_1(s), h_2(s), ω(s))
12:            end for
13:            Select the first item R with the maximum score;
14:            if y + h(R) > h then
15:                h = y + h(R)
16:            end if
17:            if h_1(s) ≥ h_2(s) then
18:                pack R against the left wall;
19:                numPackedItems = numPackedItems + 1
20:                store (x, y) coordinates of bottom left hand corner of item;
21:                update the skyline;
22:            else
23:                pack R against the right wall;
24:                numPackedItems = numPackedItems + 1
25:                store (x, y) coordinates of bottom left hand corner of item;
26:                update the skyline;
27:            end if
28:        else
29:            update the skyline;
30:        end if
31:    end if
32: end while
33: return h, numPackedItems, list of (x, y) coordinates of packed items
```

3.3.1 Gene Representation

A gene representation of solutions is required in the implementation of the MOEA algorithm. A permutation representation is adopted in this work, where a solution is represented by a sequence of integer numbers of length n. The index position refers to the item number and the value at that position gives the bin number to which the item should be allocated. In this case the items are listed with the compulsory items first followed by the non-compulsory items at the end, and the items numbered from 1 to n. Bins are similarly labeled from 1 to m where m is the total number of bins available across all the bin types. If a value of 0 is assigned it means that that particular item is not packed. For a feasible solution, only the non-compulsory item indices may have zeroes to indicate that these non-compulsory items are not allocated while all compulsory items should have a bin number assigned to their index position. An illustrative

example is given in Table 4, where a packing solution of 10 items, of which 7 are compulsory and 3 non-compulsory, is given. In this case, the gene representation is given by 1131244023. That is, bin 1 contains items 1, 2, and 4; bin 2 contains 5 and 9; bin 3 contains items 3 and 10, and bin 4 contains items 6 and 7. Item 8 has not been allocated to a bin.

Table 4: Example of gene representation solution for packing 10 items, 7 of which are compulsory and 3 non-compulsory.

Compulsory/Non-Compulsory	C	C	C	C	C	C	C	NC	NC	NC
Item	1	2	3	4	5	6	7	8	9	10
Bin	1	1	3	1	2	4	4	0	2	3

3.3.2 Initialization

Two approaches to creating an initial population were considered in this work. The first approach was the use of a random initial population. A population size N is specified and for each of the N solutions, items are randomly assigned to one of the available bins by sampling from the available bins at each index position in the solution.

The second method to generating an initial population is by using the solution to the linear relaxation of the packing problem. The motivation for using the linear relaxation and details of it can be found in Section 3.4. The rationale for using this to generate an initial population is that it gives the MOEA algorithm a good starting point for the solution instead of starting randomly. Once the solution has been obtained from the linear relaxation, the population is created by sampling from the total number of available bins for each item where the sampling probability is given by the linear relaxation result. This can be done because the linear relaxation gives the proportion of each item to be allocated to each bin and these proportions must add up to one. In order to increase the diversity in this population, for each solution in the population a quarter of the items are randomly selected to be moved and are randomly allocated to one of the available bins. In addition, a number of random solutions are added to the population. This ensures diversity in the population to avoid premature convergence and ensures more of the solution space is searched.

After the initial population of solutions has been created, it needs to be sorted based on its objectives function values as well as on how many constraints are violated, *i.e.* how feasible the solution is.

3.3.3 Constraint Violation Scoring

To ensure solutions that violate constraints are not selected as parents or have less chance of being selected, one needs to calculate a violation score which will be used as almost an additional objective value when doing the non-dominated sort of the population of solutions.

A violation score of 50 is added to a solution for each of the compulsory items that have not been packed. For the constraint that ensures that items fit in their assigned bin, an additional violation score of 100 is added for every item allocated to a bin that is not able to fit in that bin.

3.3.4 The non-dominated sorting approach

Within a MOEA algorithm, at the selection step the best or 'fittest' solutions need to be chosen to be used in crossover to create the offspring population. In order to determine which solutions are 'fittest' when there are multiple objectives present, a non-dominated sorting approach is used to compare the solutions with one another.

The non-dominated sorting method proposed by Deb, *et al.* (2002) is adopted in this work. It requires that for each solution p, a domination count (n_p) is calculated which is obtained by summing the number of solutions which dominate the solution p. In addition to this quantity, the set of solutions that solution p dominates is also recorded, S_p. By calculating these two quantities, the solutions can then be sorted into different non-dominated fronts that will each be given a different rank. Specifically, the solutions that fall into the first non-dominated front will have a domination count of 0. Then for each of the solutions p in the first front, each solution in S_p is visited and its domination count is reduced by one. Any solutions that now have a domination count of 0 are in the second front and so the process continues until all solutions have been categorized into fronts. The first front is given a rank of 1, second front a rank of 2 and so on. The pseudo code for the fast-non-dominated-sort is given in Algorithm 2.

According to the non-dominated sorting approach, solutions with the lowest rank are preferred and should be selected in the selection process. To resolve ties and to maintain diversity of solutions, an additional criterion for selecting solutions, known as the crowd comparison approach (Deb *et al.* 2002), is also adopted. This approach computes the density estimation by means of a crowding distance. The crowding distance is the average distance of two points on either side of a particular solution along each of the objectives. The overall crowding distance value for a solution is the sum of individual distance values corresponding to each objective. Algorithm 3 shows how the crowding distance is calculated on a set of non-dominated solutions.

Based on the two aforementioned criteria, a solution with the lowest rank is preferred, and if the solutions belong to the same front, the solution with a higher crowding distance and therefore in a less crowded region is preferred.

3.3.5 The selection process

The binary tournament selection operator is considered in this work. This operator chooses two random solutions from the population. These two solutions are then compared using their rank and their crowding distance. The better solution according to the non-dominated search strategy outlined in section 3.3.4 is then kept as the 'winner' and becomes a member of a parent in the crossover step.

3.3.6 The crossover step

Uniform crossover is used to create an offspring solution from the selected parent solutions. For each gene (index position) in the parent solutions, a random number is drawn from a Uniform$(0, 1)$ which is used as an indicator to determine which gene from a particular parent is allocated to one or other child. This crossover operator was chosen because it helps maintain diversity in the population by mixing the genes of the parents in a more scattered way as opposed to 1-point or n-point crossover. In addition, the gene representation for the GBPP does not contain clear structure or parts that could be split into the different n-points for crossover so uniform crossover is adopted.

Algorithm 2 Fast-non-dominated-sort

1: **for** each $p \in P$ **do**
2: $S_p = \emptyset$
3: $n_p = 0$
4: **for** each $q \in P$ **do**
5: **if** $p \prec q$ **then**
6: $S_p = S_p \cup \{q\}$
7: **else if** $q \prec p$ **then**
8: $n_p = n_p + 1$
9: **end if**
10: **end for**
11: **if** $n_p = 0$ **then**
12: $p_{\text{rank}} = 1$
13: $\mathcal{F}_1 = \mathcal{F}_1 \cup \{p\}$
14: **end if**
15: **end for**
16: $i = 1$
17: **while** $\mathcal{F}_i \neq \emptyset$ **do**
18: $Q = \emptyset$
19: **for** each $p \in \mathcal{F}_i$ **do**
20: **for** each $q \in S_p$ **do**
21: $n_q = n_q - 1$
22: **if** $n_q = 0$ **then**
23: $q_{\text{rank}} = i + 1$
24: $Q = Q \cup \{q\}$
25: **end if**
26: **end for**
27: **end for**
28: $i = i + 1$
29: $\mathcal{F}_i = Q$
30: **end while**

Algorithm 3 Crowding-distance-assignment

1: $l = |\mathcal{I}|$
2: **for** each i, set $\mathcal{I}[i]_{\text{distance}} = 0$ **do**
3: **for** each objective m **do**
4: $\mathcal{I} = \text{sort}(\mathcal{I}, m)$
5: $\mathcal{I}[1]_{\text{distance}} = \mathcal{I}[l]_{\text{distance}} = \infty$
6: **for** $i = 2$ to $(l - 1)$ **do**
7: $\mathcal{I}[i]_{\text{distance}} = \mathcal{I}[i]_{\text{distance}} + \dfrac{\mathcal{I}[i+1].m - \mathcal{I}[i-1].m}{f_m^{\max} - f_m^{\min}}$
8: **end for**
9: **end for**
10: **end for**

3.3.7 The mutation operator

The mutation step adds small random changes to the individual genes in a solution within the MOEA algorithm. The mutation step adopted in this work is summarized in Figure 4. The mutation step is not applied to all members of the population, but only to some based on a mutation probability. The mutation probability is set to 0.3. Often this probability is made slightly smaller with a standard value being 0.1; however, to ensure diversity and particularly to ensure a set of solutions in the form of a Pareto front is obtained, this probability was set to 0.3. If a single solution is selected to be mutated based on this probability another random number is generated from the Uniform$(0, 1)$. Based on its value, one of four mutation operators is performed on the single solution. The first mutation operator is the swap mutation, and it occurs with a probability of 0.2. The swap mutation selects two random items and swaps which bins they are allocated to. The second mutation occurs with a probability of 0.3 and merges two bins together. Two random bins are selected that are already used in the solution and the contents of the one bin is merged with the other, placing all items into a single bin. The third mutation operator also occurs with a probability of 0.3 and this mutation splits the items in a bin. A random bin is selected and approximately half the items in that bin, which are chosen randomly, are placed into a different bin. The final mutation type is to move some items to a different new bin, and this occurs with a probability of 0.2. The number of items that are selected to be moved and the bins to which they are moved are chosen randomly.

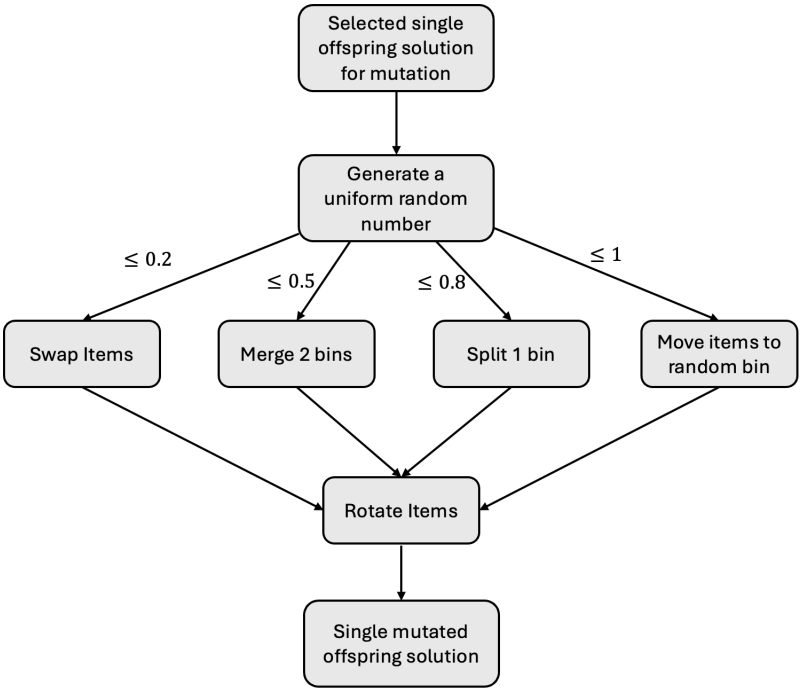


Figure 4: Summary of the mutation step adopted in the MOEA algorithm.

Once one of the four mutations have been applied to a single offspring solution, a rotation mutation is then performed. A random number of items in the solution are selected and rotated by 90 degrees. This means the item width becomes the item height, and the item height becomes the width. The rotations of the items are kept track of by a vector of 1s and -1s, with a 1 indicating that the original dimensions of the items hold and a -1 indicating that the item

should be rotated.

All of the specific mutation methods and the rotation of the items helps to maintain diversity in the population of solutions to ensure diverse Pareto fronts are obtained when optimising the problem.

3.3.8 The replacement operator

The NSGA-II uses elitism, and this is introduced by comparing the current population with previously best-found solutions. At the replacement step the best N solutions across the parent population and offspring population are chosen and kept for the next generation. In order to choose the best N solutions, the non-dominated sorting approach is implemented.

3.3.9 The main loop

A pseudocode representation of the MOEA algorithm is presented in Algorithm 4. For each generation t, the main loop of the algorithm starts with the replacement step which requires solutions from both populations to be sorted.

The incumbent and newly generated solutions are combined and scored (*resp.* sorted) according to the scoring system (*resp.* sorting strategy) presented in Sections 3.1.1 and 3.3.3. The best N solutions are chosen according to the sorting process (Section 3.3.4), which keeps the better solutions from lower fronts.

The new population returned from the elitism replacement is subject to selection, crossover, and mutation processes to form a new generation. The entire process is repeated until a pre-defined stopping criteria is satisfied. In this work, the algorithm is terminated after a set number of generations is run.

Algorithm 4 Genetic Algorithm: Main Loop

1: $\mathcal{R}_t = P_t \cup Q_t$
2: $\mathcal{F} =$ fast-non-dominated-sort$(\mathcal{R}_t)$
3: $P_{t+1} = \emptyset$ and $i = 1$
4: **while** $|P_{t+1}| + |F_i| \leq N$ **do**
5: crowding-distance-assignment$(\mathcal{F}_i)$
6: $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$
7: $i = i + 1$
8: **end while**
9: Sort$(\mathcal{F}_i, \prec_n)$ where $\prec_n$ is the crowd comparison operator
10: $P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$
11: $Q_{t+1} =$ make-new-pop(P_{t+1})
12: $t = t + 1$

3.4 Linear Relaxation of the Problem

As mentioned in Section 3.3.2, a linear relaxation (LR) of the problem is used to generate an initial population for the MOEA algorithm. In addition, the LR provides a lower bound for the problem. Because the optimal solutions for the test instances generated are not known, one does not know how good the solutions of the final Pareto front are, however the LR can help give an indication of how good the solutions are based on how close they are to the LR objective value.

A LR is adopted in this work as it is much easier to solve than an integer linear program (ILP) and it does not require a packing heuristic to be used to check that items fit. The LR is done by checking whether the sum of the weights (for the 1D problem) or area (for the 2D problem) of the items fits into the weight or area capacity of a specific bin. The lower bound is generated by allowing the decision variables to take on continuous values instead of constraining them to be integers. The rationale behind this is that the lower bound provides a benchmark against which solutions from the MOEA algorithm can be compared. Note that the lower bound will in general be an infeasible solution as it allows for fractions of bins to be used and fractions of items to be packed into different bins. The solutions from the MOEA algorithm on the other hand will be feasible solutions and not allow for fractions of bins to be used. In addition, the MOEA algorithm considers the dimensions of the items and how they are packed and not just whether the sum of the area of the items fits into the area of a particular bin (for the 2D case). Because of this, the solutions generated by the MOEA algorithm will not have objective values as small as the LR objective.

While the LR simplifies the problem, the way it is calculated using item areas means that the exact packing layout will not be solved, hence the load imbalance cannot be calculated for the 2D case. The LR is done using the cost objective without considering the other objective functions, as it simplifies the problem to be a single objective problem. This does not negatively impact the solution, as the LR is used only as an initial solution to guide the search. Since the MOEA algorithm deals with multiple objectives which may be conflicting, there could potentially be a deviation from the lower bound given by the LR objective. This is, however, not an issue since the LR is being used as a lower bound for comparison purposes.

An ILP model of the problem is given as follows:

$$\text{Minimize } \sum_j c_j y_j + \sum_{i=n_c+1}^{N} Z_i p_i,$$

subject to:

$$\sum_j x_{ij} = 1 \text{ for } i = 1, 2, ...n_c,$$

$$0 \leq \sum_j x_{ij} \leq -(Z_i - 1) \text{ for } i = n_c + 1, n_c + 2, ...n,$$

$$\sum_i \gamma_i x_{ij} \leq \Gamma_j y_j \ \forall \ j,$$

$$(\sum_i a_i x_{ij} \leq A_j y_j \ \forall \ j),$$

$$x_{ij} \in \{0, 1\},$$

$$y_j \in \{0, 1\},$$

$$Z_i \in \{0, 1\}.$$

This ILP aims to minimize the total cost of the packing allocation subject to the constraints previously outlined. The first two constraints ensure that each item must be packed into exactly one bin and all compulsory items must be allocated. The remaining constraint ensures items

fit into the allocated bin (with the 2D variant dealing with item area given in brackets above). The decision variables must take on binary values.

The LR model is similar to the ILP model except that now the integer constraints are removed and the variables x_{ij} and y_j are allowed to take on continuous values between 0 and 1. These variables now represent fractions of items or bins. The main difference to obtain the relaxation is therefore the following two constraints:

$$0 \leq x_{ij} \leq 1 \; \forall \; i, j,$$

$$0 \leq y_j \leq 1 \; \forall \; j.$$

Using the solution returned by the LR problem, an initial population to the MOEA algorithm can be generated and a lower bound for the problem obtained.

Another way to evaluate the solutions obtained from the proposed method would have been to compare them with solutions from alternative methods. One such method would be to combine the objectives using various weights and solve multiple instances to form an approximate Pareto front. This Pareto front could then be used as a baseline for comparison. However, implementing such an approach would have introduced computational challenges as this approach would require multiple optimisations to be run for each instance, one to test the proposed method and multiple to produce the baseline Pareto front from various weightings. Additionally, the method of combining objectives and solving the single objective problem with a linear relaxation could not be used because, as previously mentioned, the linear relaxation is not feasible for the load imbalance objective which requires the position of items to be known. Because of the computational costs associated with the approach of varying weights to form a baseline Pareto front, this method is not implemented and the linear relaxation on the cost objective is instead used as a basis for comparison.

4 Test Instances and Implementation

This chapter outlines the 1D and 2D test instances that were used in this work and details the implementation of the MOEA algorithm, specifying various parameters. The 1D instances were optimised across the objectives of cost and lateness while 2D instances were optimised across cost and lateness or cost and load imbalance. Cost and load imbalance can only be considered for the 2D case and not the 1D case as calculating load imbalance requires the position of each item in a bin to be known.

4.1 1D Test Instances and Implementation

The 1D test instances are the same instances used by Baldi, *et al.* (2012) which are based on the instances by Monaci (2002). Specifically, four classes of instances are considered, as detailed below.

- **Class 0** instances are the ones by Monaci (2002) and there are 300 instances in total. The number of items is varied between 25, 50, 100, 200 and 500. The item volumes come from one of three distributions: $U(1, 100)$, $U(20, 100)$ and $U(50, 100)$, which are specified as problem 1, problem 2 and problem 3 respectively. All items are compulsory and so a profit is not defined. The problems consider three and five bin types with volumes of 100, 120 and 150 for the three types and 60, 80, 100, 120 and 150 for the five types.
- **Class 1** uses the same instances as Class 0 except all items are now non-compulsory. Item profits are generated according to a uniform distribution, $p_i \in [U(0.5, 3)w_i]$.
- **Class 2** similarly uses the same instances as Class 0 with all items non-compulsory and item profit generated according to a uniform distribution, $p_i \in [U(0.5, 4)w_i]$.
- **Class 3** is a selection of 12 large instances (500 items) from Class 1 and Class 2 with a representative mix of the different characteristics such as volume, profit and bin types. For each of the 12 instances, five instances are generated with 0%, 25%, 50%, 75% and 100% of items being compulsory. The items have the same profit and volume across the five variations of compulsory proportions. This gives a total of 60 instances for this class.

These four different classes for the 1D problem are important as they represent increasing levels of complexity and capture different practical scenarios. Class 0 covers scenarios where all shipments are mandatory, with all items being compulsory. Class 1 and 2 are more complex than Class 0, introducing profit-driven decisions which capture scenarios where trade offs need to be made between profit from non-compulsory items and cost of additional bins. Class 3 reflects more realistic scenarios with a large number of items and a mixture of compulsory and non-compulsory items. Including these different classes allows one to evaluate the proposed solution method across various levels of problem complexity.

There is a specific naming convention used in this work to identify specific instances. The naming convention for Class 0, 1 and 2 instances is prob_x_Y_a_b_c where each component provides a specific detail about the instance. The first number, x, can be 1, 2 or 3 and identifies if it is problem 1, 2 or 3 which corresponds to the item volume distribution used. The letter Y can be A or B where A indicates instances with three bin types and B means it is an instance with five bin types. The next number, a, can take on the values 0, 1, 2 or 3 corresponding to 25, 50, 100 or 200 items in the instance. The letter b, indicates which class the instance belongs to 0, 1 or 2 and the final letter, c, gives the specific instance of the given combination ranging

from 0 to 9. Class 3 has a different naming convention. Class 3 contains 12 problems which are repeated five times each with varying proportions of compulsory items. These instances are labeled C3_comb_$a.b$, where a takes on a value between 1 and 12 identifying the instance and b takes on a value between 1 and 5 giving the proportion compulsory, either 0%, 25%, 50%, 75% or 100%. For example, C3_comb_4.1 is the 4th instance in class 3 with all items non-compulsory, while C3_comb_1.4 is the first instance in class 3 with 75% of items being compulsory.

Certain aspects of the problem needed to be simulated to apply to multi-objective optimisation, specifically for the calculation of the time-based objective. Bennell, *et al.* (2013) and Arbib, *et al.* (2017) use a bin processing time τ, of 100. The same is done here. Instead of using the bin number j to get the completion time of an item in a specific bin, an availability characteristic is added to each bin type, which represents the time at which the bin becomes available for packing into. This availability (or shipping time) characteristic is used as the j variable in the maximum lateness calculation. The assumption is made that more expensive bins will have an earlier availability while less expensive bins have a later availability. This is assumed as in general if one requires faster shipping or availability it comes at a premium. For the five bin types this time availability is set to 20, 15, 10, 5 and 1 for the least expensive to most expensive bins respectively. For three bin types 10, 5 and 1 are assigned for this availability characteristic.

The due dates are simulated using a Uniform $(\tau + 1, \tau\beta LB_{bin})$, where LB_{Bin} is the maximum of the availability time characteristic of the bins following the approach outlined by Bennell, *et al.* (2013). The parameter β is set to 0.6. This parameter scales the upper bound of the distribution of due dates and setting it to 0.6 makes the assumption that the due dates are moderately close to the maximum bin availability time, mimicking a practical scenario where due dates are neither very restrictive nor very lenient.

For the first three classes (0-2) only the 25, 50, 100 and 200 item instances were run. The 500 item instances were only considered for Class 3. This is because the algorithm takes longer to run for larger instances and so in the interest of time 500-item instances were only considered in Class 3. For each combination of number of items, number of bins and volume distribution, there were ten instances. Of these ten instances, five were run using a random starting point for the MOEA and five were run using the LR starting point. The population size for the MOEA algorithm was set to 200 and the algorithm was run for 500 generations.

4.2 2D Test Instances and Implementation

Test instances for the 2D problem were generated using the 2DCPackGen proposed by Silva, *et al.* (2014) with some modifications. This problem generator is able to generate 2D and 3D test instances for different cutting and packing problems and defines the problems according to the typology proposed by Wäscher, *et al.* (2007). The bin packing problem being considered here is a MBSBPP according to this typology and so these types of instances were generated using the 2DCPackGen.

A feature of interest of the 2DCPackGen is that it makes use of the Beta distribution as opposed to the uniform distribution when it comes to generating the dimensions of the items and bins. This means that different kinds of shapes can be specified, and the items and bins can then be generated based on those shape specifications. Silva, *et al.* (2014) specify 16 different types of shapes that the items and bins can take on and they use different parameters in the Beta distribution to ensure these shapes are created.

The generator allows one to generate different problems by varying many different parameters,

including: a seed for reproducibility, the number of test instances to generate for the particular combination, the minimum and maximum integer values for the size dimension of the bins and items, the shape and size of the items and bins for which there are 16 options, the number of different types of bins, the number of available bins per type, the cost of the bins, and the number of different items.

Doing many different combinations of all the available parameters leads to a large number of test instances which soon becomes infeasible to run in a realistic amount of time, particularly because the code for the 2D problem takes longer to run than for the 1D problem. Because of this only specific combinations were considered and fewer test instances were run overall in the interest of time. Despite fewer test instances being run, the results of those that are shown here allow one to see how the algorithm works, where there is room for improvement and how it optimises the multi-objective problem. The purpose of using the specified generator was mainly to generate the different item shapes instead of generating item dimensions using the uniform distribution. These instances were generated using the code given by Silva, *et al.* (2014) which needs to be run on a Windows machine.

The combinations considered are given in Table 5. From the table it can be seen that multi-objective optimisation was done for 180 2D problems. The size interval for the item shapes was $[20, 60]$. Three bin types were considered which were square with side lengths of 200, 300 and 400. The number of available bins varied depending on how many items were considered with less available of each type when there were fewer items and more available when more items needed to be packed.

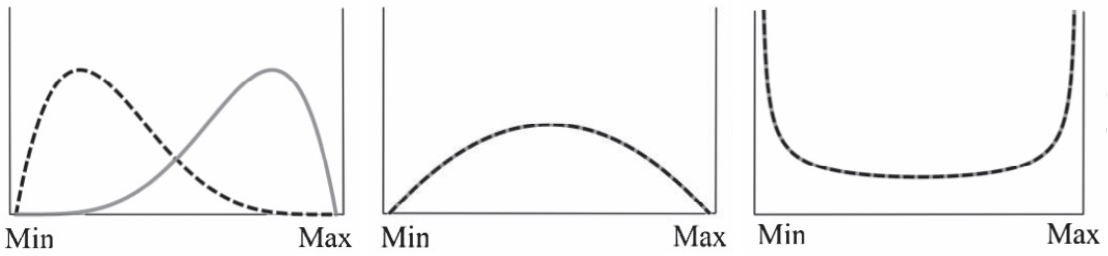
Silva, *et al.* (2014) make use of a Beta distribution varying the α and β parameters to create different item shapes given the input item size interval. Three different item shape categories were used for the 2D instances in this work, namely:

1. Long and narrow,
2. Average size and square,
3. Small and square, short and tall, long and narrow, or big and square.

The last category can be thought of as a 'mixture' of item shapes and will be referred to as the mixture category throughout the rest of this work. This mixture item shape category likely represents the most realistic packing scenario where items of various shapes and sizes need to be allocated and packed into bins. The shape of the Beta distributions to obtain these three item shape categories is given in Figure 5. In terms of the parameters' values in the Beta distribution, the long and narrow shape uses parameters $\alpha = 2$ and $\beta = 5$ for the item width and parameters $\alpha = 5$ and $\beta = 2$ for the item height. The average size and sqaure shape distribution uses parameters $\alpha = \beta = 2$ for item width and height. The mixture category which has items that are small and square, short and tall, long and narrow or big and square has parameters $\alpha = \beta = 0.5$.

As seen in Table 5, fewer instances were run for cases with larger numbers of items. For 20 and 50 items, 30 instances were optimised under each objective, 10 from each different item shape category. For 100 items, 20 instances were optimised under each objective, 10 from two of the item shape categories. For 200 items, only the mixture-shape category was optimised with 10 instances under each objective. In the results and discussion these instances will be named according to their combination number and specific instance (between 1 and 10) in that combination. For example, Combination 2.2 refers to the second instance in Combination 2

which has 20 items, a mixture of item shapes, three bins available of each type and is optimised over cost and lateness. The combination numbers will be given in the results tables which correspond to those given in Table 5.



(a) Long and narrow **(b)** Average size and square **(c)** Mixture of item shapes

Figure 5: Beta distribution shapes for three item shape categories. The dotted line is for the item width and the solid line for item height.

Table 5: Test instance combinations for 2D variant of the problem outlining the number of items, item shape, number of available bins per bin type and objectives considered. For each combination, 10 instances were generated and optimised giving a total of 180 2D instances.

	Problem Characteristics				
Combination	Items	Item Shape	Bins Available per Type	Second Objective	Number
Comb 1	20	Mixture	3	Lateness	10
Comb 2	20	Long and Narrow	3	Lateness	10
Comb 3	20	Average size and square	3	Lateness	10
Comb 4	20	Mixture	3	Load Imbalance	10
Comb 5	20	Long and Narrow	3	Load Imbalance	10
Comb 6	20	Average size and square	3	Load Imbalance	10
Comb 7	50	Mixture	3	Lateness	10
Comb 8	50	Long and Narrow	3	Lateness	10
Comb 9	50	Average size and square	3	Lateness	10
Comb 10	50	Mixture	3	Load Imbalance	10
Comb 11	50	Long and Narrow	3	Load Imbalance	10
Comb 12	50	Average size and square	3	Load Imbalance	10
Comb 13	100	Mixture	5	Lateness	10
Comb 14	100	Long and Narrow	5	Lateness	10
Comb 15	100	Mixture	5	Load Imbalance	10
Comb 16	100	Long and Narrow	5	Load Imbalance	10
Comb 17	200	Mixture	10	Lateness	10
Comb 18	200	Mixture	10	Load Imbalance	10

The profit for each item was generated by simulating $p_i \in [U(0.1, 0.3)a_i]$ where a_i is the area of the item as done by Baldi, *et al.* (2012). This was done as it is assumed that larger items generally are associated with more profit than smaller items.

In terms of compulsory versus non-compulsory items, the proportion of compulsory items is set to 80%. Due to time constraints, variations in this proportion were not able to be considered. 80% was chosen so that most items were compulsory, but some were allowed to be unallocated to see how the algorithm handled these types of items.

For the maximum lateness objective, the due dates for items were simulated in the same way as the 1D case discussed previously. Bin availability or shipping time also needed to be assigned for the different bin types. For the three bin types the bin availability value was set to 15, 8 and 1 for the least expensive to most expensive bin. Again, more expensive bins are assumed to be able to be sent earlier while less expensive bins come with the trade-off of later shipping time. This is done to ensure that the objectives are conflicting for the purpose of this multi-objective optimisation. In addition, it is likely that if one were to require a faster shipment it would be at a higher cost, as is the case with courier companies for example.

In terms of the cost of the bins, the smallest bin was given a cost of R100, the medium bin a cost of R500 and the largest bin a cost of R2000. The costs were assigned in this way as the area of the bins does not increase linearly. In addition, varying the costs in this way ensured the true Pareto front of solutions would be diverse and 'stretched' out and so these values were chosen to see if the MOEA was able to find a diverse set of solutions for each test instance. Within each type of bin, the cost was increased by a small amount to ensure preference of using the first bins within a specific bin type set. This was done in the LP formulation and the MOEA algorithm. These were small increases and would not affect which overall type of bin was used.

For each of the ten instances of the 18 combinations, five instances were optimised starting with a random initial population and five instances were optimised with an initial population based on the linear relaxation solution. These runs were done in parallel across five cores. The population size for the MOEA algorithm was set to 200 and the algorithm was run for 500 generations.

The code for the multi-objective optimisation of the GBPP was coded from scratch in R. The optimisation code for the 1D and 2D problems along with the text files for the respective test instances and the raw data results can be found in following Google Drive link:

`https://drive.google.com/drive/folders/1uuGVpHLAP1lMD2-AknYMJd791XnlVnST?usp=sharing`

5 Results

This chapter contains the results of this work, starting with the 1D problem, followed by the 2D problem. Within each section, the results are organised according to the different problem variants, including the number of items and objectives considered. In each case, the results are presented in tabular form and specific instance results are highlighted. Key themes and comparisons across the results are addressed in Chapter 6, where the main results of the work are discussed.

5.1 1D Results

The 1D results are presented by class, starting with Class 0, followed by Class 1 and 2, and then Class 3. Within each of these classes, the results are further detailed by instances size, with varying numbers of items present. As previously discussed, the 1D problem considers the optimisation of the cost and lateness objectives.

5.1.1 Class 0

The objective values obtained for the different instances in Class 0 Problem 1 are presented in Table 6 along with the lower bound obtained by the LR solution of the problem. Table 6a contains results for instances with three bin types and Table 6b contains results for instances with five bin types. Table A and Table B in the Appendix contain these results for problem 2 and problem 3 for Class 0 instances. For those instances that have a set of solutions only the lowest cost solution is given for the purpose of the tabular presentation of the results.

From Table 6a, it can be seen that as more items are present in a problem the cost and maximum lateness both increase as expected. In terms of the difference between the cost objective and its lower bound, this value also increases as the number of items increases when looking at the actual cost value. The second last column in the table gives the percentage deviation of the cost objective from the lower bound, calculated as $\frac{\text{Cost}-\text{LowerBound}}{\text{LowerBound}}$. The overall deviation from the lower bound varies between 6.91% and 22.84% across all instances in Table 6a. The deviations increase as the number of items increase but they are not as large as it initially looks upon inspection of the cost values. This result indicates that the MOEA algorithm performs relatively well at getting a cost objective close to the lower bound.

The percentage deviation is slightly smaller for the five bin types (see Table 6b) with the highest deviation across all instances being 18.88%. Again this points to the MOEA doing relatively well at finding a solution as close to the lower bound as possible while ensuring the solutions are feasible and do not violate any of the problem constraints. Interestingly when there are more bin types the deviation from the lower bound is smaller. This may be because there is more flexibility in terms of how to pack items with more different types of bins available and specifically for these instances, with smaller bin types available and therefore potentially less wasted space.

The change in objective function values across generations for test instance prob_1_B_3_0_8 is shown in Figure 6a. As one iterates over the generations, an improvement is seen in both objectives with the objective function value decreasing as expected and desired. Of interest here is the fact that 100 generations displays the characteristics of a Pareto front with the solutions showing a curve across the two conflicting objectives. The remaining 200 to 500 generations have a single objective value across all members of the population.

Table 6: Optimisation Results for Class 0 Problem 1 Instances. Problem 1A results are shown in the left table (a) and Problem 1B results are given in the right table (b). The linear relaxation bound on cost is given in brackets (LB). The % difference is calculated as $\frac{\text{Cost}-\text{Lower Bound}}{\text{Lower Bound}} \times 100$. The 'unique' column refers to the number of distinct solutions found in the final generation in terms of objective function values.

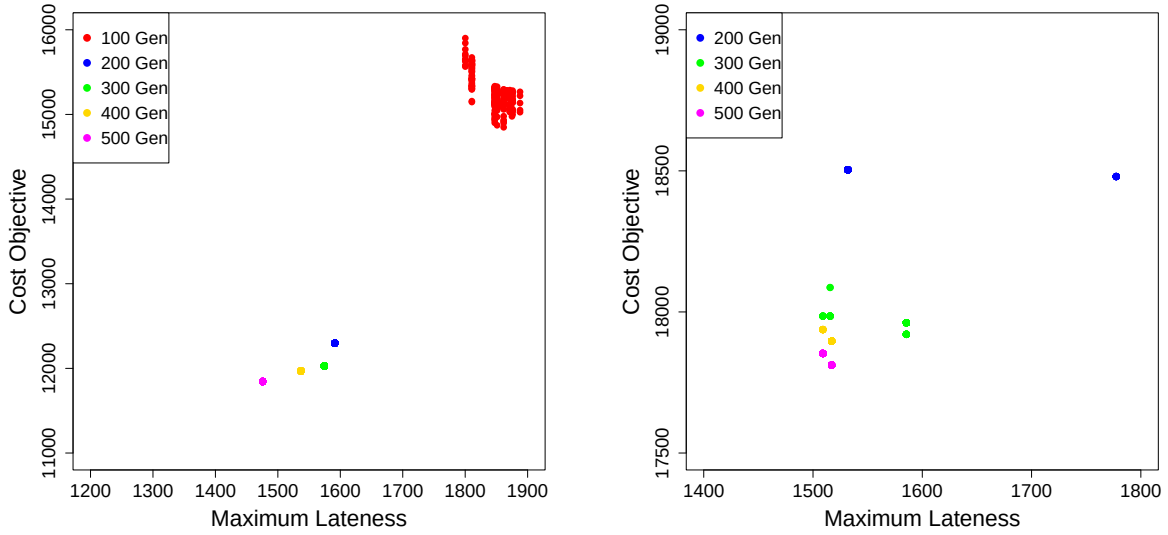
(a) Problem 1A Results

Items	Inst	Cost (LB)	Lateness	% Difference	Unique
25	1	1466.38 (1278.65)	474	14.69	1
	2	1830.31 (1587.58)	796.67	15.33	5
	3	1167.375 (1035.99)	601	12.66	3
	4	1593.05 (1341.97)	505.54	18.72	1
	5	1417.60 (1326.05)	971.85	6.91	7
	6	1462.63 (1309.85)	133	11.65	1
	7	1430.917 (1233.09)	607	15.99	2
	8	1445.25 (1282.7)	585	12.69	1
	9	1352.778 (1200.32)	456	12.72	1
	10	1316.825 (1117.2)	368	17.87	3
50	1	2505 (2194.64)	730.52	14.18	3
	2	2964.63 (2497.85)	625	18.71	2
	3	2960.21 (2542.99)	715	16.34	1
	4	2509.6 (2208.81)	611.05	13.65	1
	5	3303.19 (2895.42)	832	14.05	2
	6	3097.57 (2709.35)	649.69	14.34	1
	7	3333.5 (2831.99)	680.29	17.77	1
	8	3172.69 (2737.19)	588.93	15.86	1
	9	3025.72 (2604.36)	615	16.19	1
	10	3204.8 (2728.08)	746.04	17.47	2
100	1	6394.15 (5536.16)	738.36	15.56	1
	2	6370.96 (5342.63)	855.87	19.29	1
	3	6099.30 (5092.15)	677	19.81	1
	4	5671.93 (4787.2)	919.91	18.52	2
	5	5929.52 (5190.85)	796.61	14.18	2
	6	5662.467 (4918.12)	687.5	15.14	2
	7	5795.65 (4857.29)	765.83	19.33	1
	8	5951.09 (4988.22)	768.82	19.32	1
	9	5538.27 (4908.98)	917	13.88	1
	10	5928.49 (5171.61)	819.86	14.62	1
200	1	12801.18 (10431.77)	934.1	22.84	1
	2	12292.35 (10430.76)	886.99	17.97	1
	3	12370.59 (10517.48)	815.08	17.57	1
	4	12357.85 (10314.06)	841.86	19.83	1
	5	11712.41 (9933.697)	889.35	17.91	1
	6	12019.66 (9954.98)	883.43	20.71	1
	7	12633.04 (10535.72)	916.38	19.98	1
	8	12626.33 (10490.38)	862.62	20.35	1
	9	12413.51 (10364.72)	889.31	19.72	1
	10	11947.82 (10349.53)	874.96	15.63	1

(b) Problem 1B Results

Items	Inst	Cost (LB)	Lateness	% Difference	Unique
25	1	1458.87 (1323.42)	908.57	10.22	1
	2	1399.71 (1242.95)	859	12.61	2
	3	1262.66 (1172.67)	1089	7.67	1
	4	1487.71 (1344.45)	1358	10.65	2
	5	1469.96 (1338.40)	929.5	9.87	1
	6	1460.5 (1272.16)	705	14.83	2
	7	1479.37 (1281.23)	423	15.50	1
	8	1328.21 (1185.79)	960.74	12.03	2
	9	1087.54 (1028.73)	456	5.73	1
	10	1383.33 (1247.99)	1329.33	10.81	2
50	1	3054.39 (2649.83)	1543.28	15.27	1
	2	2909.83 (2589.36)	1630	12.42	2
	3	2742.47 (2401.74)	988.79	14.16	1
	4	3235 (2864.76)	1119.62	12.91	1
	5	2941.9 (2681.93)	933.18	9.67	4
	6	2611.93 (2305.06)	1171.47	13.29	4
	7	3126.85 (2706.16)	1158.09	15.51	1
	8	2964.4 (2681.93)	1328.67	10.54	2
	9	3219.4 (2856.87)	1205.51	12.66	3
	10	2609.49 (2319.11)	1119.32	12.55	1
100	1	5784.05 (4940.01)	1138.08	17.08	1
	2	5639.3 (4893.73)	1074.67	15.31	1
	3	6552.79 (5768.96)	1397.97	13.64	2
	4	6316.74 (5501.68)	1344.78	14.84	1
	5	6529.86 (5581)	1041.78	16.98	1
	6	5691.88 (5037.37)	1271.77	13.00	1
	7	5873.67 (5097.7)	976.43	15.22	1
	8	5519.75 (4894.74)	1230.82	13.76	1
	9	6006.42 (5164.96)	1167.06	16.30	1
	10	6111.83 (5256.5)	1307	16.29	1
200	1	12631.15 (10663.84)	1170.55	18.51	1
	2	11362.81 (9837.77)	1301.26	15.54	1
	3	11924.7 (10170.36)	1458.95	17.31	1
	4	13075.45 (11016.43)	1457.29	18.88	1
	5	11835.97 (10189.51)	1287.3	16.20	1
	6	11863.23 (10235.88)	1576.9	16.04	1
	7	11120.15 (9767.39)	1313.68	13.88	1
	8	12119.1 (10314.08)	1527.69	17.46	1
	9	11844.58 (10173.39)	1475.67	16.45	1
	10	12209.52 (10691.01)	1724.08	14.13	1

The remaining instances with 200 items exhibit a similar result except for 3 out of the 60 instances which displayed some form of Pareto front in the final generation. Two unique conflicting solutions were present in the final generation for instances: prob_2_A_3_0_3, prob_2_B_3_0_3, and prob_3_B_3_0_9. The evolution of the objective values for prob_3_B_3_0_9 across the generations is shown in Figure 6b.



(a) Test Instance: prob_1_B_3_0_8 **(b)** Test Instance: prob_3_B_3_0_9

Figure 6: Comparison of two 200 item instances from Class 0. (a) shows prob_1_B_3_0_8 containing only one unique solution in the final generation. (b) shows prob_3_B_3_0_9 containing two unique solutions in the final generation.

The test instances with fewer items were analysed to identify if Pareto fronts were present more frequently in the 500th generation for these cases. The 25 and 50 item instances had more cases where a desirable Pareto front was obtained in the final generation as seen in Figure 7.

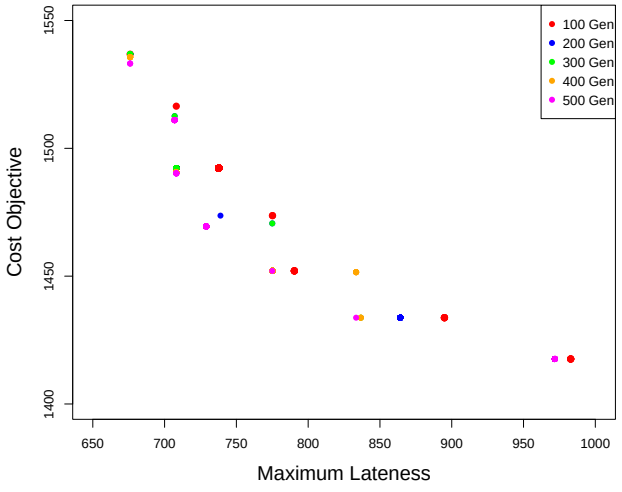


Figure 7: Evolution of objective function values for the test instance prob_1_A_0_0_4, which has 25 items and 3 bin types.

This is for instance prob_1_A_0_0_4 and here the Pareto front is more clearly seen. Again, it is seen that as one increases from 50 to 500 generations there is improvement in the objective function values, however, as opposed to the instances shown in Figure 6 this improvement is much smaller, with objective function values of the different generations solutions being much closer together.

Table 7 presents the distribution of instances with specific counts of unique solutions across various item number categories in the 500th generation. This table shows that as the number of items increases, the number of unique solutions in terms of objective values decreases, with more Pareto front type solutions sets being present for problems with 25 and 50 items and much fewer present for 100 and 200 item cases. However, even over all the combination types, there are many instances that had only a single unique solution in the final iteration of the MOEA algorithm.

Table 7: Number of unique objective values across different item number instances in Class 0.

Number of Items	Number of Unique Objective Values				
	1	2	3	4	5+
25	25	17	10	3	5
50	30	20	4	4	2
100	49	9	1	0	1
200	57	3	0	0	0

A comparison of the results with the LR starting point and a random starting point is shown in the convergence plots in Figure 8. The plots show the mean lateness (left hand side) and mean cost (right hand side) across the generations for two combinations, 50 items (top) and 200 items (bottom). The red lines indicate those runs that had the initial populations based on the linear relaxation solution while the blue lines indicate those runs starting with a random population of solutions. As expected, for the cost objective, the blue lines are in general above the red lines for the first few generations as those solution populations based on the linear relaxation would be expected to have better cost objective values even if they are constraint violating. After 100 generations there is less visible difference between the two different line colours for the cost objective.

It can be noted here that the mean cost decreases more smoothly than the mean lateness which has a more 'jumpy' convergence plot. The cost objective shows convergence with few changes after 300 generations while the lateness objective has not settled in the same way. A potential reason for the changes in the lateness objective with an unchanging cost is that items may be moved between bins that are already being used in the solution. Because extra bins are not added or current bins used not taken away, the cost remains the same, but the lateness objective can change if items are moved within the same set of bins. In terms of the mean lateness and mean cost at generation 500, there is not a clear difference between those populations that started with a random initial population and those that had a starting point based on the LR.

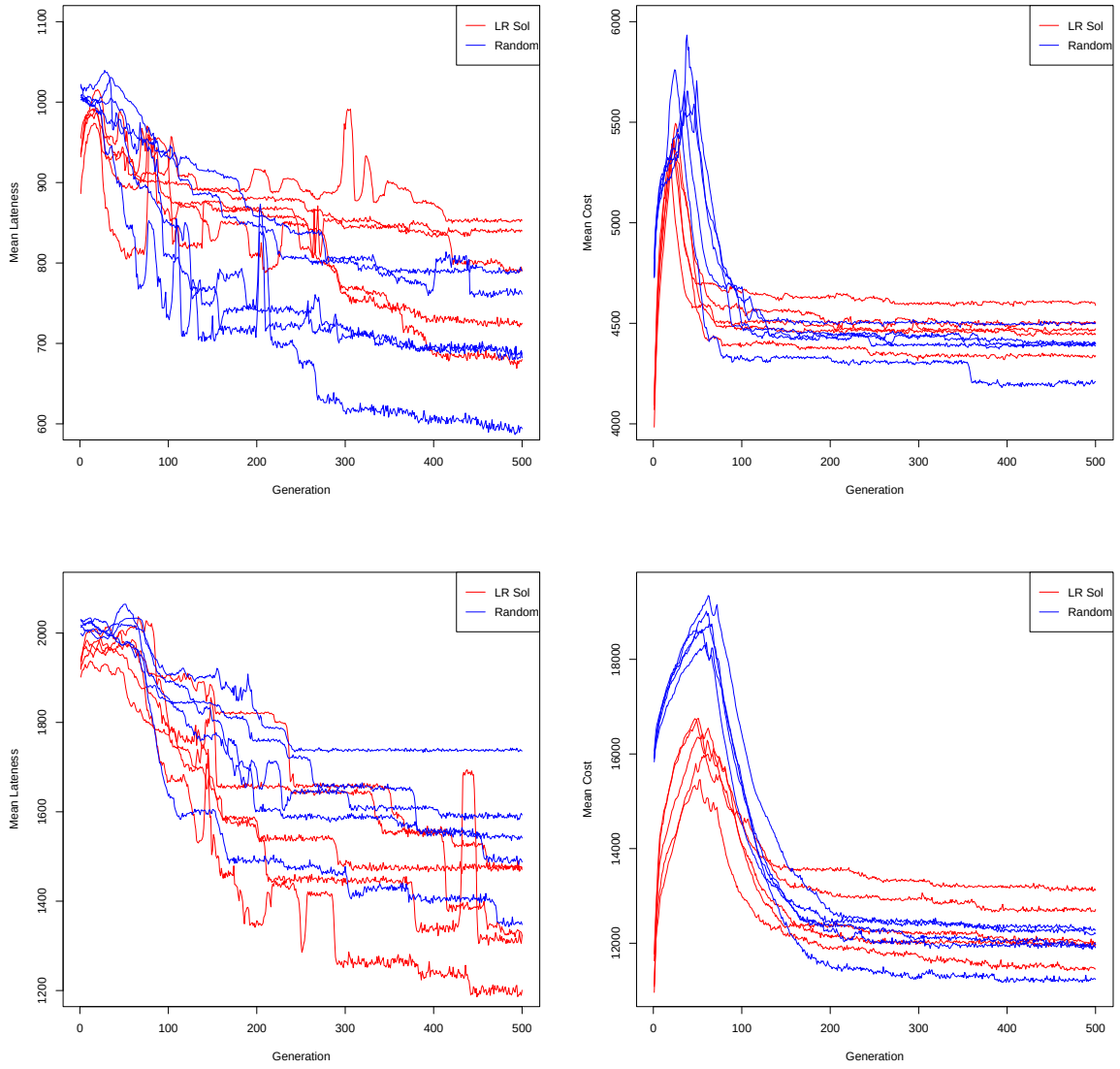


Figure 8: Mean lateness (left) and mean cost (right) over 500 generations for 10 instances of two specific feature combinations. The top two plots are for prob_3_A_1_0 which has 50 items and three bin types. The bottom plots are for prob_1_B_3_0 which has 200 items and five bin types.

5.1.2 Class 1 and Class 2

Problem 2A instances are specifically considered to investigate the effects of having all items being non-compulsory and to understand the differences between Class 1 and Class 2 instances resulting from different profit distribution. The results of the multi-objective optimisation of these specific instances are presented in Table 8, highlighting the number of unallocated items, the cost found by the MOEA algorithm, along with the LR lower bound solution, and the maximum lateness. These results are given for both Class 1 and Class 2 across the different categories of the number of items in an instance. For those instances that had more than one unique objective function value in the final generation, the lowest cost solution and associated maximum lateness value is reported here to see how close to the lower bound the MOEA

algorithm gets. The final two columns of Table 8 contain how many unique solutions exist in the final generation of the solution population, in terms of their objective function values. There may be multiple solutions in the population that result in the same objective function values so this unique column refers specifically to multiple objective function values, indicating a non-dominated front or set of solutions from which a decision-maker can make a trade-off.

Examining the number of unique solutions across both classes, a similar observation can be made as was done for Class 0. As the number of items increases there is a decrease in the number of unique solutions present in the final generation with very few instances, particularly in Class 2, showing any kind of Pareto front for 200 item instances. The evolution of the objective function values for prob_2_A_0_2_7 is shown in Figure 9 and the evolution of the objective function values for prob_2_A_3_1_1 is shown in Figure 10. Both plots show a Pareto front solution set in the 500th generation. While many of the instances in Table 8 have only one unique solution, these two examples show that the algorithm developed is able to produce a Pareto front as desired in some cases. The lack of Pareto front for many of the examples here may be as a result of the test instances. To investigate this further, different costs and shipping times are considered in the 2D case to ensure a larger conflict in objectives and to see if this changes the diversity present in the final iteration of the algorithm. It can be noted in Figure 10 that a non-dominated front containing four unique solutions is obtained in the 500th generation while in the 300th and 400th generation the fronts only contain two unique solutions and in generation 200 only one unique solution is present. This shows that the algorithm developed can find more diversity in later generations even if diversity is not present in earlier generations.

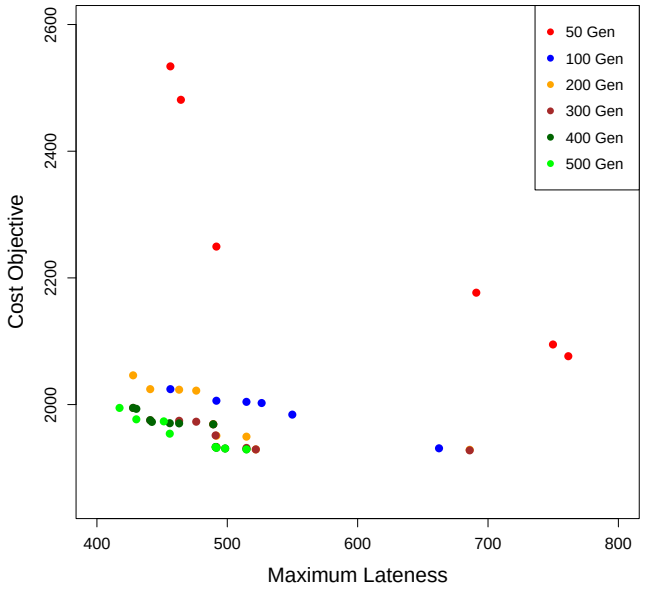


Figure 9: Evolution of objective function values over generations for prob_2_A_0_2_7.

Since Class 1 and Class 2 instances contain non-compulsory items, the number of unallocated items was monitored. As expected, when the total number of items increases there is an increase in the number of unpacked items for some instances but interestingly for a few of the 100 and 200 item instances all the items are packed while that is not the case for any 25 item instances. This may be due to the trade-off mentioned previously that the cost of leaving the items out is greater than the cost of the bins required to allocate the items. Across both classes and all items

Table 8: Comparison of optimisation results for Class 1 and Class 2 instances. The lower bound on cost is given in brackets (LB). The % difference is calculated as $\frac{\text{Cost}-\text{Lower Bound}}{\text{Lower Bound}} \times 100$. The 'unique' column refers to the number of distinct solutions found in the final generation in terms of objective function values.

Items	Inst.	Unpacked Items		Cost Objective (LB)		Lateness Objective		% Deviation		Unique	
		C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
25	1	7	3	1312.62 (1259.39)	1454.235 (1303.01)	314.82	651.77	4.23	11.61	1	4
	2	9, 10	2	1351.18 (1280.71)	1407.48 (1306.98)	531.850	572.69	5.5	7.69	3	2
	3	2, 3	1	1541.87 (1446.75)	1573.91 (1444.34)	523.29	667.14	6.57	8.97	3	3
	4	3, 4	3,4,5	1761.40 (1562.89)	1714.17 (1545.28)	424.33	330.33	12.7	10.93	6	2
	5	1	4,5	1944.29 (1695.85)	1771.21 (1632.61)	449	466.5	14.65	8.49	1	2
	6	9	3	1500.87 (1371)	1565.23 (1438.95)	360.57	377.28	9.47	8.78	2	2
	7	8	2,3	1806.40 (1620.67)	1907.6 (1669.25)	336	371.29	11.46	14.28	5	2
	8	6	1	1849.00 (1605.54)	1929.21 (1729.78)	315.88	514.71	15.16	11.53	3	8
	9	3,4,5	2	1492.42 (1359.96)	1591.82 (1445.73)	171.64	523.86	9.74	10.1	3	3
	10	2	3	1540.79 (1405.4)	1544.36 (1474.03)	342.14	160	9.63	4.77	1	2
50	1	7	10	3195.0 (2901.65)	3203.53 (2915.75)	498	563.33	10.11	9.87	2	2
	2	11	9	3448.72 (3092.5)	3553.33 (3153.71)	571.25	558.5	11.52	12.67	4	1
	3	10, 11	10	3126.98 (2855.87)	3109.02 (2800.47)	527	520.1	9.49	11.02	3	1
	4	10	9, 10, 11	3158.99 (2861.03)	3201.25 (2901.84)	536.38	583.1	10.41	10.32	3	3
	5	13	10	3317.15 (3036.44)	3393.41 (3050.38)	555.5	466.75	9.24	11.25	1	3
	6	3	0	3276.77 (2815.84)	3383.26 (2861.97)	500.55	635.17	16.37	18.21	3	2
	7	0	1	3308.8 (2662.81)	3164.23 (2756.48)	492	555.29	24.26	14.79	1	3
	8	0	3	3386.5 (2740.43)	3245.95 (2876.52)	412.93	296.55	23.58	12.84	1	3
	9	7	5	3323.53 (2996.61)	3256.97 (2921.66)	436.67	435.33	10.91	11.48	1	2
	10	10, 11	0	2746.66 (2575.35)	3047.49 (2681.83)	284.85	671.96	6.65	13.63	2	2
100	1	16	15, 16	6709.13 (5912.59)	6735.73 (5966.17)	673.16	657.52	13.47	12.9	1	1
	2	28	21	6100.4 (5501.68)	6174.13 (5579.86)	724.6	692.96	10.88	10.65	2	2
	3	29	15, 16	6377.89 (5892.14)	6809.17 (6023.16)	551.77	685	8.24	13.03	1	2
	4	35	16	6886.21 (6112)	7231.27 (6370.15)	629.62	647	12.67	13.52	1	2
	5	20	15	6391.9 (5772.4)	6634.12 (5911.27)	678.67	658.67	10.73	12.23	1	1
	6	14	5	6971.50 (5919.89)	6940.74 (5948.68)	648.92	666.48	17.76	16.68	2	2
	7	0	0	7341.32 (5819.27)	7225.65 (5855.42)	649.74	586.26	26.16	23.4	1	1
	8	0	13,14	7039.92 (5608.22)	6376.89 (5598.37)	584.1	674.97	25.53	13.91	1	2
	9	8, 9, 10	0	6956.30 (6018.29)	7233.46 (6058.34)	791.33	597.02	15.59	19.4	4	1
	10	10	8	6536.97 (5778.66)	6895.38 (5951.52)	700	631	13.12	15.86	1	1
200	1	36, 37	30	13516.35 (11813.64)	13659.92 (11835.45)	761.04	710.83	14.41	15.42	2	1
	2	33, 34	22	13561.18 (12010.53)	13933.36 (12031.73)	753.20	719.15	12.91	15.81	4	1
	3	41	39	13317.5 (11735.81)	13493.55 (11863.31)	774.99	745.49	13.48	13.74	1	2
	4	37, 38	28	12952.3 (11250.55)	13292.18 (11428.41)	742.03	687.33	15.13	16.31	2	1
	5	41	23	12331.35 (10749.39)	12938.74 (10971.06)	781.43	678.14	14.72	17.94	2	1
	6	13	0	13414.18 (11162.17)	14183.88 (11280.43)	686.74	700.99	20.18	25.74	1	1
	7	0	8	13685.45 (11013.95)	13396.06 (11216.82)	705.12	711.63	24.26	19.43	1	1
	8	18	8	14060.52 (11803.22)	14587.14 (11906.18)	657.37	726.27	19.12	22.52	2	1
	9	12	0	13888.81 (11703.83)	14789.5 (11991.25)	692.67	653.18	18.67	23.34	1	1
	10	0	0	14569.35 (11373.87)	14494.15 (11523)	737.48	708.71	28.09	25.78	1	1

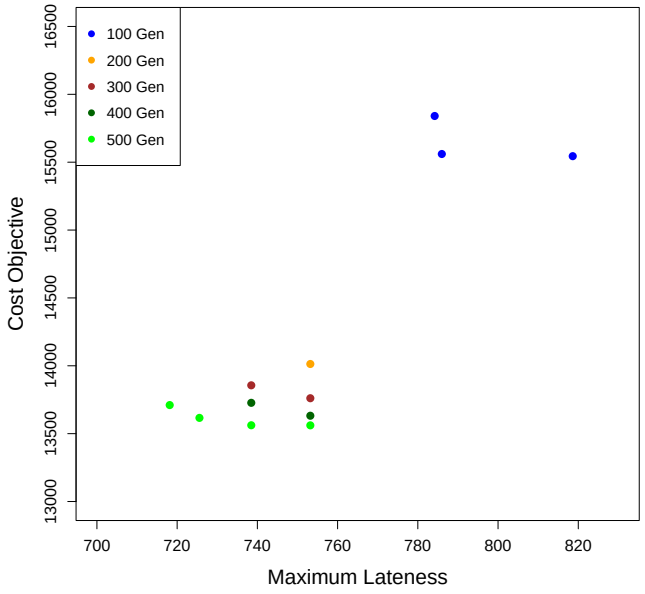


Figure 10: Evolution of objective function values over generations for prob_2_A_3_1_1.

combinations, in the 40 instances presented here, 31 instances have equal to or fewer unallocated items in Class 2 than in Class 1. This could be due to the fact that Class 2 items are more expensive and so leaving more items out leads to a higher cost incurred than in the Class 1 case. It may therefore be that in Class 2 more items are packed as the cost of the used bins is less than the opportunity cost of leaving the items out. However, there are some exceptions, for example prob_2_A_2_1_7 packs all items while in Class 2 items 13 or 14 are left out for this instance.

The average percentage deviations of the cost obtained by the MOEA algorithm from the cost lower bound are given in Table 9 for different item categories across the two classes. Again, it is noted that this deviation from the lower bound increases as there are more items in the problem. Across Class 1 and Class 2 the percentage deviations are very close together with Class 2 having a slightly smaller deviation for all except the 200 item case. The average deviation from the lower bound is less than 20% across both classes and the four item categories.

Table 9: Class 1 and Class 2 average deviation from lower bound on cost objective.

Items	Class 1	Class 2
25	9.91%	9.71%
50	13.25%	12.61%
100	15.42%	15.16%
200	18.1%	19.6%

In terms of the maximum lateness objective, there is no clear difference or pattern between Class 1 and Class 2 in terms of which class performs better on this objective. This is expected because the differing profit values should not impact on the lateness calculations. From Table 8 it is seen that the lateness objective increases as more items are present, as expected. When more items

are present, more bins are required which have later shipping times if all shorter shipping time bins have been used.

5.1.3 Class 3

Similar to what was seen in the results of the previous classes, out of the 60 instances in Class 3 only seven instances displayed conflicting solutions in the final generation of the MOEA algorithm. The remaining instances only gave one unique non-dominated solution (in terms of objective values) in the 500th generation.

Table 10 gives the cost and maximum lateness objective values for each of the 12 instances across the different proportions of compulsory items. The lower bound using the LR could not be calculated for this class because the constraint matrix grew very large, and the computation would not run. This issue was particularly evident when all 500 items were non-compulsory, as this required a large number of additional columns to be added to the constraint matrix. For those combinations that have more than one non-dominated front, the additional objective values are given in rows underneath respectively.

There are a few anomalies that can be noted from the results presented in Table 10. First, it is seen that two instances where all items were compulsory have one unallocated item, this means the solution obtained is not feasible and has a constraint violation score of 50. Particularly odd is to note that for C3_comb_6 where this occurs in the 100% instance, the 50% and 75% compulsory item instances had all items packed. Therefore, feasible solutions packing all items were possible and found but were not found by 500 generations in the last case.

In addition, there are some instances where all items are packed no matter how many items are compulsory, and these solutions are not the same as each other or do not contain additional solutions in the set. For example, C3_comb_8.1 (0%) has a cost of 37632.53 and a lateness of 871. The results for 25% and 100% cases have worse objective values but all items are packed so it should have reached the same objective values as the 0% results. The results for 50% and 75% cases are in between in terms of cost but worse in terms of lateness. These are therefore solutions that could form part of a Pareto front with the first solution but were not found as a set of solutions in each case. There is therefore room for improvement and more diverse solutions could be found in some cases.

In general, it can be seen that when less items are compulsory there are more unallocated items but in this case the number of items left out is relatively small. It is thus unlikely that the obtained results represent the true underlying Pareto front. Considering C3_comb_5 for example, the lowest cost is achieved when 25% of items are compulsory and the minimum lateness is achieved when 75% of items are compulsory (see Figure 11). The red triangle, which is the solution for all items being non-compulsory, falls in between the 25% and 75% solutions with a slightly lower cost than the 75% but a higher lateness, while compared to the 25% it has a lower lateness but higher cost. There is not a clear pattern as to how the cost and lateness changes as the proportion of compulsory items increases and this is due to how the cost objective is calculated. If the cost objective only considered bin costs, one would expect the overall cost to increase as more items become compulsory. Since the profit of the unpacked items is added to the bin costs, there is a trade-off between packing the items which leads to additional bin costs and leaving them unpacked, resulting in a 'lost profit' cost. The reason there are differences between the solutions for 50%, 75% and 100%, despite the three variants all resulting in all items being packed, is because the allocation of items to bins is different in each case. However,

Table 10: Class 3 optimisation results. There are 12 instances replicated 5 times with varying proportions of compulsory items, 0%, 25%, 50%, 75% and 100%. L_{max} is the maximum lateness objective value. N_U is the number of unallocated items.

Inst.	**0%**			**25%**			**50%**			**75%**			**100%**		
	N_U	**Cost**	L_{max}	N_U	**Cost**	L_{max}	N_U	**Cost**	L_{max}	N_U	**Cost**	L_{max}	N_U	**Cost**	L_{max}
1	0	30113.21	898	6	30212.25	887	0	30343.99	889	0	30651.49	857	0	30413.5	898
2	4	29414.52	1687	0	29505.08	1755	0	29828.13	1804	1	29644	1831	0	29556.85	1730
													0	29637.82	1667
3	0	31253.33	1813	2	31546.6	1823	0	31458.36	1737	0	31140.67	1841	0	30644.02	1823
							0	31538.79	1730						
4	0	30149.71	1785	5	30400.46	1785	0	30134.55	1796	0	30033.53	1705	0	30900.72	1858
							0	30218.05	1760						
5	2	30209.11	1749	4	29766.48	1765	0	30595.04	1749	0	30356.19	1589	0	30302.91	1699
	2	30294.06	1732							0	30462.20	1568			
6	4	36985.13	867	5	36984.62	881	0	36691.39	866	0	36667.36	880	1	37911.05	880
				5	37003.4	880									
7	0	37110.42	871	0	37824.34	883	0	37840.13	890	0	37957.87	887	0	37748.55	887
8	0	37632.53	878	0	37806.24	880	0	36708.56	891	0	37434.51	891	0	37861.98	873
9	0	37327.25	1855	0	37215.99	1782	2	37214.43	1855	0	37347.53	1784	0	37090.46	1705
10	0	48398.16	898	0	46997.44	863	2	47489	893	2	47340.29	893	0	48228.01	889
11	0	47907.3	884	1	47472.77	887	1	48034.51	885	1	47438.98	887	1	48482.35	891
12	0	47096.21	1892	1	46877.95	1812	0	47945.3	1860	1	47251.73	1856	0	47595.3	1860
				1	46978.22	1715									

ideally the algorithm would have selected all the non-dominated solutions across these variants to be potential solutions for each of them.

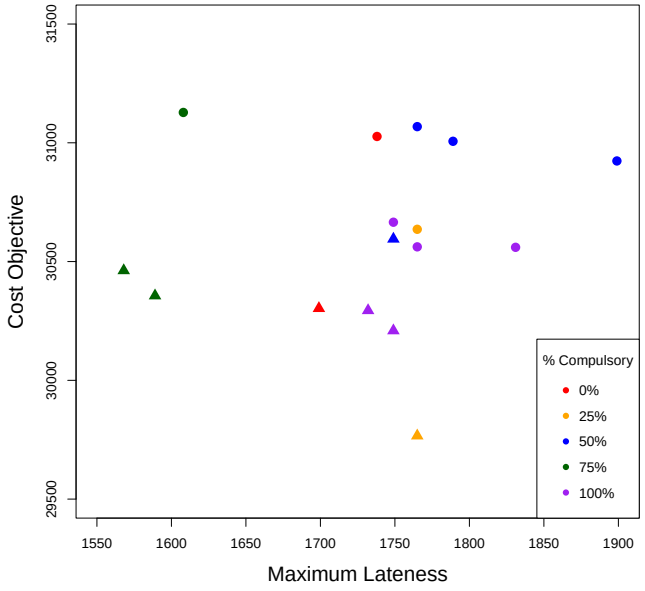


Figure 11: Objective function values in generation 400 (circles) and generation 500 (triangles) for C3_comb_5 for the varying proportions of compulsory items indicated in the legend.

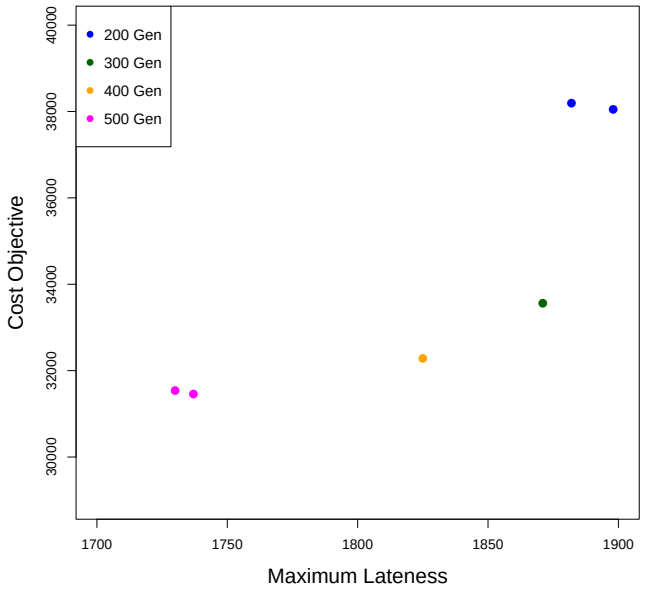


Figure 12: Evolution of objective function values for C3_comb_3.3

Another interesting result is the evolution of the objective function values across generations

for C3_comb_3.3 which has 50% of items compulsory as displayed in Figure 12. There are two unique solutions in generations 200 and 500 and only one unique solution in generations 300 and 400. In the 500th generation the difference between the two objective values came from having three items in different places. Specifically here it was found that two solutions that produced one of each of these had different allocations only for item 53, item 227 and item 413. The difference in the objective function values is small and comes from only a small change in the packing allocation. As was seen previously, if there is only one unique solution, in terms of objective function values, in a previous generation it does not stop the algorithm from finding more diverse solutions, as seen here from generation 400 to 500. Ideally one would want to see Pareto fronts across all generations showing a good diversity in the solution population, but this instance shows that more diversity in objective function values can appear in a later generation even if in a previous generation diversity in objective values did not exist.

5.2 2D Results

Having presented the 1D results, the 2D results are now presented. These results are presented according to the size of the test instances first considering 20 items, then 50, 100 and 200 items. In each of these size categories, both the cost and lateness optimisation and cost and load imbalance optimisation results are presented.

5.2.1 20 Items

Table 11a shows the cost and lateness optimisation results for 30 20-item instances that were considered while Table 11b gives the cost and load imbalance optimisation results.

Optimising Cost and Lateness Objectives
First considering the cost and lateness results, out of the 30 instances, 15 instances found three conflicting solutions, 14 had two conflicting solutions and one instance converged to only one unique solution. The % deviation from the lower bound is larger than was seen in the 1D case but this is due to the lower bound being the linear relaxation. If one were to consider only feasible solutions (where the whole bin and not just a fraction must be used), the lower bound for these instances would be 100 which is the cost of one small bin. Many of these solutions were able to get to this feasible lowest cost of 100. However, 11 instances had a lowest cost of 500 which is a large deviation from the lower bound, using a medium sized bin instead of a small bin.

The packing configuration for Comb1.1 with three conflicting solutions in terms of the cost and lateness objective values is shown in Figure 13. Two other such instances are presented in the Appendix in Figure A and Figure B. The packing layouts given shows how the constructive heuristic starts packing from the bottom of the bin and shows that there is some favouring of the left hand side.

Optimising the cost and lateness objectives for 20-item instances had fast convergence with the three unique solutions often being found in earlier generations, by generation 50 or 100. This is likely because the problem only has 20 items and so is small in scale. It is expected that only three solutions be found for these instances, packing items all into the smallest, medium or largest bin. The instances that have fewer than three unique solutions suggest that there is not enough diversity present in the population and the algorithm may have converged prematurely.

Table 11: Optimisation Results for 20-item instances, with cost and lateness results in (a) and cost and load imbalance in (b). The linear relaxation bound on cost is given in brackets (LB). The % difference is calculated as $\frac{\text{Cost}-\text{Lower Bound}}{\text{Lower Bound}} \times 100$. The 'unique' column refers to the number of distinct solutions found in the final generation in terms of objective function values.

(a) Cost and lateness optimisation results

Comb (Item Shape)	Inst	Cost (LB)	Lateness	% Difference	Unique
Comb 1 (Mixture)	1	100 (70.75)	1376	41.34	3
	2	100 (67.51)	1381	48.13	3
	3	100 (64.94)	1362	53.99	2
	4	100 (77.8)	1372	28.53	3
	5	100 (63.16)	1388	58.33	2
	6	100 (78.7)	1374	27.06	3
	7	100 (61.96)	1396	61.39	3
	8	500 (66.83)	676	648.17	2
	9	500 (93.89)	696	432.54	2
	10	105 (79.81)	1381	31.56	3
Comb 2 (Long and Narrow)	1	100 (72.07)	1344	38.75	2
	2	100 (73.14)	1344	36.72	2
	3	100 (70.54)	1213	41.76	3
	4	100 (75.63)	1368	32.22	3
	5	100 (70.8)	1396	41.24	2
	6	500 (74.58)	660	570.42	2
	7	500 (69.72)	683	617.15	1
	8	500 (71.43)	611	600	2
	9	500 (81.15)	657	516.14	2
	10	500 (76.79)	665	551.13	2
Comb 3 (Average Size and Square)	1	100 (75.13)	1345	33.1	3
	2	100 (73.74)	1385	35.61	3
	3	100 (70.86)	1325	41.12	3
	4	100 (79.82)	1381	25.28	3
	5	100 (70.48)	1341	41.88	3
	6	100 (78.21)	1344	27.86	3
	7	500 (68.47)	690	630.25	2
	8	500 (72.04)	693	594.06	2
	9	500 (86.96)	676	474.98	2
	10	100 (80.95)	1391	23.53	3

(b) Cost and load imbalance optimisation results

Comb (Item Shape)	Inst	Cost (LB)	Load Imbalance	% Difference	Unique
Comb 4 (Mixture)	1	100 (68.6)	95.41	45.77	1
	2	100 (89.77)	97.24	11.4	3
	3	100 (89.14)	97.99	12.18	4
	4	100 (73.24)	94.64	36.54	1
	5	100 (85.08)	94.17	17.54	1
	6	100 (73.02)	94.02	36.95	1
	7	100 (82.27)	94.48	21.55	1
	8	100 (78.27)	92.36	27.76	1
	9	100 (84.74)	92.82	18.01	1
	10	100 (78.68)	94.53	30.41	1
Comb 5 (Long and Narrow)	1	100 (71.28)	91.5	40.29	1
	2	100 (80.69)	92.51	23.93	2
	3	100 (78.65)	91.62	27.15	1
	4	100 (75.08)	94.39	33.19	1
	5	100 (79.75)	93.04	25.39	1
	6	100 (72.02)	91.07	38.85	1
	7	100 (77.54)	93.77	28.97	1
	8	100 (76.31)	93.71	31.04	1
	9	100 (79.65)	93.39	25.55	1
	10	100 (74.81)	91.13	33.67	1
Comb 6 (Average Size and Square)	1	100 (71.61)	95.9	39.65	1
	2	100 (85.82)	96.44	16.52	3
	3	100 (86.53)	96.72	15.57	2
	4	100 (78.85)	95.19	26.82	1
	5	100 (83.87)	94.17	19.23	1
	6	100 (74.33)	95.04	34.54	1
	7	100 (82.08)	93.45	21.83	1
	8	100 (78.33)	93.74	27.67	1
	9	100 (84.91)	99.14	17.77	2
	10	100 (79.68)	93.27	25.5	1

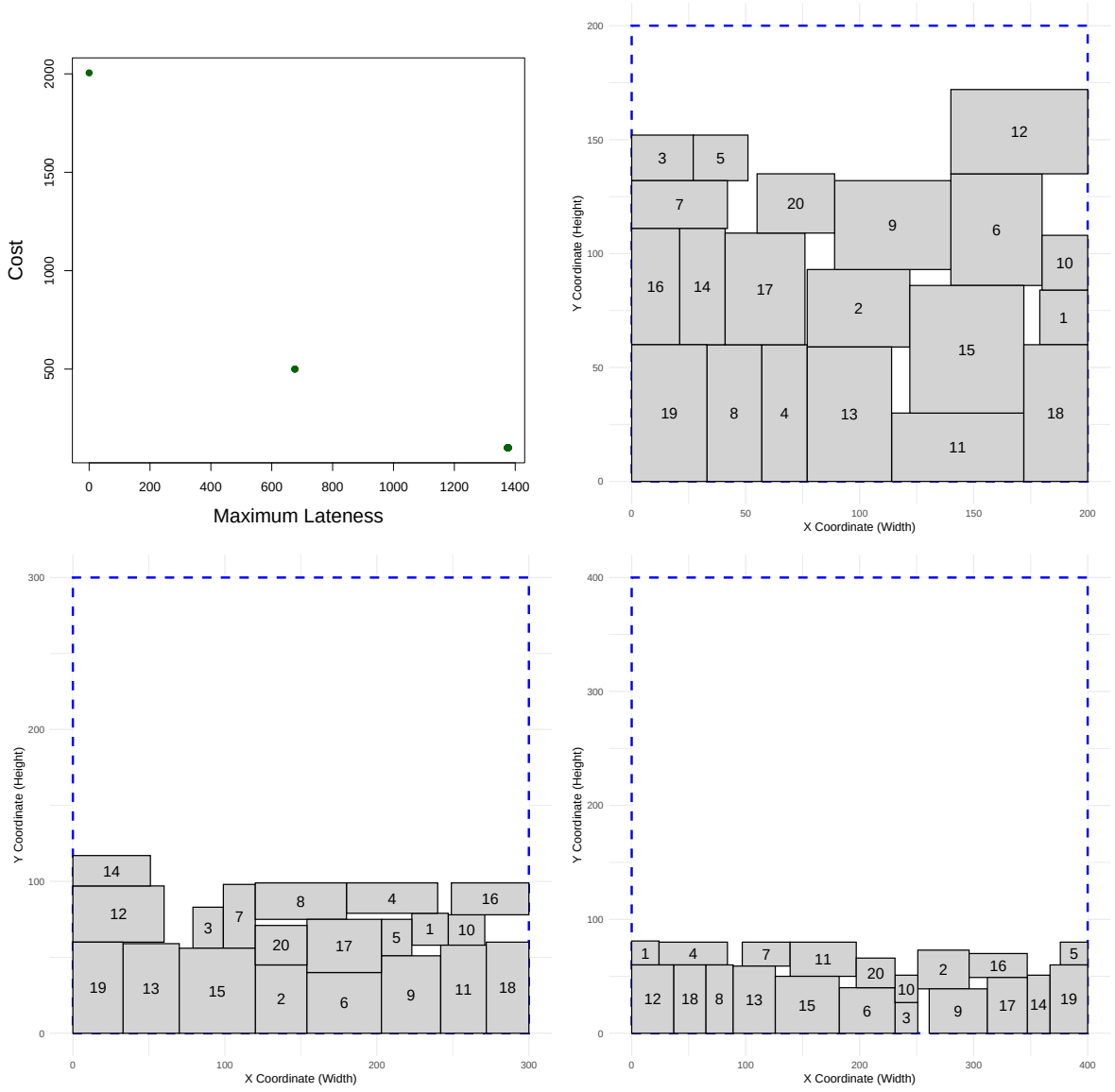


Figure 13: Instance Comb1.1 solutions and packing configurations. The top left plot shows the set of solutions found in the final generation of the MOEA. The remaining three plots show the packing layout of each of the three solutions.

Optimising Cost and Load Imbalance Objectives

Considering the optimisation of the cost and load imbalance objectives, it can be seen that across all instances the lowest cost option found is to pack all items into the smallest bin. Across the 30 instances, only six instances show more than one solution. Here, these differences occur because different non-compulsory items are allocated, which changes the cost objective due to the 'lost profit', as opposed to items being packed into different size bins. As an example, the objective function values and three associated packing configurations for Instance Comb6.2 are given in Figure 14. From these packing configurations the difference in the three objective values results from leaving different items out of the bin allocation. In the first case item 17 is left out, the second option leaves out item 17 and 18 and the last option includes all items resulting in the lowest cost but the highest load imbalance. All solutions here use the same bin size and there is no variation caused by using different numbers of bins. Considering the other five instances that

have more than one unique solution in the final generation it was found that all the variation was from different non-compulsory items not being allocated and from differing item rotations as opposed to using various bin sizes.

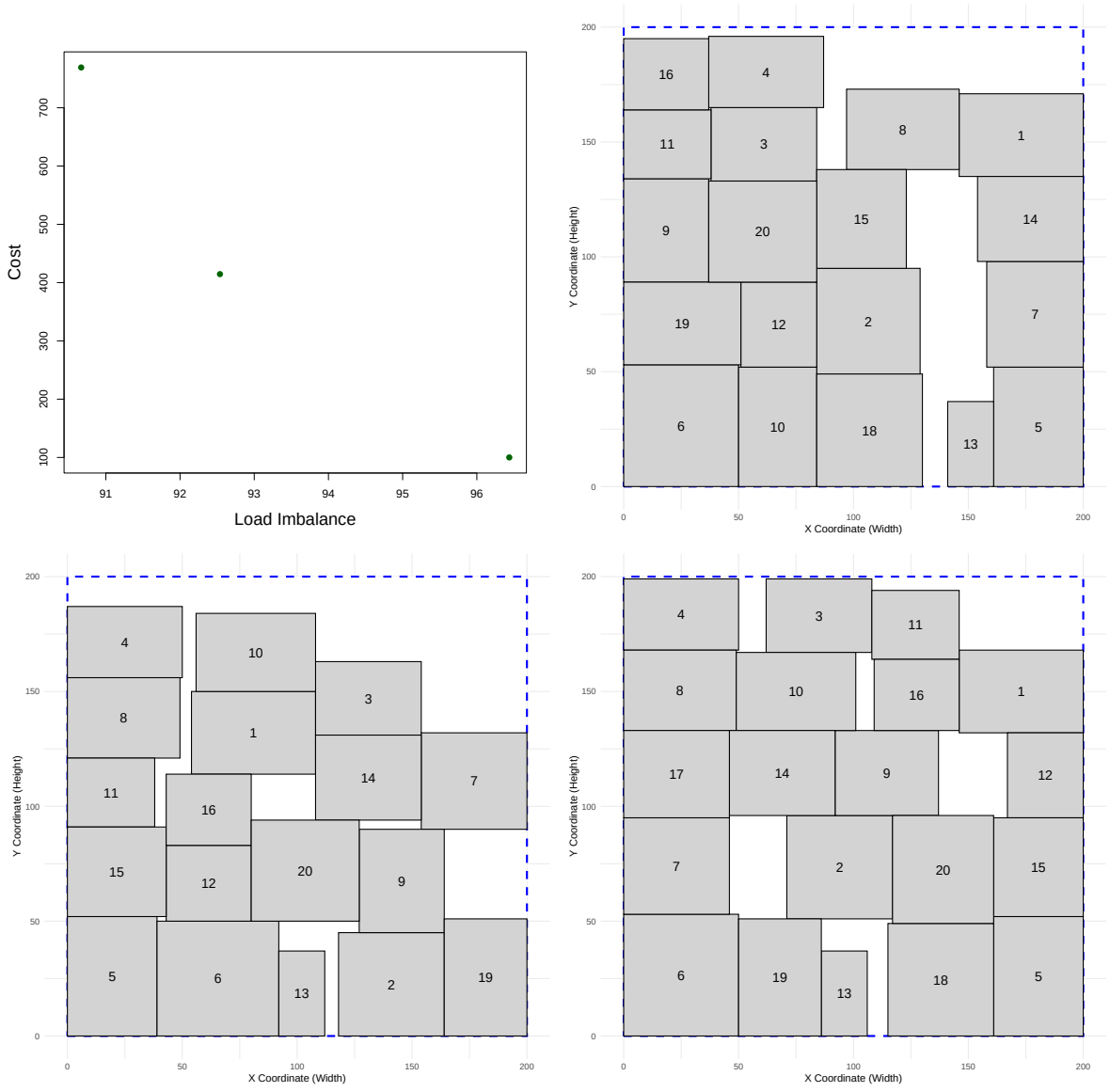


Figure 14: Instance Comb6.2. The top left plot shows the set of solutions found in the final generation of the MOEA. The remaining three plots show the packing layout of each of the three solutions.

5.2.2 50 Items

Table 12a shows the cost and lateness optimisation results while Table 12b gives the cost and load imbalance optimisation results for the 50-item instances that were considered.

Optimising Cost and Lateness Objectives

Out of the 30 instances in Table 12a, four showed three conflicting objective values, 22 had two conflicting objective values and four had only one unique objective function value found by the MOEA algorithm. In those instances where three conflicting solutions were found, it can be seen that the lowest feasible cost was obtained. There are two instances where a deviation

Table 12: Optimisation Results for 50-item instances, with cost and lateness results in (a) and cost and load imbalance in (b). The linear relaxation bound on cost is given in brackets (LB). The % difference is calculated as $\frac{\text{Cost}-\text{Lower Bound}}{\text{Lower Bound}} \times 100$. The 'unique' column refers to the number of distinct solutions found in the final generation.

(a) Cost and lateness optimisation results

Comb (Item Shape)	Inst	Cost (LB)	Lateness	% Difference	Unique
Comb 7 (Mixture)	1	307.5 (194.88)	1378	57.79	3
	2	500.0 (193.24)	691	158.75	2
	3	307.5 (179.33)	1390	71.47	3
	4	2000 (213.51)	0	836.72	1
	5	307.5 (217.69)	1361	41.26	3
	6	500 (190.85)	686	161.99	2
	7	500 (199.31)	672	150.87	2
	8	1005 (215)	694	367.44	2
	9	500 (208.82)	697	139.44	2
	10	500 (190.58)	698	162.36	2
Comb 8 (Long and Narrow)	1	500 (189.53)	691	163.81	1
	2	500 (190.31)	694	162.73	1
	3	307.5 (187.24)	1396	64.23	3
	4	500 (195.77)	653	155.4	2
	5	500 (199.36)	698	150.8	2
	6	502.5 (189.74)	680	164.84	2
	7	500 (189.53)	689	163.81	2
	8	500 (202.18)	696	147.3	2
	9	500 (194.43)	683	157.16	2
	10	500 (185.20)	689	169.98	2
Comb 9 (Average Size and Square)	1	500 (197.91)	695	152.64	2
	2	502.5 (199.61)	660	151.74	2
	3	307.5 (189.312)	1388	62.43	2
	4	502.5 (210.61)	688	138.59	2
	5	2005 (213.07)	0	841.01	1
	6	500 (196.11)	690	154.96	2
	7	505 (202.17)	678	149.79	2
	8	500 (208.77)	641	139.5	2
	9	500 (207.41)	632	141.07	2
	10	500 (191.13)	648	161.6	2

(b) Cost and load imbalance optimisation results

Comb (Item Shape)	Inst	Cost (LB)	Load Imbalance	% Difference	Unique
Comb 10 (Mixture)	1	307.5 (173.02)	300	77.73	2
	2	307.5 (204.75)	279.3	50.18	1
	3	307.5 (191.52)	288.06	60.56	1
	4	307.5 (196.4)	297.23	56.57	2
	5	307.5 (212.62)	284.15	44.62	2
	6	502.5 (197.02)	145.23	155.05	1
	7	500 (210.69)	145.61	137.32	3
	8	307.5 (183.02)	298.16	68.01	2
	9	602.5 (223.19)	240.28	169.95	1
	10	307.5 (196.18)	288.24	56.74	2
Comb 11 (Long and Narrow)	1	307.5 (181.65)	298.32	69.28	2
	2	307.5 (196.6)	291.47	56.41	2
	3	307.5 (187.27)	292.63	64.2	2
	4	307.5 (194.72)	292.5	57.92	1
	5	307.5 (198.92)	286.36	54.58	1
	6	505 (190.12)	144.76	165.62	1
	7	307.5 (195.56)	289.97	57.24	2
	8	307.5 (182.56)	293.92	68.44	2
	9	500 (205.6)	149.19	143.19	1
	10	307.5 (188.95)	297.85	62.74	2
Comb 12 (Average Size and Square)	1	307.5 (185.76)	299.25	65.54	2
	2	307.5 (205.34)	292.98	49.75	2
	3	307.5 (194.47)	294.74	58.12	2
	4	307.5 (201.25)	296.73	52.8	2
	5	307.5 (210.98)	280.67	45.75	1
	6	500 (199.31)	144.61	150.87	1
	7	307.5 (206.28)	291.98	49.07	2
	8	307.5 (187.7)	298.76	63.83	2
	9	600 (216.84)	243.59	176.7	1
	10	307.5 (198.4)	311.49	54.99	2

greater than 800% from the linear relaxation results is seen. In both cases, all items have been allocated the largest, most expensive bin, but obtain a minimum lateness value of zero. While such solutions are good if a decision-maker wants to minimize lateness, it is desired that the developed algorithm should also have found the lower cost, greater lateness solutions so that a trade-off could be made.

Figure 15 shows the packing layouts of the three unique solutions for instance Comb7.1. The top three boxes are for one solution, using the three lowest cost boxes resulting in the largest lateness. The medium box is the next solution followed by using the largest box which has greatest cost but lowest maximum lateness objective function value. These are three packing configurations that could be used by the decision maker depending on the trade-off they want to make between the two objectives. For the first packing option, there are many different ways the items can be allocated to the three bins that result in the cost and lateness given, the configuration shown here in the top three plots is just one such option. All three of the packing configurations shown here include all the non-compulsory items in the allocation.

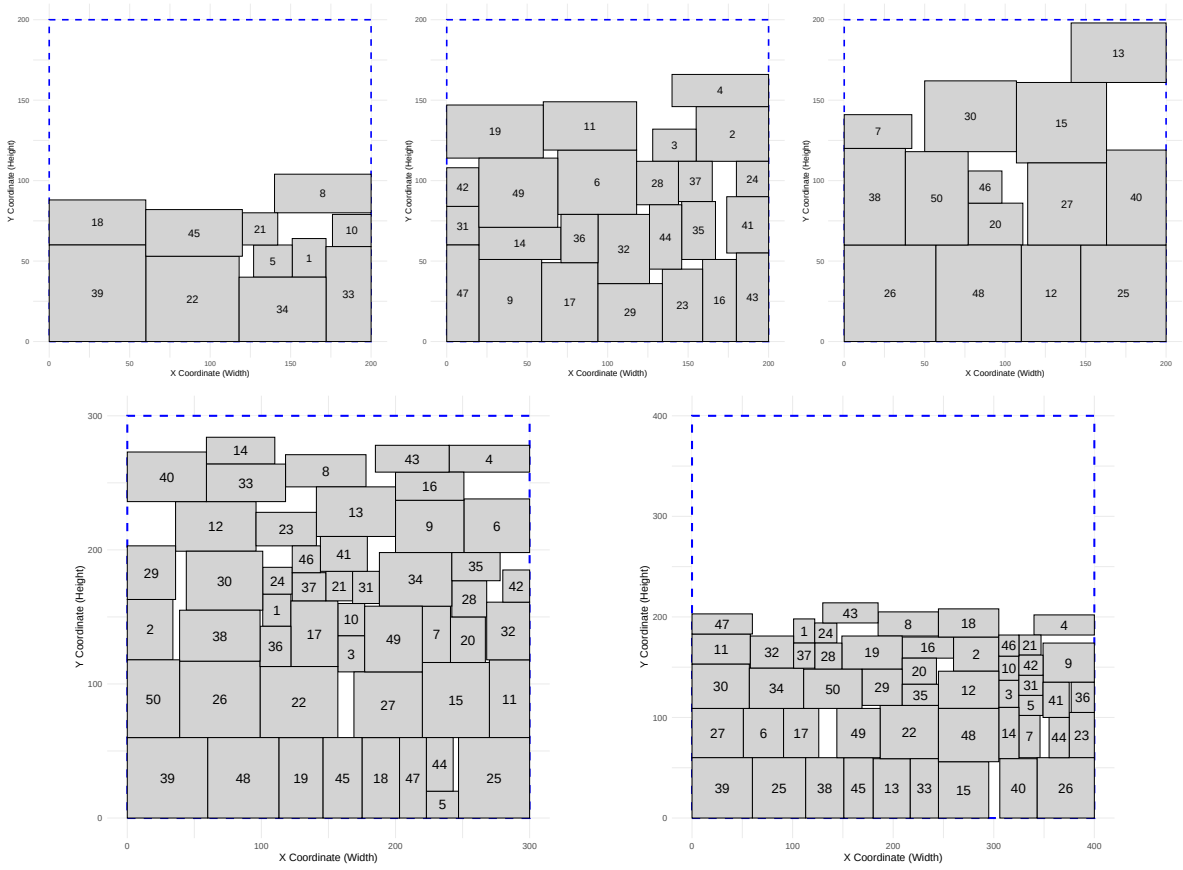


Figure 15: Three Packing Layouts for Instance Comb 7.1. The first packing solution is the top row, using three small bins, the second packing solution is the bottom left bin which is a medium sized bin and the final solution is the largest bin shown on the bottom right.

Optimising Cost and Load Imbalance Objectives

Many instances in Table 12b have a lowest cost option of 307.5 which is when three small bins are used. Slightly more variation is observed here compared to the 20-item case as more than one small bin is required to be used to allocate all the necessary items. Across the 30 50-item

instances, 11 have just one solution after 500 generations, 18 have two unique solutions and one instance has three unique solutions. In some cases the different solutions are a result of using different bin types, for example Comb10.1, the one solution places all items in a medium bin (cost 500) and the other in three small bins (cost 307.5), while other times the unique solutions are a result of leaving out different non-compulsory items as in the case Comb10.7. Overall, there is limited solution diversity in the final generation.

The deviations from the lower bound are relatively large but once again, because of the nature of the linear relaxation bound it is difficult to judge how good the solutions found are compared to the true optimal Pareto front. For example, in instance Comb 10.2, the sum of the item areas is 80856 units squared. The area of one small bin is 40000 units squared which means that if only using small bins all three bins would be required to be used leading to a cost of 307.5 if one takes the relaxation away in terms of fractions of bins being used. Therefore, while this instance says there is a 50% deviation from the lower bound, in terms of a feasible solution the lowest cost option has been found. Limitations of using the linear relaxation lower bound will be discussed further in Section 6.

5.2.3 100 Items

The results for optimising 100-items for cost and lateness and cost and load imbalance are given in Table 13a and Table 13b respectively.

Optimising Cost and Lateness Objectives
For the 100-item instances, one would expect that a more diverse Pareto front would be found because there is more variation in how one can pack items which should result in different cost and lateness objective values. From Table 13a it is seen that only two instances had four solution options after 500 generations, with three instances having three solution options, 11 having two conflicting solutions and four instances having only a single solution identified in terms of objective function values.

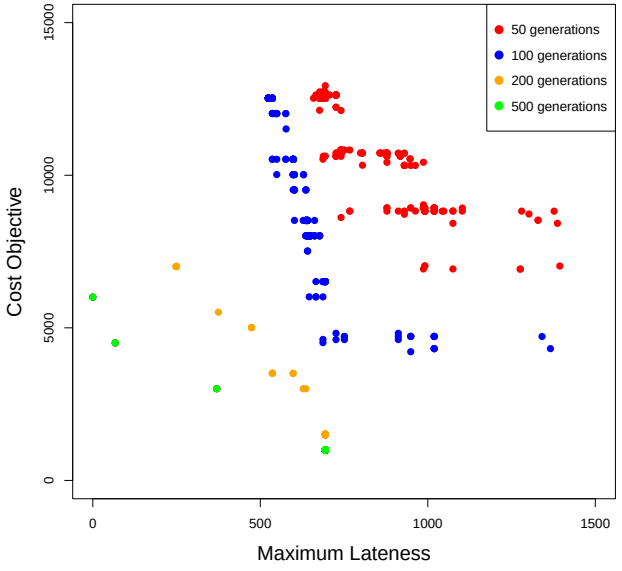


Figure 16: Evolution of objective function values for Comb14.10. Generation 300 and 400 values are the same as generation 500 values.

Table 13: Optimisation Results for 100-item instances, with cost and lateness results in (a) and cost and load imbalance in (b). The linear relaxation bound on cost is given in brackets (LB). The % difference is calculated as $\frac{\text{Cost}-\text{Lower Bound}}{\text{Lower Bound}} \times 100$. The 'unique' column refers to the number of distinct solutions found in the final generation.

(a) Cost and lateness optimisation results

Comb (Item Shape)	Inst	Cost (LB)	Lateness	% Difference	Unique
Comb 13 (Mixture)	1	1005 (386.43)	696	160.07	1
	2	1001.25 (399.06)	689	150.9	2
	3	812.50 (419.13)	1339	93.85	2
	4	1001.25 (393.94)	682	154.16	2
	5	1001.25 (403.38)	698	148.22	3
	6	1002.5 (405.02)	682	147.52	3
	7	1002.50 (399.92)	687	150.68	2
	8	1506.25 (400.57)	698	276.03	2
	9	1001.25 (387.05)	698	158.69	3
	10	4003.75 (405.26)	0	887.95	1
Comb 14 (Long and Narrow)	1	1001.25 (381.96)	694	162.13	2
	2	1001.25 (385.40)	690	159.8	1
	3	1001.25 (395.82)	697	152.96	2
	4	1001.25 (383.2)	690	161.29	2
	5	1001.25 (388.08)	692	158	4
	6	1001.25 (392.39)	697	155.17	2
	7	1007.50 (380.41)	698	164.85	2
	8	1001.25 (382.23)	692	161.95	1
	9	1001.25 (383.56)	698	161.04	2
	10	1001.25 (386.15)	694	159.29	4

(b) Cost and load imbalance optimisation results

Comb (Item Shape)	Inst	Cost (LB)	Load Imbalance	% Difference	Unique
Comb 15 (Mixture)	1	810 (416.14)	437.49	94.65	2
	2	803.75 (404.6)	436.78	98.65	3
	3	803.75 (411.97)	442.59	95.10	1
	4	805 (410.54)	435.81	96.08	1
	5	805 (394.3)	437.07	104.16	1
	6	805 (383.19)	444.19	110.08	2
	7	810 (409.68)	437.47	97.72	2
	8	806.25 (402.09)	440.55	100.51	2
	9	1105 (449.37)	338.27	145.9	2
	10	805 (388.31)	445.38	107.31	2
Comb 16 (Long and Narrow)	1	806.25 (396.34)	444.05	103.42	2
	2	807.5 (392.55)	442.85	105.71	2
	3	805 (387.4)	442.79	107.8	1
	4	807.5 (389.32)	437.9	107.41	1
	5	808.75 (382.13)	442.45	111.64	1
	6	805 (380.88)	447.47	111.35	2
	7	806.25 (393.28)	440.4	105.01	2
	8	805 (395.32)	437.25	103.63	1
	9	807.5 (411.4)	439.66	96.28	2
	10	805 (382.38)	439.23	110.52	1

Figure 16 shows different generations' objective values for instance Comb 14.10. The objective values obtained in generation 300, 400 and 500 are the same, which is why the generation 300 and 400 values cannot be seen. The shape of the collection of points at each generation shows the conflicting nature of the cost and lateness objectives. However, more diversity is required potentially in previous generations to find a more diverse final Pareto front. The lowest cost option here is 1001.25, however with five available small bins there should be a lowest cost option of around 500 with a larger maximum lateness if all items are allocated to the smallest bin size.

Figure 17 shows the mean cost and lateness over the generations for ten instances of Comb 13. The blue lines are those instances that had an initial population based on the LR of the problem while the red lines are those instances that started with random initial solution populations. Considering the mean cost on the right, it can be noted that for the first 200 generations, the instances that started with a random solution population generally have a higher cost value than those that started based on the linear relaxation solution. This is intuitive because the LR solution is minimizing cost. The opposite is seen on the left plot for mean lateness. Due to the conflicting nature of the objectives this also makes intuitive sense. Generally there is convergence from 300 generations onward.

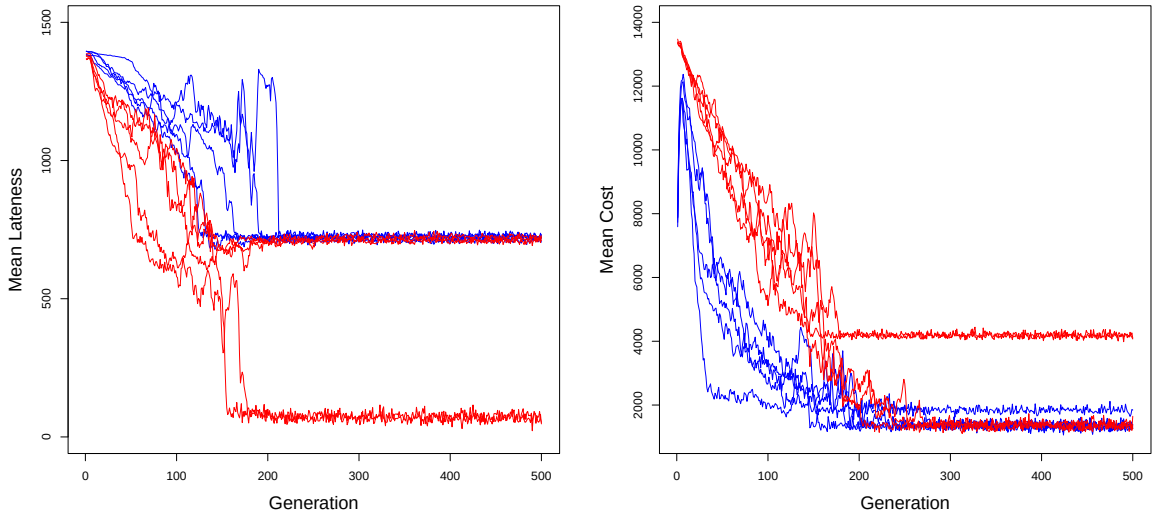


Figure 17: Mean lateness (left) and mean cost (right) over 500 generations for 10 instances of Comb 13. Blue lines represent instances that had initial populations based on linear relaxation while red lines represent instances that had random initial solution populations.

Optimising Cost and Load Imbalance Objectives

Table 13b shows the results for the 100-item instances optimising over the cost and load imbalance objectives. As seen with the previous results for load imbalance again here there is limited diversity in terms of the unique solutions found, with eight instances having only one unique solution, 11 having two conflicting solutions and one instance having three unique solution.

Figure 18 shows the evolution of the population in terms of objective function values for instance Comb 15.2. It can be noted here that the structure of this plot is different to what was seen previously for cost and lateness in Figure 16. There seems to not be the same pattern of conflicting objectives in the earlier generations. In addition, it can be noted that for certain

costs a number of different load imbalance values are obtained as seen by some horizontal points. In the final generation there are three unique solutions but two of them are very close in cost and load imbalance values with a cost of 803.75 with load imbalance of 436.78 while the other has cost of 810 and load imbalance of 436.38. The third solution is more obviously different from the other two with a cost of 1005 and load imbalance of 294.19.

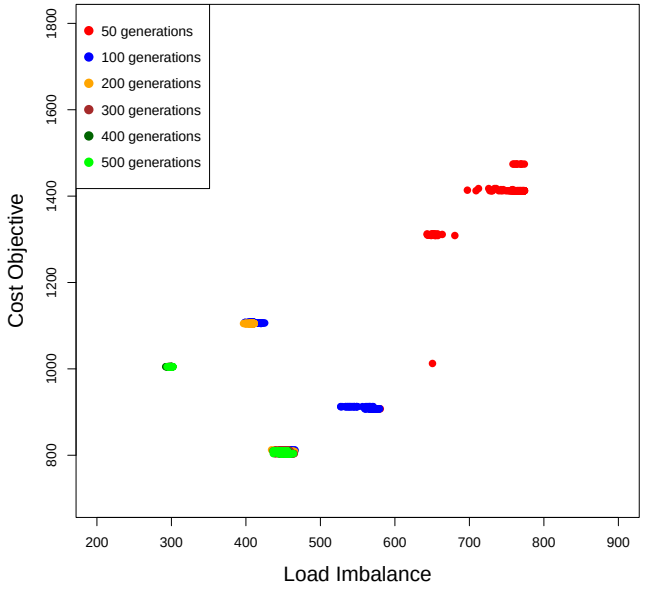


Figure 18: Evolution of objective function values across generations for instance Comb15.2.

Considering the LR starting point (instances 1-5) versus random initial starting point (instances 6-10), it can be noted that there is not a clear difference, with one class not having much lower deviations from the LR than the other but rather relatively similar deviations, particularly for the Long and Narrow items. For Comb15, the mixture of item shapes, one could observe that the solutions based off the linear relaxation have slightly smaller deviations from the lower bound than the remaining solutions, but this could also be a feature of the instances themselves. The convergence plots for these instances are shown in Figure G in the Appendix. As noted with other examples there is not much difference between the evolution of solutions that had an initial population based on the LR solution compared to instances with initial random population of solutions, except for the first few generations, where in general those that had random initial populations performed worse on the objectives than the LR solutions, as was expected. For this particular case, convergence happens very quickly with not much change after 100 generations. While it is desired for an optimisation method to converge, quick convergence may point to some solutions not being found in the search space and prematurely settling on one set of solutions.

5.2.4 200 Items

Optimising Cost and Lateness Objectives

Ten instances with 200 items were run optimising the cost and item lateness objectives, with the item shape specified to be a mixture of shapes. The results of these instances are presented in Table 14a. It is seen that in the final generation, there are relatively fewer packing configurations than what would have been expected. Some very large deviations are seen from the LR lower bound and this points to there possibly being lower cost options that the MOEA algorithm is

not able to find and a lack of diversity in the final solution set. All the costs found by the MOEA algorithm are above 2000. Comparing the first five and last five instances in this group, it is again seen that there is not a clear difference in the deviation % values between those instances that started with a population based on the linear relaxation solution and those that had random starting points.

The evolution of objective function values across generations for instance Comb 17.10 is shown in Figure 19. The evolution of the objective function values across generations shows that the MOEA algorithm is moving towards solutions with lower cost and lower maximum lateness and the conflicting nature of the objectives is seen in this specific instance, even if in other instances there is sometimes convergence of solutions. Of importance to note here is that the solutions of the 500th generation cover four of the five solutions in the 400th generation. The remaining visible solution from the 400th generation is not dominated by any of the solutions in the 500th generation and should not have been 'lost' as was the case here. This again points to some potential problems with the MOEA algorithm which will be discussed in Section 6. The four packing configurations associated with the four solutions found in the final generation are given in Figure C, Figure D, Figure E and Figure F in the Appendix.

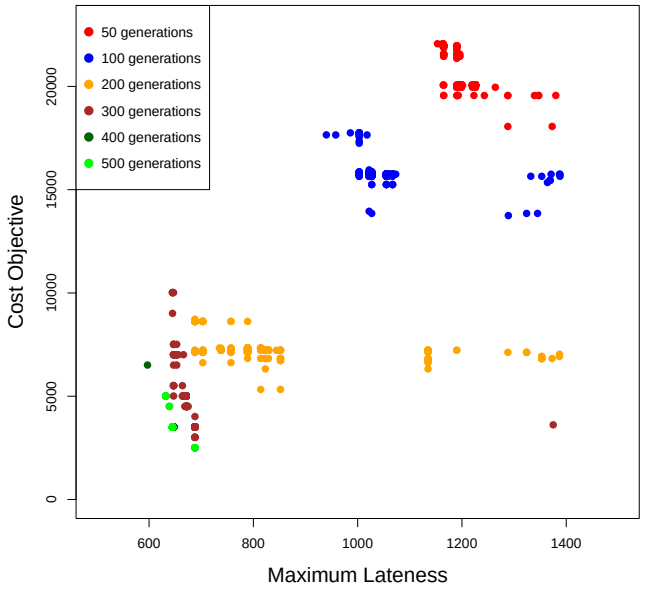


Figure 19: Evolution of objective function values across generations for instance Comb17.10.

Optimising Cost and Load Imbalance Objectives

Finally considering cost and load imbalance optimisation for 200-item instances, shown in Table 14b, again one can note there is limited diversity in the solutions after 500 generations, with six instances having only one unique solution in the final generation, and four having only two conflicting solutions. Here all the instances that had random starting points converge to a single solution while the first five instances that had initial populations based on the linear relaxation have four cases where at least two solutions are found. There are not enough examples here to conclude that this difference in starting point has caused this, and it is likely just a feature of the test instances as this was not found in earlier examples with fewer items.

Table 14: Optimisation Results for 200-item instances, with cost and lateness results in (a) and cost and load imbalance in (b). The linear relaxation bound on cost is given in brackets (LB). The % difference is calculated as $\frac{\text{Cost}-\text{Lower Bound}}{\text{Lower Bound}} \times 100$. The 'unique' column refers to the number of distinct solutions found in the final generation.

(a) Cost and lateness optimisation results

Comb (Item Shape)	Inst	Cost (LB)	Lateness	% Difference	Unique
Comb 17 (Mixture)	1	2505.56 (864.03)	693	189.99	3
	2	2510.56 (781.73)	688	221.15	1
	3	2509.44 (796.13)	697	215.2	3
	4	2108.33 (814.98)	998	158.7	2
	5	2511.67 (872.39)	699	187.91	1
	6	2505.56 (774.55)	695	223.49	2
	7	2509.44 (801.54)	696	213.08	2
	8	2005.56 (797.73)	698	151.41	2
	9	2005.56 (821.27)	694	144.2	4
	10	2505.56 (795.47)	688	214.98	4

(b) Cost and load imbalance optimisation results

Comb (Item Shape)	Inst	Cost (LB)	Load Imbalance	% Difference	Unique
Comb 18 (Mixture)	1	1715 (846.74)	992.16	102.54	2
	2	1612.78 (801.93)	882.19	101.11	1
	3	1615.56 (783.43)	893.99	106.22	2
	4	1617.78 (778.98)	890.03	107.68	2
	5	1715.56 (833.21)	1010.49	105.9	2
	6	2208.33 (857.85)	788.13	157.43	1
	7	2210 (854.61)	706.21	158.6	1
	8	1915.56 (812.61)	832.06	135.73	1
	9	1912.22 (809.03)	841.43	136.36	1
	10	1809.44 (806.62)	728.19	124.32	1

Figure 20 shows the evolution of the objective function values across different generations for instance Comb18.1. As was seen in Figure 18, it is again seen that there does not seem to be a conflict between the two objectives. Additionally, there is not much change between 200 and 500 generation, pointing to early convergence of the MOEA algorithm. One of the packing solutions in the final generation is given in Figure H in the Appendix. There are no obvious changes that should be made or certain bins that are very empty which suggests the solution is potentially close to what is optimal, and it is a feasible solution.

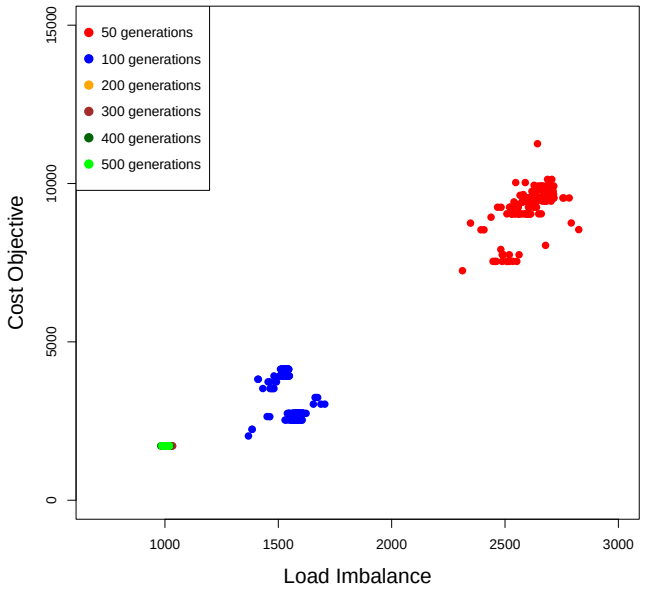


Figure 20: Evolution of objective function values across generation for instance Comb 18.1, minimizing load imbalance and cost.

Having presented the results obtained from the computational experiments using the developed MOEA algorithm, and having highlighted specific attributes of the results, a discussion of these results including limitations and implications for future research or improvements is given in Sections 6 and 7.

6 Discussion

The results presented in Section 5 are discussed in this section. Some features of interest related to the choice of objectives under investigation, the computational efficiency of the proposed model and the lower bound computation are presented first. This is followed by a discussion of the limitations and strengths of the developed MOEA algorithm.

6.1 Cost and lateness objectives

The results of the multi-objective optimisation for the objectives of cost and lateness show promise for the MOEA algorithm particularly for the 1D case where certain instances show diversity in the final generation. These objectives are conflicting in nature based on the way the bin availability characteristic and costs have been assigned. It was found in some cases that there was convergence to a single solution. This could have been a characteristic of the instances or a result of not enough diversity being introduced into the population in the MOEA algorithm. In the 1D case it was found that the MOEA algorithm seemed to perform well, particularly in terms of minimizing the cost objective. The deviation from the lower bound in the 1D case showed that the solutions found by the MOEA algorithm were nearing the true optimal solution in terms of cost. Slightly harder to assess is the lateness objective which did not have a lower bound calculated. Interestingly it was found that as the number of items increased from 25 to 200, the number of unique solutions in the final generation decreased. The population size remained constant across all item size instances, and this leads one to the idea that the population size compared to the number of items matters. In terms of the matrix of solutions in the population this would mean having many more rows than columns and this would potentially increase the initial diversity and different solution combinations in the population. However, the way in which the algorithm works, diversity should theoretically still be able to be found through the outlined crossover and mutation operators.

Moving to the 2D case, it was found that in some instances the MOEA algorithm successfully identified multiple solutions in the final generation while in other instances there was convergence to a single solution when in reality multiple solutions were available. An example of an instance where the MOEA algorithm successfully finds all possible solutions is Instance Comb1.1, which is shown in Figure 13. Here three solutions are found which correspond to three different cost and lateness objective values. For most of the 20-item instances one expects this kind of solution to be the case as they should fit into the smallest bin, medium bin, and then the largest bin, which increases in cost and decrease in lateness. However from Table 11a it was noted that not all instances had this occur and this points to the MOEA algorithm not always finding all the possible solutions that it should. One such example is instance Comb2.7 which is made up of long and narrow items. Here only one solution is found and all items are allocated to the medium size bin resulting in a cost of 500 and a lateness of 683. However, upon investigating this instance, it was found that it would be possible to fit these 20 items into all three bin sizes, shown in Figure 21. This means that there are additional solutions that the MOEA is not finding in its optimisation, and this is a flaw of the developed algorithm.

For 50, 100 and 200 item instances, it was seen that in some cases a set of solutions could be found, as shown in Figure 16 for example. However, in many cases the number of conflicting solutions found was only two or three, with four solutions being found for only a few of the 100 and 200 item instances. It would be expected that as the number of items increased, more unique solutions would be found depending on how items are arranged. One possible reason for

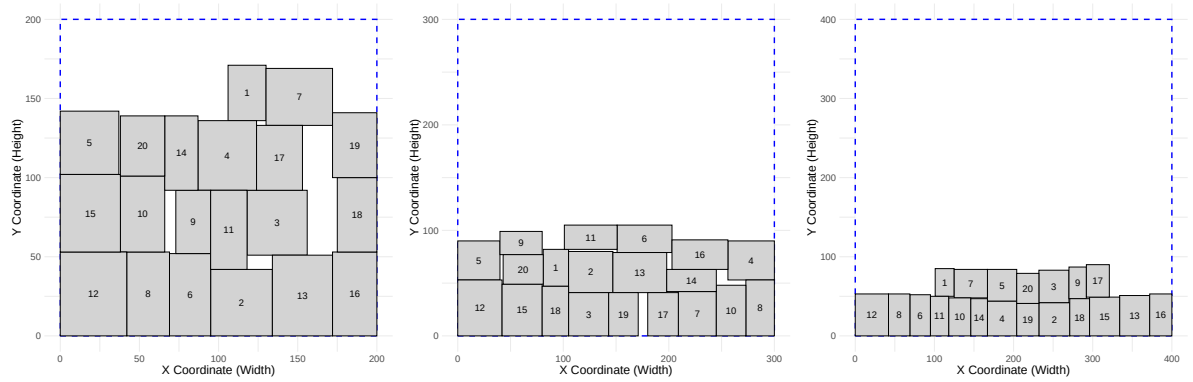


Figure 21: Three Packing Layouts for Instance Comb2.7. These are all feasible packing solutions, but only the middle solutions was found by the MOEA algorithm.

the lack of solution diversity is that if a set number of bins are being used and the MOEA is only changing the item allocation within these bins, then it would not change the cost objective and may only change the lateness objective value.

The results from optimising instances based on cost and lateness show that the developed MOEA algorithm can find a set of solutions across conflicting objectives; however, there is room for improvement as it is not finding all the possible solution options. Specifically, for what could be seen as a trivial case of 20 items, the MOEA should be finding all three possible solution options but in some instances it is not. Work therefore must be done on how to ensure a broader search space is considered so that the algorithm does not converge prematurely. Despite this flaw, the developed MOEA shows promise, decreasing the objective function values across generations and finding feasible solutions in both the 1D and 2D cases.

6.2 Cost and load imbalance objectives

A major feature of the cost and load imbalance results shown in Section 5 is a lack of diversity in solutions in the final generations. This is evident in some cases for cost and item lateness, but it is particularly evident for the load imbalance objective with many cases having only one unique solution in the final generation. A potential reason for the lack of diversity in solutions is because the cost and load imbalance objectives may not actually be conflicting based on how the instances have been set up and item weights assigned. Intuitively, if a bin is packed more tightly or densely it will likely be less imbalanced from the bottom center of the bin compared to a bin with fewer items in it which may be packed to one particular side. More full bins have a lower load imbalance, and more full bins are lower cost solutions overall compared to distributing items across many bins, thus the objectives are not conflicting. The weight of the items was made to be 0.1 multiplied by the item area so the weight was proportional to the area of the item. By filling as much area as possible it gives an alignment between the cost minimisation and load imbalance objectives. If the number of bins is minimised, therefore minimising the cost, the items are packed more densely and this would lead to a more balanced load.

However, Liu, *et al.* (2008), were able to get clear Pareto fronts with sets of solutions that were conflicting in their objective function values. Their objectives were minimising the number of bins used and the load imbalance. A difference in what has been done in this research compared to their work, is that here the weight is proportional to the size of the item. Liu, *et al.* (2008)

have item weights that are randomly generated from a Uniform$(0, 20)$ or a Uniform$(0, 100)$. In addition, they had a PSO operator[4] as part of their MOEA which was shown to help improve the diversity of the Pareto front.

A test was done on one of the 50-item instances, Comb10.6 which as shown in Table 12b had only one solution found with a cost of 502.5 and load imbalance of 145.23 when the weights were 0.1 multiplied by the item areas. Changing the weights to be randomly generated from a Uniform$(0, 100)$ instead of being proportional to area, resulted in two unique solutions being found, one with a cost of 505 and load imbalance of 125.6 and the other with a cost of 307.5 and a load imbalance of 247.36. The MOEA algorithm was able to find two solutions in the Pareto front where previously only one solution had been found and this shows how the test instance and way in which the weights are simulated can impact the results. However, it should be noted that this is still not a very diverse Pareto front with only two solutions present.

Because multiple tests were not done using different weights it is difficult to tell if the lack of conflicting objectives is due to the proportional relationship between weight and item area, leading to objectives that are not actually conflicting or if the MOEA algorithm is lacking the diversity one expected to see. It is likely that as seen with cost and lateness, the MOEA is not finding as much diversity as it should and in addition, the way the weights are simulated here is leading to non-conflicting objectives, which together lead to the problem of only single unique solutions being found in many cases for the minimisation of cost and load imbalance.

6.3 Computational efficiency of the MOEA algorithm

A boxplot of the time taken for Class 0 instance runs across the different number of items is shown in Figure 22. It can be noted that as expected, the 200 items take the longest to run while 25 items runs the fastest. This is due to how the solution matrix is set up with more columns when more items are present. Because more items need to be checked in terms of whether they fit in particular bins the algorithm takes longer to run for larger instances.

Figure 23a below gives boxplots for how long the 500 generations took to run for different size instances for the cost and lateness objectives while Figure 23b shows the time taken for instances optimizing cost and load imbalance. Like what was seen in the 1D case, as the number of items increase, the time taken to perform the optimisation using the MOEA algorithm increases. Note that the relationship shown between time and number of items is not of linear complexity but sublinear complexity because the increase in number of items is not uniform. However it is still seen that as the number of items increases, the time taken to run also increases. For 20-item instances, the 500 generations took less than an hour to run. 50-item instance took between 3 and 4 hours to run, 100-item instances took between 5 and 7 hours to run and 200-item instance took 9-10 hours to run. The reason for this increase is because of the structure of the gene representation. The gene representation gets longer as more items are included. In general, this means more bins are used which require more constraint calculations to be done checking that items fit into the different bins and this requires the constructive heuristic. The 20 items had the smallest variation in time taken to run while the 200 items had the largest variation in run time. Because these runs were done in parallel across different cores this may have been the reason for some variation in run times. The parallel runs were done for the same instance size, *i.e.* 20 item instances in parallel and then 50 item instances in parallel and so on to ensure evenly distributed load across cores.

[4]Particle swarm optimisation (PSO) is based on swarming theory where all particles in a neighborhood depend on the discoveries and past experiences of their neighbours (Liu, *et al.* 2008).

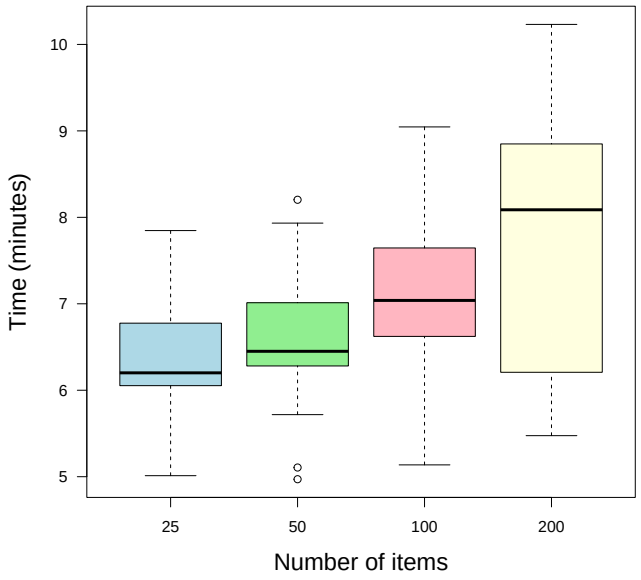
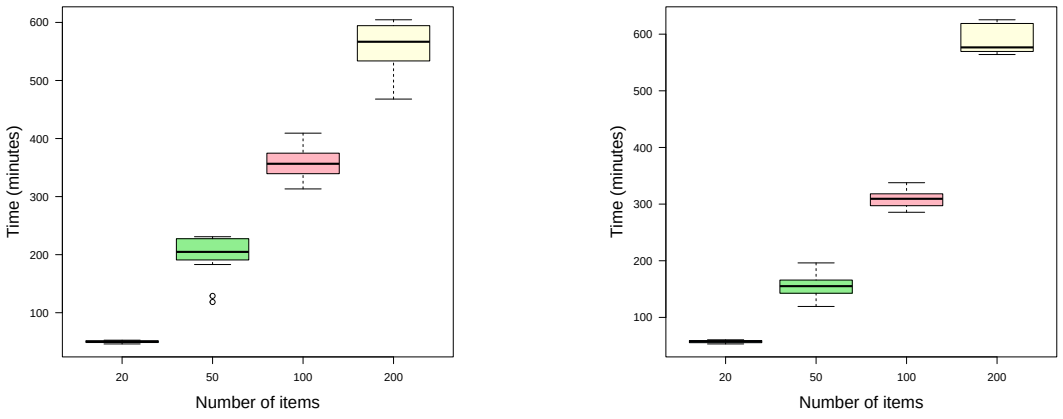


Figure 22: Time taken (in minutes) for the MOEA algorithm to run on 25, 50, 100 and 200 item instances in Class 0.



(a) Cost and Lateness Objectives

(b) Cost and Load Imbalance Objectives

Figure 23: Time taken (in minutes) to run the 2D MOEA algorithm for different item numbers, optimising cost and lateness objectives (left) and cost and load imbalance objectives (right).

From the boxplots, it can be seen that the MOEA algorithm had reasonable run times for the 1D case, with the 200-item instances completing optimisation in around 8-10 minutes. However, when moving to the 2D case, it was found that 200-item instances required nearly 10 hours to run. This large difference between the two problems occurs because of the added complexity that comes with the 2D case, which requires item dimensions and positions within a bin to be taken into account. The 2D problem requires the use of the constructive heuristic to ensure feasible packing configurations, while the 1D problem only requires a check to be done that the sum of the total weight (or volume) of the items is less than the capacity of the bin. This check done in the 1D case is much faster to perform. While running the constructive heuristic for one

bin does not take particularly long, applying the check to every solution in a population and to each bin within each solution, increases the overall computational time.

To improve the time and efficiency of running all the test instances, the instances were run in parallel with five instances optimised simultaneously across five cores. Within each instance, the optimisation method using the MOEA algorithm was not parallelized. In real-world scenarios, such as optimising different warehouses or shipments concurrently, this parallelisation strategy would be beneficial. An alternative parallelisation approach would be to perform parallelisation on a single instance at the constraint check step of the algorithm. This parallelisation would split the population of solutions between the cores which would perform the constraint check using the constructive heuristic for each solution in the population. If there are N solutions in a population and four cores for example, each core would calculate the constraint violation score for $N/4$ solutions. However, due to the iterative nature of the MOEA algorithm which would require the constraint violation scores to be regrouped at the end of each iteration, it might be found that the synchronization overhead associated with regrouping the results from each core may outweigh the benefit of parallelizing the constraint check.

Despite the MOEA algorithm taking a long time to run for large instances, exact methods are very computationally expensive for large scale problems and in some cases become intractable. This means that the MOEA algorithm is still a suitable optimisation technique for bin packing problems, particularly 1D problems.

6.4 Usefulness of the linear relaxation lower bound

The LR of the problem was solved to derive a lower bound for the cost objective and to be used as a base off which to get an initial solution population for the MOEA algorithm for certain test instances. The purpose of computing the lower bound is to evaluate how well a particular solution performs relative to the theoretical optimum, which in this case is given by the LR optimum.

For the 1D case the percentage deviation from the lower bound was at most 23.69% which indicates that the lowest cost objective was not too far off from the LR optimum found. In addition, because the LR solution will allow for a fraction of a bin to be used (and therefore a fraction of the cost), it is expected that there will be a slight deviation from the lower bound present when looking at the MOEA algorithm solution as this does not allow for a fraction of a bin in the solution and therefore it accounts for the full bin cost resulting in a higher cost objective than the LR solution.

For the 2D case, it can be noted that the deviation from the lower bound is much larger with many values above 50% and 100%. Despite some of these values being quite large, in certain cases, the lowest possible cost that is a feasible solution has actually been found and the deviation value is a result of having to account for the full cost of the bin where previously a fraction of the cost was accounted for in the LR. An example is for instance Comb1.7 which has 20 items and was optimised over cost and lateness objectives. A lowest cost of 100 is found by the MOEA algorithm while the LR had an optimal cost of 61.96 (see Table 11a). This results in a 61% deviation value however the cheapest bin does cost 100 and because a fraction of the bin cannot be used, this is the lowest feasible solution.

The LR is a 1D problem as it considers only the item area and not the shape of the items. This means that the LR allows for fractional use of bin space which cannot be replicated in a feasible solution by the MOEA algorithm. When these infeasible solutions are changed to be feasible

packing configurations, there is often more bin space that is not used and therefore a higher cost than for the LR. The 1D case does not have this spatial complexity and so there are not 'wasted' space areas that are seen in the 2D case. This means that it is the nature of the 2D problems that lead to larger deviations from the cost lower bound and it is likely not the case that these instances are simply obtaining worse final solutions than those in the 1D case.

In addition, some of the convergence plots (see Figure 8 and Figure 17) indicated that the use of the LR solution as a starting point for the MOEA algorithm did not lead to a large difference in terms of the final solution obtained or to greater diversity of the final solution set. Thus while the LR can give some indication of how good the MOEA algorithm solutions are, because of the deviation that occurs when solutions need to be made feasible, this lower bound is not as useful at face value and some further investigation needs to be done in order to understand if a particular solution is good or not.

6.5 Limitations and strengths of the developed MOEA algorithm

The MOEA algorithm shows promise for solving the MOGBPP but there are certain areas that need improvement. One limitation of the work is the lack of solution diversity in the final populations which may be due to premature convergence. Another limitation related to the computational testing is the population size and number of generations adopted. In most cases, convergence was seen and so the number of generations run is less of a concern; however, in some cases it may have been seen that running the algorithm for more generations would have led to further improvements in the objective function values. An alternative stopping criteria, such as returning the results after there has been no change in the objective values after a specified number of generations, could have been explored. The population size of 200 is potentially a bit small, particularly for instances involving 200 items or more. If more items are added, a larger population may be required to maintain solution diversity.

A further limitation of the proposed model is the number of instances that were able to be tested and the amount of variation of characteristics that could be tested, particularly for the 2D case. The computational time required for the 2D tests made it impractical to explore a wider variety of test instances. Additionally, for the 2D case, no variation in the number of compulsory items was considered, with all instances having 80% of items compulsory and 20% non-compulsory. It would have been of interest to see how the results differed if all items were compulsory, or all non-compulsory. A further limitation of the instances optimised here is that only one profit distribution was considered which was based on item size. More variation in profit would have allowed one to extract further results about how the profit along with proportion of compulsory items changes the item allocation.

An additional limitation of this study is that no performance metrics such as general distance or hypervolume measures were calculated for the results. Such measures help one to understand how close the results obtained are to the true underlying Pareto front. These measures assess the spread and density of the resulting Pareto fronts. As seen in the results many of the Pareto fronts obtained here had very few solutions in them. Calculating these performance metrics would therefore have likely resulted in very poor measures contributing little meaningful insight to the current results analysis. The current tabular and visual presentation of the results allow one to assess the performance of the algorithm and see where some of its limitations lie. Given this, and the time constraints of the project, adding these measures was out of the scope of this work. Future research could consider such measures if greater spread in the Pareto fronts is obtained.

Despite these limitations, the main objectives of the work have been achieved, developing a mathematical model and algorithm to solve the multi-objective GBPP. The mathematical model and algorithm developed in this work provide a good framework and base for solving the problems and one can change certain components of the MOEA from how it has been coded and set up. For example, substituting the mutation operator for a different operator would integrate into the current developed algorithm, provided it outputs the results in the expected format.

In terms of strengths, the algorithm successfully minimized the objectives, with solutions showing decreasing objective values across generations. In addition, it was able to effectively find feasible solutions, achieving constraint violation scores of zero within the first 50 to 100 generations in many instances. Although it was found that there was a limited diversity of solutions in the final generation for some cases, the MOEA algorithm was able to account for conflicting objectives and simultaneously optimise them. The limited diversity does not come from the way in which the algorithm evaluates and sorts solutions but rather there being a need for more diversity to be introduced, or to slow down the rate at which the solutions converge.

Depending on the problem size, the developed algorithm can be run on real-world cases and is able to give an indication of some of the available solution options for a decision-maker without requiring explicit preference weightings for the objectives. While testing more diverse scenarios would have led to potentially interesting insights, time constraints made this infeasible. The scope of the project was to develop a mathematical model and algorithm and conduct initial testing which has been accomplished. The identified limitations could be further explored in future work.

7 Conclusion

This section provides a summary of the work presented, highlighting the objectives that were achieved and the contributions that have been made. In addition, future research directions and recommendations are given.

7.1 Summary of work

This work has explored the multi-objective GBPP, specifically considering the 1D and 2D variants of the problem. The objectives that have been considered are cost and maximum lateness as well as cost and load imbalance. The specific aims and objectives of the work outlined in Section 1 have been achieved.

A review of the relevant literature was done in Section 2, highlighting the current literature on cutting and packing problems and multi-objective optimisation techniques. A mathematical model and MOEA algorithm were developed to solve the 1D and 2D MOGBPP. This algorithm was implemented in R and a number of runs done on various test instances. The performance of the algorithm was assessed both in terms of computational efficiency and in terms of solution quality. The discussion in Section 6 highlighted interesting features of the results and specific strengths and limitations of the developed algorithm.

The developed algorithm was able to minimize the selected objectives and in many cases present a set of solutions from which a trade-off could be made. It was found that the algorithm was better at doing this when optimising over cost and lateness as opposed to cost and load imbalance and this was because the cost and load imbalance objectives may not have been conflicting. While some Pareto front sets of solutions were found, the discussion emphasized the need for an increase in diversity in the set of solutions, particularly for test instances with a larger number of items. In terms of the computational results it was found that the developed algorithm had reasonable run times for solving the 1D variant of the problem but took much longer to run when moving to the 2D variant particularly for instances with 100 and 200 items. Some of these limitations identified could be studied further in future work as discussed in the following section.

7.2 Recommendations for future work

As pointed out in Section 6.5, there exist a few areas where future work could be beneficial to improve the current MOEA algorithm. These areas, along with a few other suggestions for future research, are outlined below.

Use of variety of test instances: It is recommended that the MOEA algorithm be run on a wider variety of test instances, varying the different components of the problem to see how this effects the optimisation results. This would help highlight what types of problems work best with the current developed MOEA algorithm.

Investigate hyperparameters in the MOEA algorithm: Hyperparameters of the MOEA algorithm should be further investigated to find what parameter values work best for the multi-objective optimisation. This includes parameters such as mutation probability, the size of the population and the number of generations run. It is hypothesized that a larger solution population and a larger mutation probability would lead to greater diversity in the set of solutions found, however this might require the algorithm to be run for more generations to minimize the objectives function values as much as possible. For example, instance Comb17.8 which has 200

items was rerun with a population size of 400, double the previously run population size and the mutation probability was increased from 0.3 to 0.8. Figure 24 shows the objective function values of the population across generations and Figure 25 gives the mean cost (left) and mean lateness (right) across the generations. An increase in the diversity of the population can be seen in Figure 24 and it was found that the final generation contained eight non-dominated solutions which varied across the cost and lateness objectives. This is four times more solutions than were present when the MOEA algorithm was run with 200 generations and a mutation probability of 0.3. The lowest cost solution had a cost of 2505.56 where previously a lowest cost of 2005.56 was obtained, so there is more diversity but it has not reached as low a cost. This means it should be run for more generations. Also, the convergence plots in Figure 25 indicate that the algorithm has not yet converged. Running for more generations would likely decrease the objectives further, however, it might also then be found that the diversity decreases. This relationship with the hyperparameters of the algorithm would therefore be interesting to research further.

It may be interesting to implement a similar MOEA algorithm to the one developed but to change the mutation probability as the generations continue, starting out with a larger mutation probability that encourages diversity and decreasing this as more generations are run to try and encourage convergence of the algorithm. Additionally, one could investigate implementing something similar with how much crossover is performed between parent solutions.

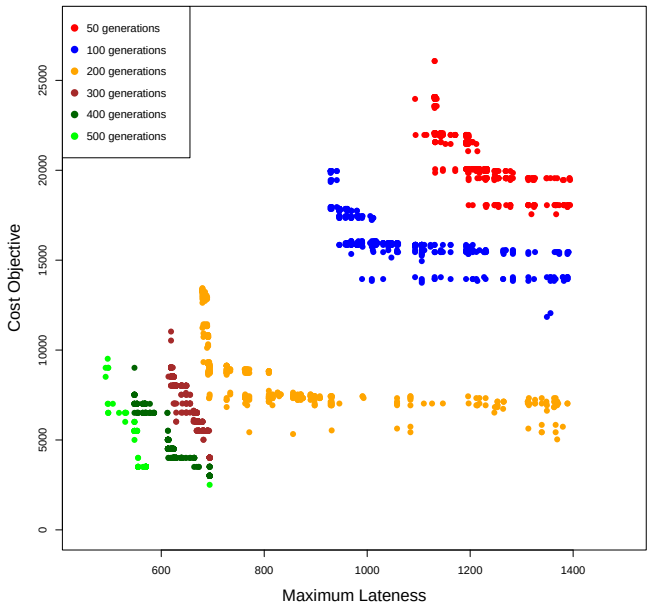


Figure 24: Evolution of objective function values across generation for instance Comb17.8, while minimizing cost and lateness. Here the population size is set to 400 and the mutation probability is set to 0.8.

Explore different solution selection techniques: As highlighted in the results, the lack of solution diversity was one of the major limitations of the developed algorithm. In addition to considering how changing the population size and mutation probability effect the results diversity, further research could also consider other ways to increase diversity such as introducing new sub-populations of solutions at different points in the algorithm. However, it must be noted that increasing the diversity brought into the solution population needs to be balanced with

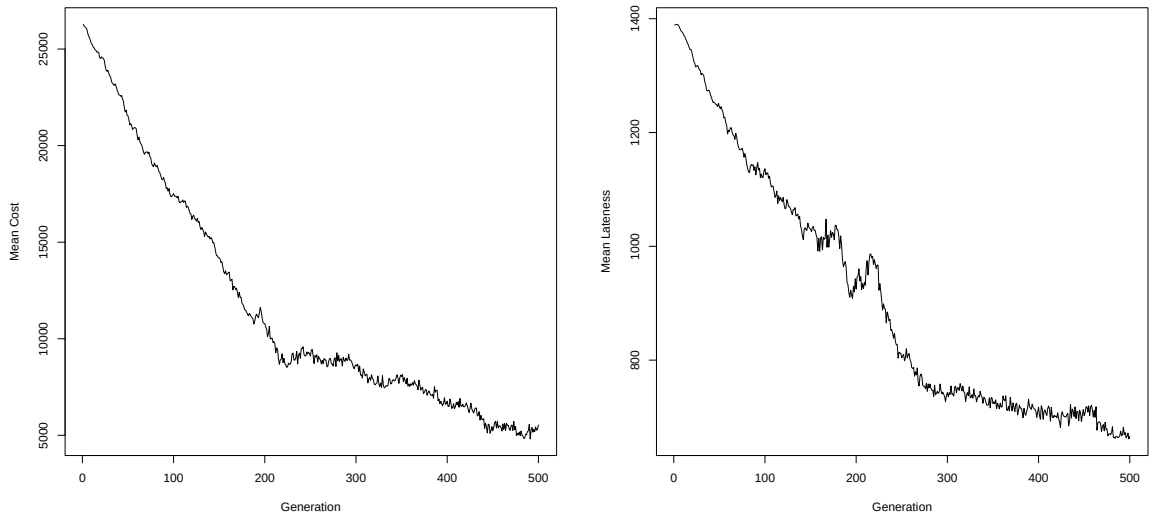


Figure 25: Mean cost (left) and mean lateness (right) over 500 generations for instance Comb17.8, with population size of 400 and mutation probability of 0.8.

finding solutions that minimize the objectives as much as possible and converge to a stable set of solutions.

The developed algorithm compares solutions with one another by considering their constraint violation value and objective function values. First, their constraint violation scores are compared, one solution is preferred over another if it has a lower constraint violation score. If both have a score of zero or if the solutions have the exact same violation score, the objectives are then considered and the lower objective value solution favoured. A potential problem with this selection technique is that one solution may be much better in terms of objectives but be slightly constraint violating, with a score of say 50, while another may be feasible but a really poor solution in terms of objective values, in such a case the poor solution is favoured. While it is good that the algorithm ensures solutions are feasible, this comparison could be an area of future research to investigate what happens if at first solutions with constraints under a certain value are allowable focusing on comparing objective values and then later there is more strictness in terms of constraint violation scores. A balance must be found between selecting good solutions and feasible solutions.

Improve the packing heuristic: A final area for future work would be to improve the speed of the algorithm, and this would be best improved if a more efficient packing heuristic method was found. As previously discussed, the constraint check involving the constructive heuristic slowed down the algorithm, particularly as the number of items increased. Finding a faster packing heuristic method would speed up this check and the overall algorithm. In addition, a packing heuristic method that does not necessarily favour the left side, as most heuristics currently do, may also improve the optimisation of the load imbalance objective.

7.3 Contributions to the field

This work makes the following contributions to the field of multi-objective optimisation specifically of the MOGBPP.

Contribution 1: *A review of the relevant existing work on the MOGBPP and various extensions of the problem.*

One of the main aims of this work was to develop a mathematical model and MOEA algorithm to solve the MOGBPP. In order to do this a review of relevant literature including existing work and methods for solving multi-objective optimisation problems was done in Section 2. This review helps one to better understand the MOGBPP problem and the methods that have been put forward for solving it.

Contribution 2: *Design of a tailored MOEA algorithm for the MOGBPP with implementation in R.*

The second contribution of this work is a mathematical model and developed MOEA algorithm to solve the 1D and 2D MOGBPP problem. This developed algorithm has been implemented in such a way that it can be used as a foundation by others and parts of the MOEA algorithm easily adapted to the users specific problem. For example, one could change the selection or mutation step and still use the overall algorithm as these smaller functions are easily integrated into the overall algorithm. While the focus of this work is the MOGBPP, the implemented algorithm could also be adapted to other multi-objective optimization problems by changing the constraint violation calculations and objective evaluation functions.

Contribution 3: *Generation of test instances for the 2D GBPP.*

To the best knowledge of the author there were no previous 2D GBPP test instances available in the literature and existing repositories. A further contribution of this work is thus the generation of 2D test instances, with variation in the number of items and item shapes. The 180 instances generated can be used to test different optimisation algorithms and can serve as a standard set of instances for the 2D GBPP.

Contribution 4: *Introduction of linear relaxation model as a lower bound.*

The use of the linear relaxation model and the LR result as a lower bound for the MOEA algorithm is a further contribution of this work. While there were in certain cases large deviations from this lower bound, the use of the LR result allows one to understand more about how a solution performs. This is particularly beneficial as the generated test instances do not have known solutions and the LR model offers a relatively simple technique to obtain a lower bound.

Contribution 5: *Highlights on limitations of the MOEA algorithm and challenges faced when considering multi-objective optimisation.*

A further contribution of the work is that it has highlighted limitations of MOEA algorithms for solving the MOGBPP and has shown the challenges that can be faced when performing multi-objective optimisation. These limitations and challenges include early convergence and limited solution diversity in the final iterations of the algorithm. By highlighting the limitations, one can better understand the complexities of performing multi-objective optimisation.

Contribution 6: *Identification of future research areas.*

Linked to Contribution 5, this work has identified areas where improvements can be made and where future work could be done. Section 7.2 highlighted four specific future research areas. These areas would help improve the developed MOEA algorithm and allow for more robust testing of the algorithm, making the algorithm more applicable to real-world problems.

8 References

[1] Ali S, Ramos AG, Carravilla MA & Oliveria JF, 2022, *On-line three-dimensional packing problems: a review of off-line and on-line solution approaches*, Computers and Industrial Engineering, **168(1)**, pp. 108-122.

[2] Arbib C & Marinelli F, 2017, *Maximum lateness minimization in one-dimensional bin packing*, Omega, **68**, pp. 76-84.

[3] Arbib C, Marinelli F & Pizzuti A, 2021, *Number of bins and maximum lateness minimization in two-dimensional bin-packing*, European Journal of Operational Research, **291(1)**, pp. 101-113.

[4] Baker BS, Coffman Jr EG & Rivest RL, 1980, *Orthogonal packings in two dimensions*, SIAM Journal on Computing, **9(4)**, pp. 846-855.

[5] Baldi MM & Bruglieri M, 2017, *On the generalized bin packing problem*, International Transactions in Operations Research, **24**, pp. 425-438.

[6] Baldi MM, Crainic TG, Perboli G & Tadei R, 2012, *The generalized bin packing problem*, Transportation Research Part E, **48(6)**, pp. 1205-1220.

[7] Baldi MM, Manerba D, Perboli G & Tadei R, 2019, *A generalized bin packing problem for parcel delivery in last-mile logistics*, European Journal of Operational Research, **274(3)**, pp. 990-999.

[8] Bennell JA, Lee LS & Potts CN, 2013, *A genetic algorithm for two-dimensional bin packing with due dates*, International Journal of Production Economics, **145(2)**, pp. 547-560.

[9] Burke EK, Kendall G & Whitwell G, 2004, *A new placement heuristic for the orthogonal stock-cutting problem*, Operations Research, **52(4)**, pp. 655-671.

[10] Crainic TG, Fomeni FD & Rei W, 2021, *Multi-period bin packing model and effective constructive heuristics for corridor-based logistics capacity planning*, Computers & Operations Research, **132**, 105308.

[11] Dahmani N, Clautiaux F, Krichen S & Talbi EG, 2014, *Self-adaptive metaheuristics for solving a multi-objective 2-dimensional vector packing problem*, Applied Soft Computing, **16**, pp. 124-136.

[12] De Armas J, Leon C, Miranda G & Segura C, 2010, *Optimization of a multi-objective two-dimensional strip packing problem based on evolutionary algorithms*, International Journal of Production Research, **48(7)**, pp. 2011-2028.

[13] Deb K, Pratap A, Agarwal S & Meyarivan T, 2002, *A fast and elitist multiobjective genetic algorithm: NSGA-II*, IEEE Transactions on Evolutionary Computation, **6(2)**, pp. 182-197.

[14] Deb K, 2008, *Introduction to evolutionary multiobjective optimization*, pp. 59-96 in Branke J, Deb K, Miettinen K & Słowiński R (Eds.), *Multi-objective optimization: lecture notes in computer science*, **5252**, Springer, Berlin, Heidelberg.

[15] Dorigo M, 1992, *Optimization, learning and natural algorithms*, PhD Thesis, Politecnico di Milano, Milan.

[16] El Yaagoubi A, Charhbili M, Boukachour J & El Hilali Alaoui A, 2022, *Multi-objective optimization of the 3d container stowage planning problem in a barge convoy system*, Computers and Operations Research, **114**, 105796.

[17] Elhossini A, Areibi S & Dony R, 2010, *Strength Pareto particle swarm optimization and hybrid EA-PSO for multi-objective optimization*, Evolutionary Computation, **18(1)**, pp. 127-156.

[18] Erbayrak S, Özkir V & Yildirim UM, 2021, *Multi-objective 3d bin packing problem with load balance and product family concerns*, Computers and Industrial Engineering, **159**, 107518.

[19] Fernandez A, Gil C, Banos R & Montoya MG, 2013, *A parallel multi-objective algorithm for two-dimensional bin packing with rotations and load balancing*, Expert Systems with Applications, **40(13)**, pp. 5169-5180.

[20] Geiger MJ, 2008, *Bin packing under multiple objectives - a heuristic approximation approach*, Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization, Matsushima, pp. 53-56.

[21] Gomez JC & Terashima-Marin H, 2012, *Building general hyper-heuristics for multi-objective cutting stock problems*, Computacion y Sistemas, **16(3)**, pp. 321-334.

[22] Gomez JC & Terashima-Marin H, 2018, *Evolutionary hyper-heuristics for tackling bi-objective 2d bin packing problems*, Genetic Programming and Evolvable Machines, **19**, pp. 151-181.

[23] Gradisar D & Glavan M, 2020, *Material requirements planning using variable-sized bin-packing problem formulation with due date and grouping constraints*, Processes, **8(10)**, 1246.

[24] Guerreiro A, Fonseca C & Paquete L, 2021, *The hypervolume indicator: computational problems and algorithms*, Association for Computing Machinery Computing Surveys, **54(6)**, pp. 1-42.

[25] Holland JH, 1975, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, University of Michigan Press, Ann Arbor.

[26] Huang W, Zhang Y & Li L, 2019, *Survey on multi-objective evolutionary algorithms*, Journal of Physics Conference series, **1288(1)**, 012057.

[27] Jakobs S, 1996, *On genetic algorithms for the packing of polygons*, European Journal of Operational Research, **88(1)**, pp. 165-181.

[28] Kaabi J, Harrath Y, Bououdina H, & Qasim A, 2018, *Toward smart logistics: A new algorithm for a multi-objective 3d bin packing problem*, Proceedings of Smart Cities Symposium, Bahrain, pp. 1-5.

[29] Kennedy J & Eberhart R, 1995, *Particle swarm optimization*, Proceedings of ICNN'95 - International Conference on Neural Networks, **4**, pp. 1942-1948.

[30] Khanafer A, Clautiaux F, Hanafi, S & Talbi EG, 2012, *The min-conflict packing problem*, Computers and Operations Research, **39(9)**, pp. 2122-2132.

[31] Lara A, Sanchez G, Coello C & Schutze O, 2009, *Hcs: A new local search strategy for memetic multiobjective evolutionary algorithms*, IEEE Transactions on Evolutionary Computation,

14(1), pp. 112-132.

[32] Leung S, Wong WK & Mok PY, 2008, *Multi-objective genetic optimization of the spatial design for packing and distribution carton boxes*, Computers and Industrial Engineering, **54(4)**, pp. 889-902.

[33] Leung S & Zhang D, 2011, *A fast layer-based heuristic for non-guillotine strip packing*, Expert Systems with Applications, **38(10)**, pp. 13032-13042.

[34] Leung S, Zhang D & Sim KM, 2011, *A two-stage intelligent search algorithm for the two-dimensional strip packing problem*, European Journal of Operational Research, **215(1)**, pp. 57-69.

[35] Liu DS, Tan KC, Huang SY, Goh CK & Ho WK, 2008, *On solving multiobjective bin packing problems using evolutionary particle swarm optimization*, European Journal of Operational Research, **190**, pp. 357-382.

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

[37] Monaci M, 2002, *Algorithms for packing and scheduling problems*, PhD Thesis, Universita di Bologna, Bologna.

[38] Naderi B & Yazdani M, 2014, *A real multi-objective bin packing problem: a case study of an engine assembly line*, Arabian Journal for Science and Engineering, **39**, pp. 5271-5277.

[39] Placek M, 2023, *Logistics industry worldwide - statistics & facts* [Online][Cited February 13th 2024], Available from: https://www.statista.com/topics/5691/logistics-industry-worldwide.

[40] Rajappa G, 2012, *Solving combinatorial optimization problems using genetic algorithms and ant colony optimization*, PhD Thesis, University of Tennessee, Knoxville.

[41] Rakotonirainy RG, 2018, *Metaheuristic solution of the two-dimensional strip packing problem*, PhD Thesis, University of Stellenbosch, Stellenbosch.

[42] Sathe M, Schenk O & Burkhart H, 2009, *Solving bi-objective many-constraint bin packing problems in automobile sheet metal forming processes*, Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization, Nantes, pp. 246-260.

[43] Silva E, Oliveira JF & Wäscher G, 2014, *2DCPackGen: a problem generator for two-dimensional rectangular cutting and packing problems*, European Journal of Operational Research, **237(3)**, pp. 846-856.

[44] Tiwari S & Chakraborti N, 2006, *Multi-objective optimization of a two-dimensional cutting problem using genetic algorithms*, Journal of Materials Processing Technology, **173**, pp. 384–393.

[45] Trivella A & Pisinger D, 2016, *The load-balanced multi-dimensional bin-packing problem.* Computers and Operations Research, **74**, pp. 152-164.

[46] Wang S, Wang L, Shi R & Ge M, 2010, *Study on improved ant colony optimization for bin-packing problem,* Proceedings of the International Conference on Computer Design and Applications, Qinhuangdao, pp. 489-491.

[47] Wäscher G, Haubner H & Schumann H, 2007, *An improved typology of cutting and packing problems*, European Journal of Operational Research, **183(3)**, pp. 1109-1130.

[48] Zhang D, Kang Y & Deng A, 2006, *A new heuristic recursive algorithm for the strip rectangular packing problem*, Computers and Operations Research, **33(8)**, pp. 2209-2217.

[49] Zheng W, Ren P, Ge P, Qiu Y & Liu Z, 2012, *Hybrid heuristic algorithm for two-dimensional steel coil cutting problem*, Computers and Industrial Engineering, **62(3)**, pp. 829-838.

[50] Zhou A, Qu BY, Li H, Zhao SZ, Suganthan PN & Zhang Q, 2011, *Multiobjective evolutionary algorithms: A survey of the state of the art*, Swarm and evolutionary computation, **1(1)**, pp. 32-49.

[51] Zitzler E, 1999, *Evolutionary algorithms for multiobjective optimization: methods and applications*, PhD Thesis, Swiss Federal Institute of Technology Zurich, Zurich.

[52] Zitzler E, Deb K & Thiele L, 2000, *Comparison of multiobjective evolutionary algorithms: empirical results*, Evolutionary Computation, **8(2)**, pp. 117-132.

[53] Zitzler E & Kunzli S, 2004, *Indicator-based selection in multiobjective search*, Proceedings of the 8th International Conference on Parallel Problem Solving from Nature, Birmingham, pp. 832-842.

[54] Zitzler E, Laumanns M & Thiele L, 2001, *SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization*, TIK Report, **103**, no page.

9 Appendix

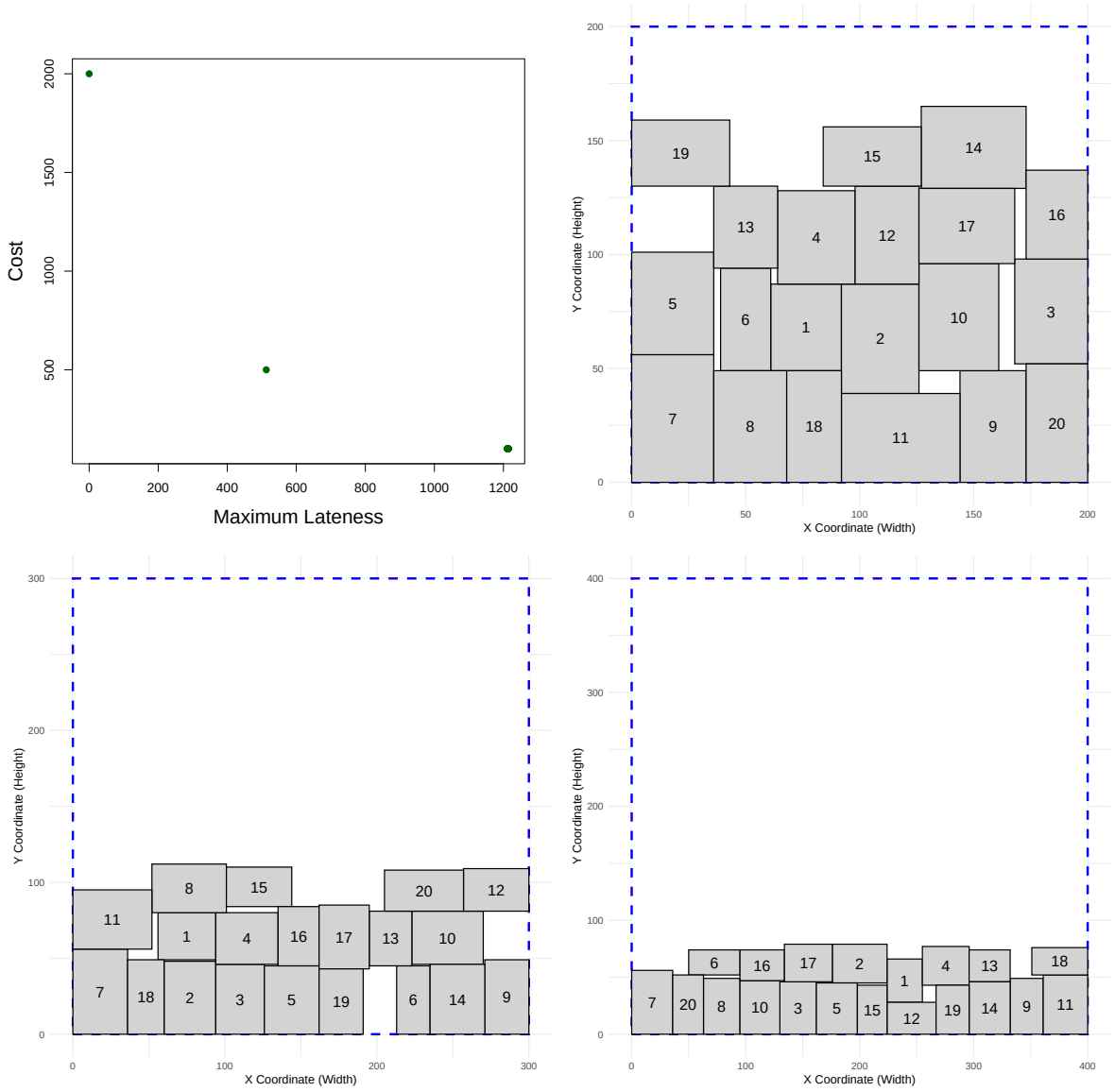


Figure A: Instance Comb2.3 solutions and packing configurations. The top left plot shows the set of solutions found in the final generation of the MOEA. The remaining three plots show the packing layout of each of the three solutions.

Table A: Optimisation Results for Class 0 Problem 2 Instances. Problem 2A results are shown in the left table (a) and Problem 2B results are given in the right table (b). The linear relaxation bound on cost is given in brackets (LB). The % difference is calculated as $\frac{\text{Cost}-\text{Lower Bound}}{\text{Lower Bound}} \times 100$. The 'unique' column refers to the number of distinct solutions found in the final generation in terms of objective function values.

(a) Problem 2A Results

Items	Inst	Cost (LB)	Lateness	% Difference	Unique
25	1	1576.46 (1357.89)	575.77	16.16	2
	2	1585.15 (1370.02)	423.00	15.69	1
	3	1697.16 (1481.91)	481.43	14.47	1
	4	1796.91 (1576.44)	632.00	14.00	5
	5	1995.04 (1707.36)	737.00	17.00	3
	6	1656.83 (1482.92)	475.57	11.68	1
	7	1969.16 (1746.69)	505.76	12.74	1
	8	2003.74 (1743.65)	584.76	14.88	3
	9	1624.60 (1445.73)	453.57	12.39	2
	10	1641.67 (1506.23)	262.00	9.03	1
50	1	3587.72 (3027.93)	756.67	18.52	1
	2	3846.55 (3298.20)	706.00	16.60	1
	3	3485.74 (2960.82)	785.52	17.74	2
	4	3308.98 (2970.94)	848.00	11.42	1
	5	3845.77 (3228.57)	698.00	19.09	1
	6	3428.58 (2965.88)	601.41	15.61	1
	7	3276.92 (2843.13)	717.29	15.23	2
	8	3298.39 (2944.62)	569.67	12.00	1
	9	3599.88 (3044.15)	587.67	18.23	1
	10	3219.67 (2753.40)	659.81	16.87	1
100	1	7541.80 (6152.39)	710.10	22.61	1
	2	6905.18 (5816.97)	611.61	18.68	2
	3	7732.29 (6300.24)	777.81	22.80	1
	4	7713.43 (6523.55)	806.75	18.20	1
	5	7320.33 (6075.56)	718.67	20.46	2
	6	7181.18 (6174.42)	756.26	16.36	1
	7	7349.54 (6169.35)	823.00	19.17	1
	8	6907.73 (5911.63)	764.17	16.82	1
	9	7330.64 (6184.55)	869.30	18.54	1
	10	7241.27 (6093.80)	776.33	19.01	1
200	1	14371.94 (12223.06)	840.56	17.63	1
	2	15155.61 (12429.39)	882.88	21.88	1
	3	15001.15 (12398.99)	861.07	21.09	1
	4	14233.53 (11875.74)	920.96	19.85	2
	5	13474.41 (11312.05)	803.86	19.07	1
	6	14435.04 (11839.26)	889.22	21.94	1
	7	13679.44 (11552.56)	899.19	18.39	1
	8	15143.15 (12422.30)	956.42	21.86	1
	9	14569.93 (12421.28)	881.26	17.25	1
	10	14321.78 (12070.31)	973.07	18.59	1

(b) Problem 2B Results

Items	Inst	Cost (LB)	Lateness	% Difference	Unique
25	1	1683.77 (1493.14)	1183.11	12.79	2
	2	1803.07 (1585.50)	983.23	13.73	1
	3	1744.19 (1542.21)	948.05	13.12	4
	4	1697.27 (1546.25)	592.05	9.77	1
	5	1657.74 (1489.11)	1334.67	11.37	1
	6	1558.33 (1382.59)	951.27	12.70	3
	7	1791.30 (1592.57)	518.00	12.46	2
	8	1926.81 (1717.17)	681.57	12.26	1
	9	1772.71 (1570.36)	515.33	12.87	2
	10	1686.58 (1522.09)	115.67	10.81	1
50	1	3736.79 (3226.52)	700.00	15.84	1
	2	3494.20 (3072.69)	1551.41	13.65	4
	3	3208.64 (2890.99)	852.53	11.00	1
	4	3358.44 (2974.36)	1420.08	12.94	1
	5	3514.47 (3085.82)	1161.82	13.91	2
	6	3492.10 (3085.82)	1036.37	13.19	2
	7	3406.90 (3006.64)	927.80	13.37	2
	8	3332.97 (2958.27)	1042.67	12.70	3
	9	3264.58 (2893.82)	1190.17	12.83	2
	10	3616.92 (3168.15)	1109.39	14.21	2
100	1	6755.66 (5862.51)	1192.62	15.27	1
	2	7503.42 (6351.69)	1099.85	18.09	2
	3	7483.83 (6349.67)	961.71	17.68	1
	4	7049.96 (6100.59)	1084.76	15.61	1
	5	6900.05 (6043.10)	914.33	14.21	1
	6	7497.13 (6403.01)	1446.58	17.09	1
	7	7478.13 (6504.58)	1354.63	14.95	1
	8	6904.64 (6075.39)	1456.00	13.66	1
	9	6750.25 (5924.68)	1176.78	13.98	1
	10	6884.50 (5999.19)	1192.18	14.79	1
200	1	13985.70 (11854.67)	1727.94	18.08	1
	2	14793.94 (12286.78)	1200.26	20.41	1
	3	14120.22 (12116.89)	1342.63	16.61	1
	4	13689.23 (11770.08)	1398.51	16.62	2
	5	14487.86 (12142.07)	1187.15	19.34	1
	6	14183.53 (12089.67)	1385.00	17.29	1
	7	13850.79 (11875.75)	1528.51	16.58	1
	8	14652.82 (12302.71)	1432.03	19.11	1
	9	13758.10 (11891.91)	1672.00	16.00	1
	10	13453.28 (11498.81)	1318.70	17.03	1

Table B: Optimisation Results for Class 0 Problem 3 Instances. Problem 3A results are given in the left table (a) and Problem 3B results are shown in the right table (b). The linear relaxation bound on cost is given in brackets (LB). The % difference is calculated as $\frac{\text{Cost}-\text{Lower Bound}}{\text{Lower Bound}} \times 100$. The 'unique' column refers to the number of distinct solutions found in the final generation in terms of objective function values.

(a) Problem 3A Results

Items	Inst	Cost (LB)	Lateness	% Difference	Unique
25	1	2334.15 (1949.07)	840	19.75	3
	2	2168.39 (1939.95)	883	11.76	4
	3	2132.14 (1899.82)	534.89	12.24	1
	4	2407.25 (2060.75)	787	16.77	1
	5	2270.11 (1980.13)	880.32	14.68	5
	6	2330.45 (2039.72)	645	14.26	2
	7	2222.08 (1846.12)	502.56	20.37	1
	8	2103.97 (1783.13)	565	18.04	2
	9	2225.77 (1970.00)	661.63	13.00	5
	10	2255.72 (1903.88)	502.89	18.55	1
50	1	4504.79 (3978.71)	849	13.21	2
	2	4607.67 (3752.10)	772.46	22.83	1
	3	4465.50 (3912.41)	712.79	14.16	1
	4	4442.03 (3782.13)	833	17.42	2
	5	4331.64 (3838.70)	662.53	12.86	1
	6	4196.06 (3537.57)	575	18.60	1
	7	4483.68 (3806.45)	883.16	17.78	2
	8	4371.74 (3680.60)	927.89	18.75	2
	9	4398.39 (3737.15)	667	17.71	1
	10	4391.86 (3714.01)	669	18.28	1
100	1	9087.11 (7571.63)	714.33	19.97	1
	2	8901.83 (7417.59)	837.21	19.93	1
	3	9065.69 (7365.56)	703.92	22.81	1
	4	9198.65 (7542.81)	746.22	21.96	1
	5	9142.79 (7789.23)	877.05	17.37	3
	6	9114.54 (7656.31)	832.95	19.04	1
	7	9284.09 (7506.33)	828.27	23.69	1
	8	9214.12 (7555.42)	781.33	21.93	1
	9	9051.94 (7553.39)	830.67	19.74	1
	10	9101.55 (7578.72)	847.67	20.52	1
200	1	18234.64 (15227.12)	855.54	19.75	1
	2	18373.97 (15432.43)	876.28	19.06	1
	3	18258.13 (14810.13)	878.80	23.33	1
	4	18339.66 (15180.95)	927.33	20.94	1
	5	18482.64 (14986.38)	882.84	23.31	1
	6	18708.97 (15173.85)	936.67	23.34	1
	7	18299.95 (14938.40)	916.86	22.55	1
	8	18473.73 (15092.61)	929.40	22.39	1
	9	18663.41 (15203.06)	890.74	22.86	1
	10	18502.54 (15198.17)	871	21.91	1

(b) Problem 3B Results

Items	Inst	Cost (LB)	Lateness	% Difference	Unique
25	1	2094.92 (1840.80)	955.33	13.83	3
	2	2188.16 (2033.68)	1350.06	7.59	2
	3	2133.56 (1916.26)	1326.39	11.35	3
	4	2188.01 (2027.64)	995.00	7.91	1
	5	2184.04 (2015.54)	1189.30	8.36	3
	6	2070.77 (1872.99)	1064.71	10.57	3
	7	2120.97 (1968.33)	939.50	7.78	4
	8	1967.25 (1828.70)	1179.00	7.59	2
	9	2134.18 (2008.49)	1064.00	6.27	2
	10	2030.23 (1790.72)	847.90	13.38	2
50	1	4270.08 (3779.15)	1711.26	13.02	2
	2	4528.25 (3835.56)	890.84	17.98	1
	3	4372.79 (3660.60)	1109.00	19.47	1
	4	4320.01 (3647.48)	1181.67	18.42	4
	5	4302.08 (3774.16)	1034.16	14.02	2
	6	4175.23 (3727.05)	1136.18	11.99	3
	7	4360.37 (3855.74)	1408.81	13.15	6
	8	4164.85 (3743.05)	1450.07	11.29	5
	9	4393.64 (3838.58)	1359.05	14.44	2
	10	4430.64 (3969.20)	1612.69	11.63	2
100	1	9079.00 (7704.38)	1127.80	17.86	1
	2	9486.56 (7968.65)	1387.27	18.99	5
	3	9070.25 (7512.52)	1285.48	20.82	1
	4	8898.35 (7731.43)	1133.94	14.99	1
	5	8889.79 (7508.48)	1706.68	18.46	2
	6	8800.77 (7709.43)	1164.20	14.14	1
	7	8915.36 (7659.16)	1522.77	16.32	2
	8	8764.05 (7582.79)	1137.40	15.58	1
	9	9238.15 (7790.87)	1369.12	18.46	1
	10	9014.66 (7856.00)	1784.46	14.75	1
200	1	18013.26 (15298.44)	1451.42	17.68	1
	2	18290.15 (15375.61)	1234.14	18.99	1
	3	18053.72 (15085.21)	1648.20	19.74	1
	4	18033.48 (15068.52)	1306.55	19.75	1
	5	18278.03 (15044.28)	1311.25	21.47	1
	6	17860.38 (15120.49)	1442.83	18.26	1
	7	18173.16 (14990.87)	1603.32	21.28	1
	8	18407.76 (15265.28)	1665.87	20.64	1
	9	18471.77 (15342.69)	1826.18	20.42	1
	10	17811.65 (14948.78)	1517.08	9.19	2

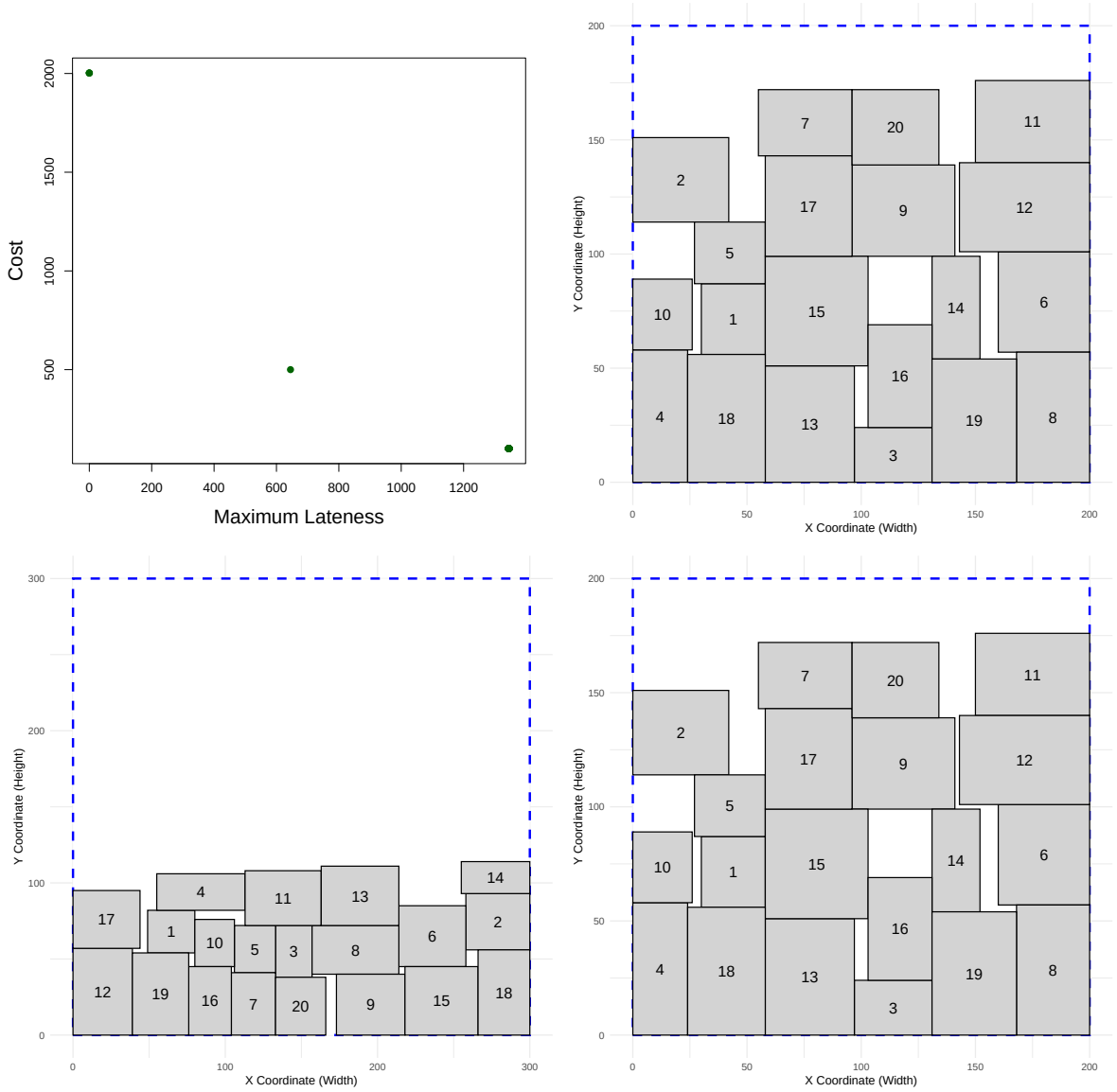


Figure B: Instance Comb3.1 solutions and packing configurations. The top left plot shows the set of solutions found in the final generation of the MOEA. The remaining three plots show the packing layout of each of the three solutions.

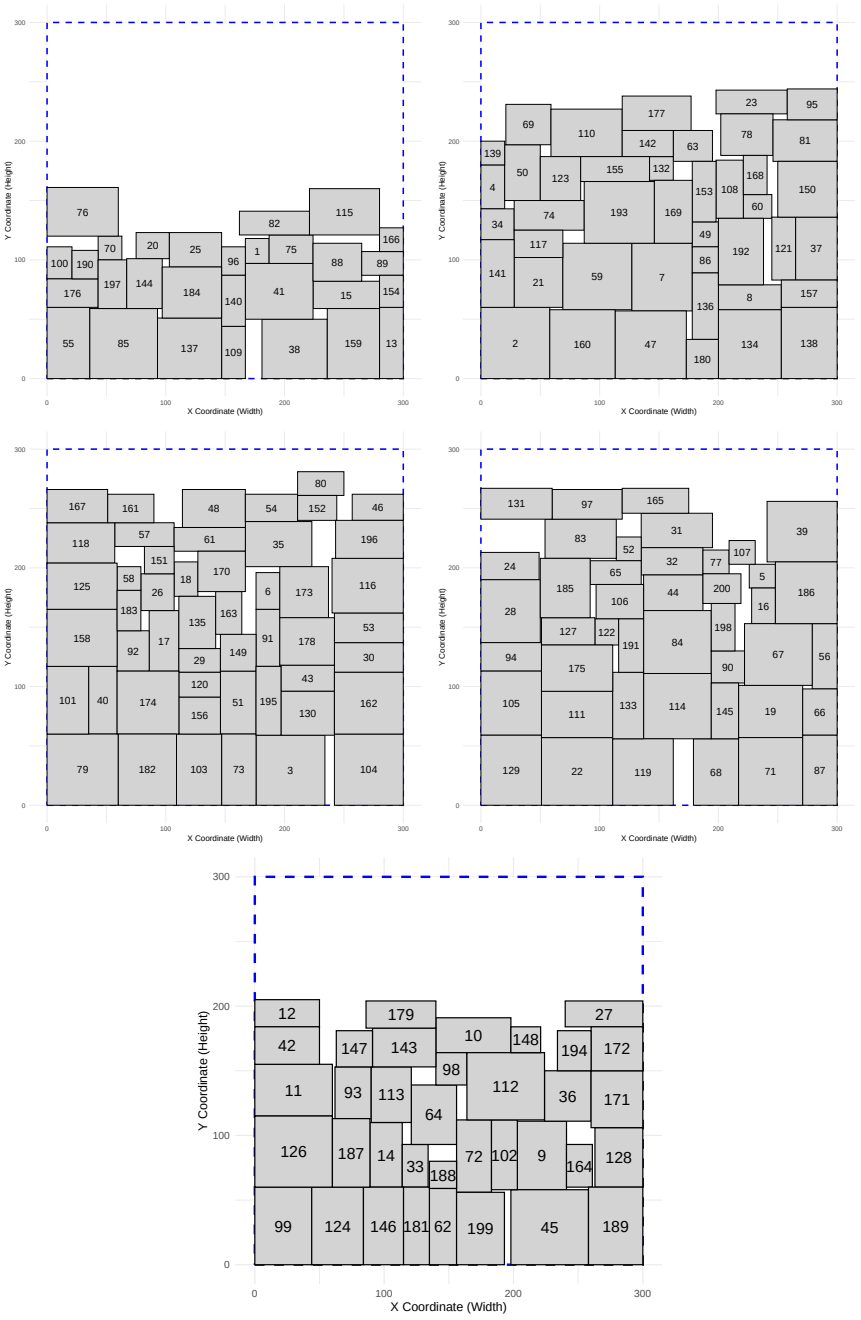


Figure C: Instance Comb17.10, final generation packing solution option 1.

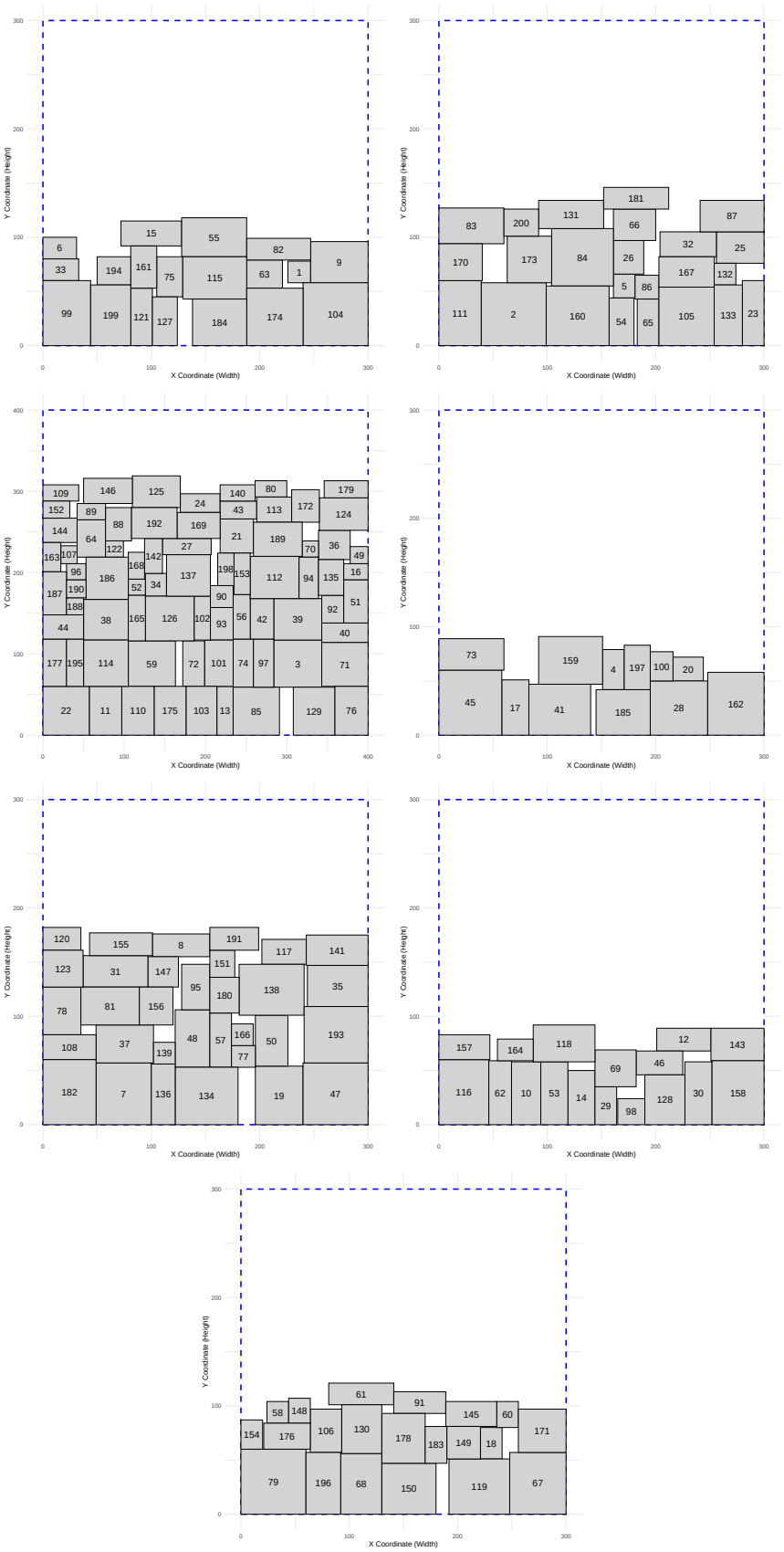


Figure D: Instance Comb17.10, final generation packing solution option 2.

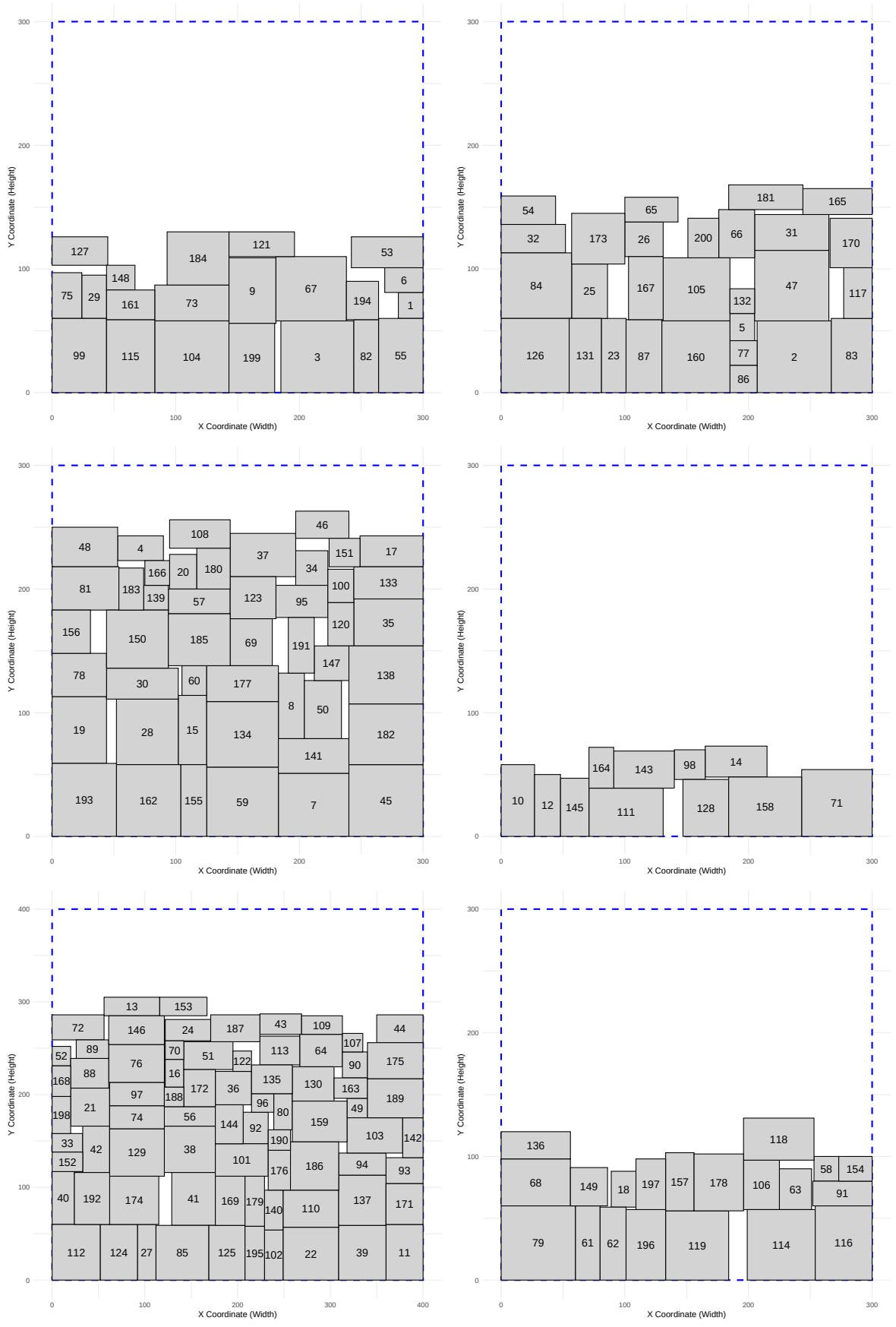


Figure E: Instance Comb17.10, final generation packing solution option 3.

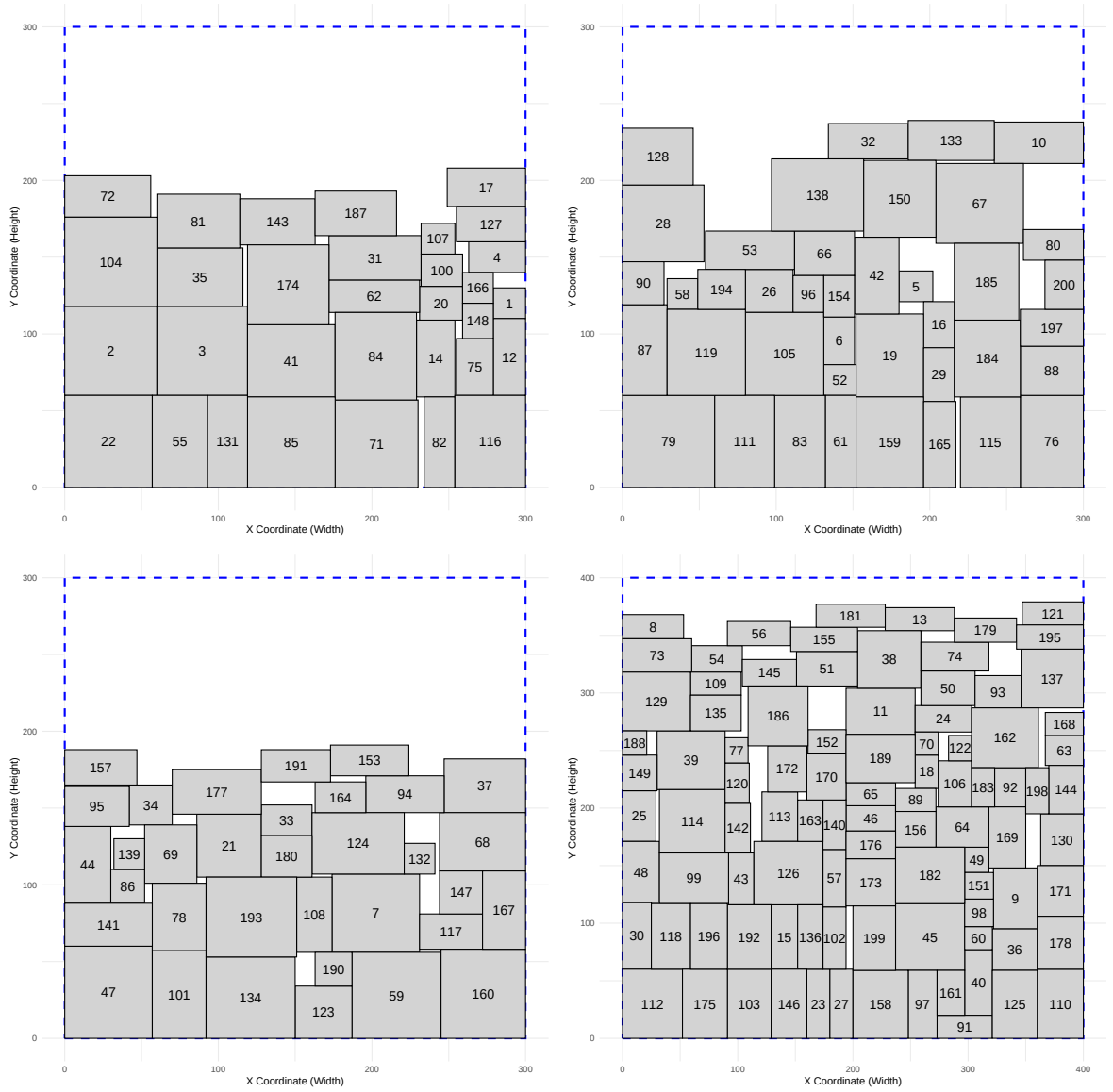


Figure F: Instance Comb17.10, final generation packing solution option 4.

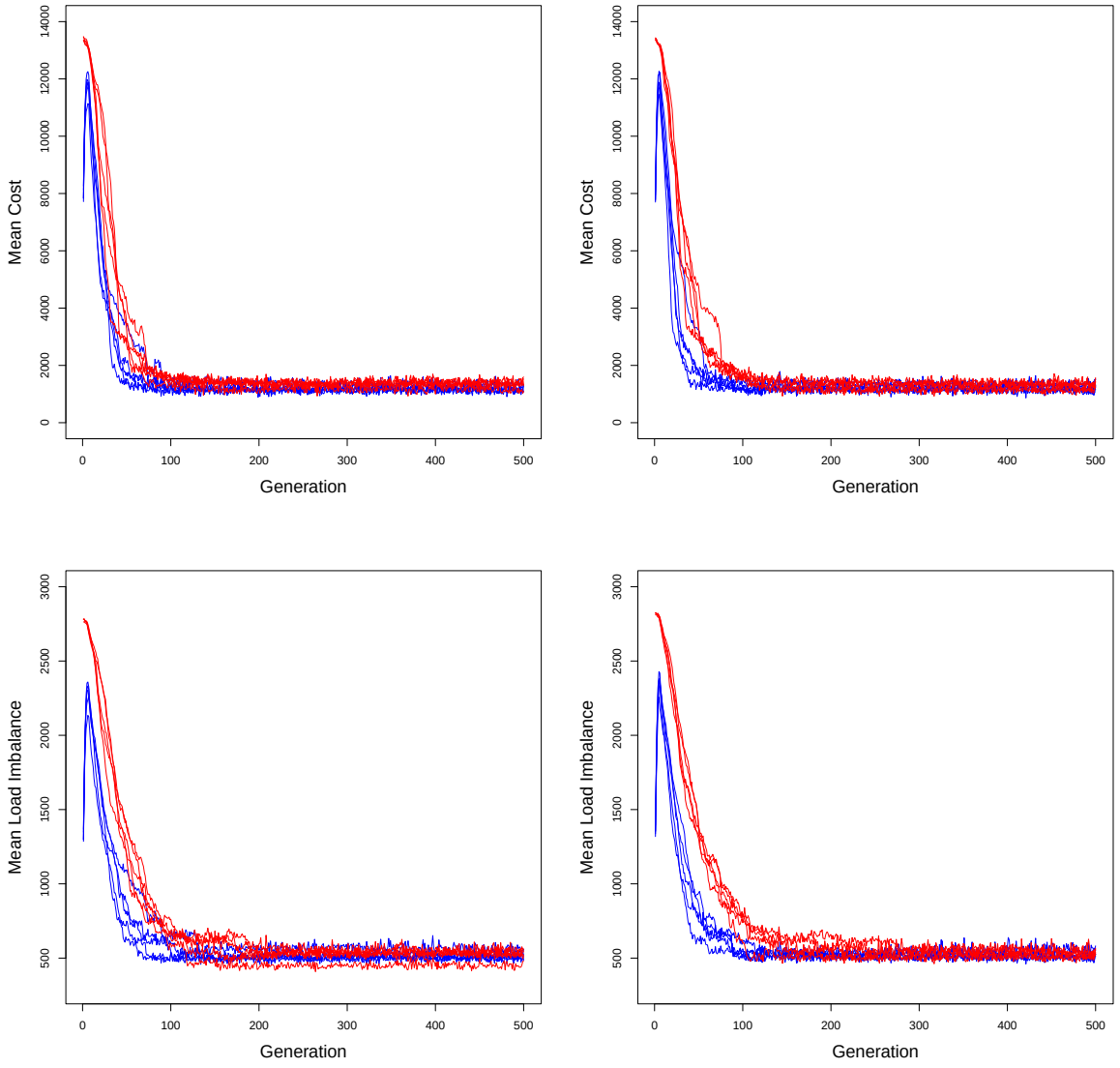


Figure G: Convergence plots for 100-item instances, mixture item shapes, Comb13 (left) and long and narrow items, Comb14 (right) for cost (top) and load imbalance (bottom).

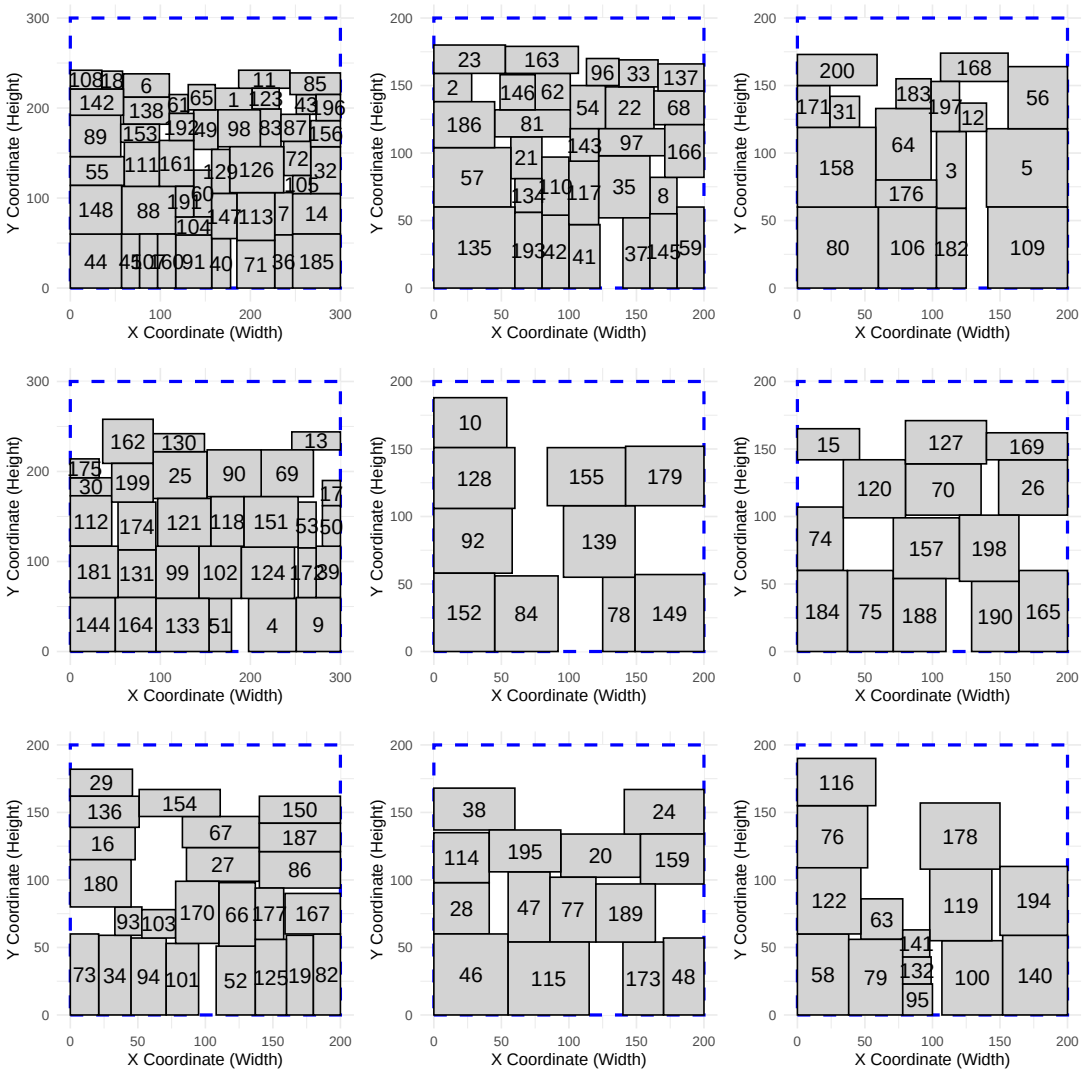


Figure H: One packing configuration for instance Comb18.1, minimizing cost and load imbalance.