Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN

INGEGNERIA BIOMEDICA, ELETTRICA E DEI SISTEMI

Ciclo 34

Settore Concorsuale: 09/G1 - AUTOMATICA

Settore Scientifico Disciplinare: ING-INF/04 - AUTOMATICA

THE THREE-DIMENSIONAL SINGLE-BIN-SIZE BIN PACKING PROBLEM: COMBINING METAHEURISTIC AND MACHINE LEARNING APPROACHES

Presentata da: Gabriele Ancora

Coordinatore Dottorato

Michele Monaci

Supervisore

Claudio Melchiorri

Co-supervisore

Gianluca Palli

Esame finale anno 2022

Declaration of Authorship

I, Gabriele ANCORA, declare that this thesis titled, “The Three-Dimensional Single-Bin-Size Bin Packing Problem: Combining Metaheuristic and Machine Learning Approaches” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

In the beginning was the Word
And by mutation came the Gene
Word
Wore
Gore
Gone
Gene

Michael A. Arbib

ALMA MATER STUDIORUM - UNIVERSITY OF BOLOGNA

Abstract

School of Engineering and Architecture
Department of Electrical, Electronic and Information Engineering (DEI)
"Guglielmo Marconi"

Doctor of Philosophy

The Three-Dimensional Single-Bin-Size Bin Packing Problem: Combining Metaheuristic and Machine Learning Approaches

by Gabriele ANCORA

The Three-Dimensional Single-Bin-Size Bin Packing Problem is one of the most studied problem in the Cutting & Packing category. From a strictly mathematical point of view, it consists of packing a finite set of strongly heterogeneous “small” boxes, called items, into a finite set of identical “large” rectangles, called bins, minimizing the unused volume and requiring that the items are packed without overlapping. The great interest is mainly due to the number of real-world applications in which it arises, such as pallet and container loading, cutting objects out of a piece of material and packaging design. Depending on these real-world applications, more objective functions and more practical constraints could be needed. After a brief discussion about the real-world applications of the problem and a exhaustive literature review, the design of a two-stage algorithm to solve the aforementioned problem is presented. The algorithm must be able to provide the spatial coordinates of the placed boxes vertices and also the optimal boxes input sequence, while guaranteeing geometric, stability, fragility constraints and a reduced computational time. Due to NP-hard complexity of this type of combinatorial problems, a fusion of metaheuristic and machine learning techniques is adopted. In particular, a hybrid genetic algorithm coupled with a feedforward neural network is used. In the first stage, a rich dataset is created starting from a set of real input instances provided by an industrial company and the feedforward neural network is trained on it. After its training, given a new input instance, the hybrid genetic algorithm is able to run using the neural network output as input parameter vector, providing as output the optimal solution. The effectiveness of the proposed works is confirmed via several experimental tests.

Acknowledgements

My sincere thanks go to my supervisor Claudio Melchiorri and my co-supervisor Gianluca Palli for the encouragement and guidance they have given to me over the last three years. Through their support and trust, I have had the opportunity to direct my part in an interesting and very challenging project, and to enjoy myself along the way.

I want to deeply thank my family for their support in this path as a PhD student. I could not feel more lucky for the encouragement you gave me.

I would like to thank Chiara, for her unconditional love and constant support, always being there for me.

Finally, a great hug goes to friends and all the people who have shared any kind of moment with me.

Thank you!

Gabriele

Contents

Declaration of Authorship **i**

Abstract **iii**

Acknowledgements **v**

1 Introduction **1**
1.1 Overview 1
1.2 Structure of Thesis 2

2 Literature review **3**
2.1 Introduction 3
2.2 Cutting and Packing problems 3
2.3 Single-Bin-Size Bin Packing Problem 6
2.3.1 One-Dimensional variant 6
2.3.1.1 Heuristics 6
2.3.1.2 Metaheuristics 11
2.3.2 Two-Dimensional variant 12
2.3.2.1 Heuristics 13
2.3.2.2 Metaheuristics 17
2.3.3 Three-Dimensional variant 19
2.3.3.1 Real-world constraints 19
2.3.3.2 Heuristic/metaheuristics methods 24

3 Preliminary mathematical tools **26**
3.1 Feedforward Neural Networks 26
3.2 Genetic Algorithms 28

4 A Two-Stage Algorithm for the Three-Dimensional Single-Bin-Size Bin Packing Problem **30**
4.1 Hybrid Genetic Algorithm setup 30
4.1.1 Chromosome structure 30
4.1.2 Initial population 31

4.1.3 Evaluation process 31
4.1.3.1 A new heuristic procedure 32
4.1.3.2 Feasibility check 32
4.1.3.3 Fitness function 32
4.1.4 Genetic Operators 33
4.1.4.1 Selection 33
4.1.4.2 Crossover and mutation 33
4.2 A new two-stage algorithm 34
4.2.1 First stage 34
4.2.1.1 Dataset creation 34
4.2.1.2 Training process 36
4.2.2 Second stage 37
4.3 Results 37
4.3.1 HGA parameters 37
4.3.2 FNN parameters 38
4.3.3 Input instances and graphical results 39
4.3.3.1 Training and validation processes 39
4.3.3.2 Test process 40

5 Conclusion and future works **44**
5.0.1 Conclusion 44
5.0.2 Future works 44

Bibliography **46**

List of Figures

2.1 Illustration of a semi-loaded pallet. 5
2.2 Basic Cutting and Packing problem types. 5
2.3 Intermediate types for the Bin Packing Problem. 6
2.4 Solution using First-Fit Algorithm. 7
2.5 Solution using First-Fit-Decreasing Algorithm. 8
2.6 Solution using Max-Rest Algorithm. 9
2.7 Solution using Next-Fit Algorithm. 9
2.8 Solution using Next-Fit-Decreasing Algorithm. 10
2.9 Solution using Best Fit Algorithm. 11
2.10 Solution using Best-Fit-Decreasing Algorithm. 11
2.11 Three classical strategies for the strip packing. 14
2.12 Algorithm HFF. 14
2.13 Variation of HFF algorithm. 15
2.14 FNF algorithm. 16
2.15 FFF algorithm. 16
2.16 FBL algorithm. 17
2.17 NBL algorithm. 17
2.18 Six possible orientations. 21
2.19 Percentage of papers in which constraint types have been addressed (population size = 163). 23

3.1 Structure of an artificial neuron. 27
3.2 FNN with three hidden layers. 27

4.1 Frequency distribution of the different dimension box typologies of all packing instances. 39
4.2 MSE evaluation on training set and validation set. 40
4.3 Best solutions found related to some relevant test instances. . 43

List of Tables

4.1 Local weight vectors. 38
4.2 Target vs. predicted local weight vector. 41
4.3 Target vs. predicted global weight vector. 41
4.4 Simulation time (S), number of different dimension box typologies (M) and total number of boxes (N) reported for some test instances. 42
4.5 Time simulation statistics on subset $\mathcal{D}_{\text{test}}$. 42

Dedicated to my parents

Chapter 1

Introduction

1.1 Overview

One of the most extensive and researched fields within combinatorial optimization is the one concerning Cutting & Packing (in short C & P) problems. This because of their wide range of applications, especially in industries, transportation, warehousing and supply chain management. The cutting problems arise, for example, with the production of steel bars and plates, paper, adhesive tape, glass, pipes and textile. In all these cases, it is usually more economic to produce large objects in only a few standard sizes at first and later cut them into the sizes requested by the customers, than produce the required sizes directly. The packing problems could arise, instead, in the manufacturing and distribution industries, where all kind of goods are packaged in cartons for easy handling and then packed into a bin for transportation and warehouse storage. Within this family, a key-role is played by Bin Packing Problem (BPP), studied since the thirties [37]. BPP involves placing a set of items of different sizes on bins of various sizes with the goal to pack as many items as possible in the least number of bins. It arises in forms of one, two or three dimensions and, consequently, the related industrial applications vary in according to them. In the one–dimensional (1D | BPP) form, also denoted strip packing, only one dimension of the items to be packed is considered. The remaining ones are either immaterial or (more usually) fixed. A practical application for the 1D | BPP is cutting lengths of stock material that has fixed width, such as pipes for plumbing applications, or metal beams. A set of orders for different lengths must be fulfilled by cutting the stock lengths into smaller pieces while minimising the wasted material. Regarding the two–dimensional packing problem (2D|BPP), 2D shapes are packed on 2D bins. The shapes can either be rectangular or irregular. Typical applications could be glass cutting, sheet metal cutting and cardboard cutting. The three–dimensional problem (3D|BPP), instead, involves packing 3D regular or irregular shapes into a 2D or 3D bins. Typical applications include pallet loading and loading of cargo containers inside the ship. The BPP is well treated in the literature using different mathematical approaches. They are classified in two main clusters: exact and heuristic/metaheuristic. Due to NP-hardness of the problem [46], exact optimization techniques [12], [36] can not be used in real-world applications because, in general, they require a huge amount of boxes, and consequently high computational burden. In fact, some industries require that the code be run on embedded systems with very limited memory and sometimes even without online memory allocation. Also heuristic techniques [11], out of a metaheuristic framework, are not deemed as suitable choices, because they are often too greedy, and usually get trapped in a local optimum very far from the global one. For these purposes, the aim of this research project is to develop and implement an algorithm based on a metaheuristic framework coupled with a machine learning approach, with the aim to produce high quality solutions for a special class of the 3D|BPP, called Three-Dimensional Single-Bin-Size Bin Packing Problem (3D|SBS|BPP), having the property that all the bins are identical. Moreover, real-world practical constraints must be also satisfied and the optimal boxes input sequence is required as output.

1.2 Structure of Thesis

The thesis is organized as follows. In Chapter 2, a literature review and a exhaustive classification about C & P problems is presented, focusing on the BPP and its dimensional variants. In Chapter 3, a summary of genetic algorithm and feedforward neural networks is shown, discussing how they can be adapted and combined to solve the 3D|BPP. In Chapter 4, the complete structure of the proposed algorithm is presented and simulation results are showed and discussed. Finally, in Chapter 5 some final considerations and plans for future activities are described.

Chapter 2

Literature review

2.1 Introduction

A literature review concerning the C & P problems is now presented. In particular, in Section 2.2, the complete family of C & P problems is described, relying on a useful categorization developed in [64], extending the one previously depicted in [22], with the aim to unify even more notations and definitions and concentrate further research on special types of problems. After that, in Section 2.3 the Single-Bin-Size Bin Packing problem is addressed and different methodologies found in literature are described for the one, two and three dimensional forms in 2.3.1, 2.3.2, 2.3.3, respectively.

2.2 Cutting and Packing problems

C & P problems are among the most well-studied combinatorial optimization problems in literature. This is due to their versatility, allowing many real-world applications to be modelled as such. Using the tipology defined by [64], they can be defined as follows:

Given are two sets of elements, namely a set of large objects (input, supply) and a set of small items (output, demand), which are defined exhaustively in one, two, three or more geometric dimensions. Select some or all small items, group them into one or more subsets and assign each of the resulting subset to one of the large objects such that the geometric conditions hold, i.e., the small items of each subset have to be laid out on the corresponding large object such that

- *all small items of the subset lie entirely within the large object,*
- *the small items do not overlap,*

and a given (single-dimensional or multi-dimensional) objective function is optimized.

In according to the aforementioned improved topology, a generic C & P problem can be classified using five criteria:

- **Kind of assignment:**
  - **Input minimization**: The large objects set is sufficient to guarantee that all small items can be assigned to a subset of it. It is therefore not necessary to make a selection among the small items;
  - **Output maximization**: The large objects set is not sufficient to accomodate the entire set of small items. Then, a selection among the small items is mandatory.
- **Assortment of small items:**
  - Identical small items;
  - Weakly heterogeneous assortment of small items;
  - Strongly heterogeneous assorment of small items.
- **Assortment of large objects:**
  - Identical large items;
  - Weakly heterogeneous assortment of large items;
  - Strongly heterogeneous assorment of large items.
- **Problem dimensionality:**
  - One-dimensional;
  - Two-dimensional;
  - Three-dimensional.
- **Shape of small items**:
  - Regular small items (rectangles, cuboids, circles, cylinders);
  - Irregular items.

The use of the first two criteria allows us to define the so called *basic problem types*. If also the third criterion is used, it is possible to define the *intermediate problem types*. Using all five criteria, the *refined problem types* are obtained. For further details, see [64]. In Figure 2.2, the six basic problem types are represented while in Figure 2.3 the intermediate problem types for the Bin Packing problem are shown. Considering all the aforementioned criteria, the bin packing problem addressed in this research can be identified as a Three-Dimensional Rectangular Single-Bin-Size Bin Packing Problem (3D|SBS|BPP). Moreover, large objects are assumed to be industrial pallets while small objects are assumed to be rectangular parallelepiped (boxes) to be palletized. Notice that pallets do not have solid walls. This means that there is less lateral support for the boxes packed then what occurs when large objects are assumed to be containers. An illustration of a semi-loaded pallet is shown in Figure 2.1.

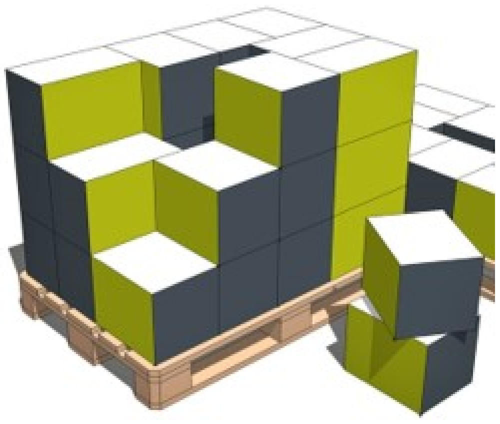


FIGURE 2.1: Illustration of a semi-loaded pallet.

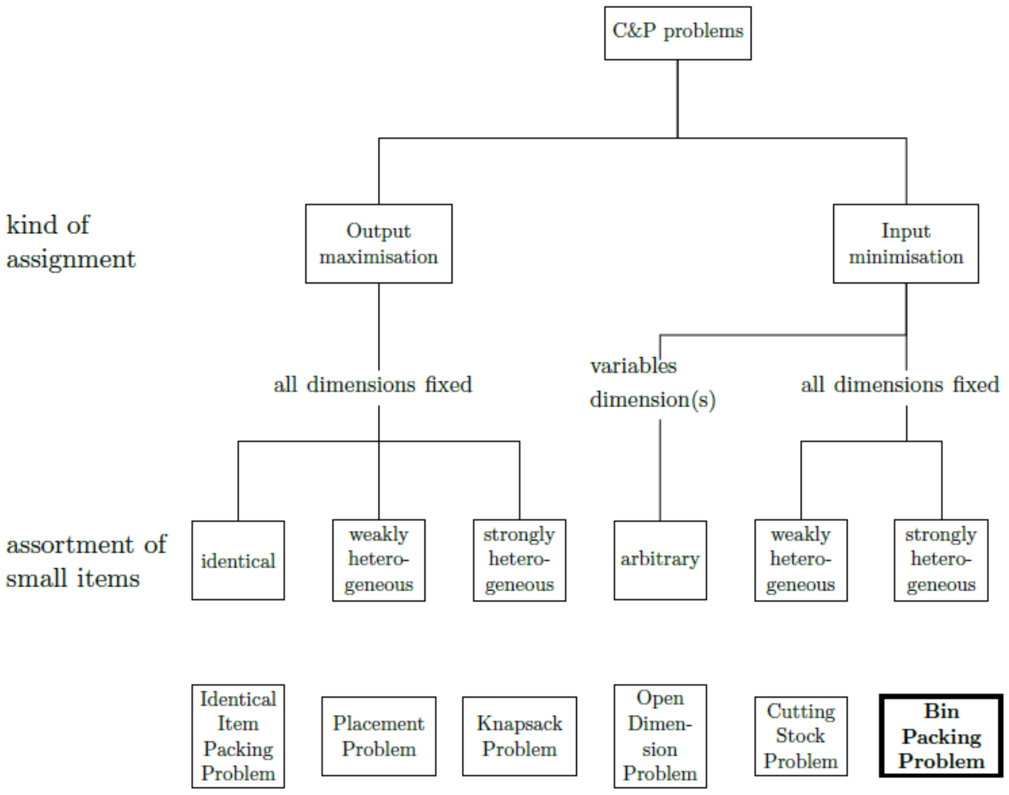


FIGURE 2.2: Basic Cutting and Packing problem types.

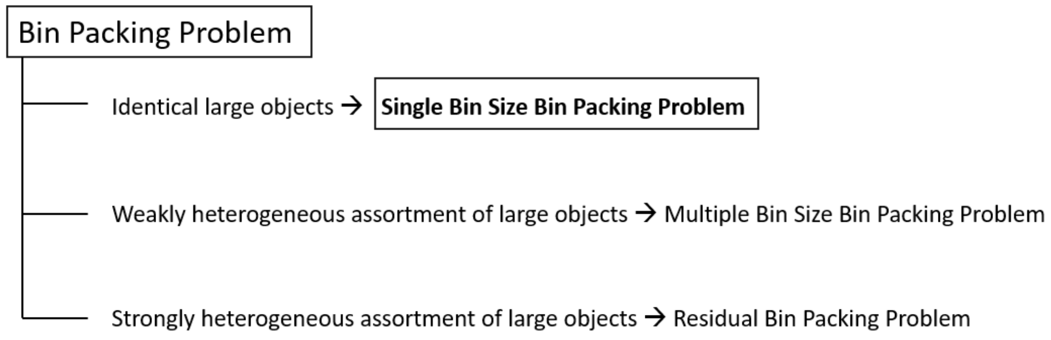

Bin Packing Problem
Identical large objects → Single Bin Size Bin Packing Problem
Weakly heterogeneous assortment of large objects → Multiple Bin Size Bin Packing Problem
Strongly heterogeneous assortment of large objects → Residual Bin Packing Problem

FIGURE 2.3: Intermediate types for the Bin Packing Problem.

2.3 Single-Bin-Size Bin Packing Problem

Single-Bin-Size Bin Packing Problem (SBS|BPP), arising in forms of one, two or three dimensions, involves placing a set of strongly heterogeneous items in identical bins with the goal to pack as many items as possible in the least number of bins. Many algorithms can be find in literature to solve the SBS | BPP for all of its dimensional variants. Basically, it is possible to divide them into two main clusters: exact and heuristic/metaheuristic algorithms. Due to both NP-hardness and real-world point of view of the problem, only algorithms belonging to the second cluster will be analyzed. However, references of some exact techniques are also provided. Regarding the three-dimensional variant of the problem, several real-world industrial constraints are also analyzed and discussed.

2.3.1 One-Dimensional variant

One-Dimensional Single-Bin-Size Bin Packing Problem (1D|SBS|BPP) is the easiest variant of the SBS|BPP. In this version, items have only a single dimension (cost, time, size, weight, etc.). The most relevant heuristic and metaheuristic techniques adopted to face the problem are now described. Regarding exact methods, instead, refer to [16].

2.3.1.1 Heuristics

The heuristics now described are on-line algorithms. This means that the items are given to the algorithm one by one from a list, and the next items is given as soon as the current items is irrevocably placed. In the off-line version, instead, it is assumed that the algorithm has full knowledge of the

whole input.
In order to make clear the descriptions of the following heuristic algorithms, an example set of items having a weight sequence $S = \{4, 8, 5, 1, 7, 6, 1, 4, 2, 2\}$ and a bin capacity equals to 10 is given.

- **First-Fit Algorithm**: *"Place the items in the order in which they arrive. Place the next item into the lowest numbered bin in which it fits. If it does not fit into any open bin, start a new bin"* [53]. The pseudocode of the algorithm is shown in Algorithm 1. In the worst-case, a new bin has to be opened each time a new object is inserted. Thus, there are 1, 2, 3, ..., $n-1$ executions of the inner loop, which yields an asymptotical factor of $\mathcal{O}(n^2)$. The solution related to the example is depicted in Figure 2.4.

Algorithm 1 First-Fit

```
1: for All items $i$=1,2,...,$n$ do
2:     for All bins $j$=1,2,... do
3:         if Item $i$ fits in bin $j$ then
4:             Pack item $i$ in bin $j$
5:             Break the loop and pack the next item
6:         end if
7:     end for
8:     if Item $i$ did not fit in any available bin then
9:         Create a new bin and pack item $i$
10:    end if
11: end for
```

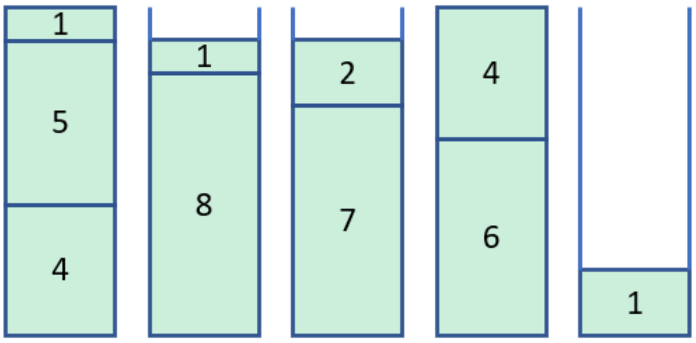


FIGURE 2.4: Solution using First-Fit Algorithm.

- **First-Fit-Decreasing Algorithm:** The pseudocode of the algorithm is shown in Algorithm 2. Since Counting Sort has a complexity of $\mathcal{O}(n+k)$, where k is the largest weight, the algorithm is obviously dominated by the running time of First-Fit, which yields a factor of $\mathcal{O}(n^2)$. The solution related to the example is depicted in Figure 2.5.

Algorithm 2 First-Fit-Decreasing

```
1: Sort objects in decreasing order using Counting Sort
2: Apply First-Fit to the sorted list of objects
```

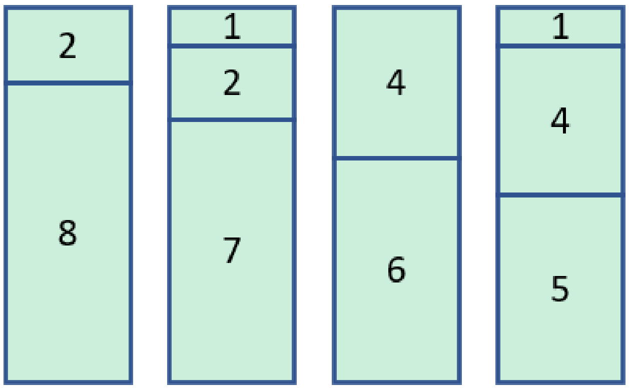


FIGURE 2.5: Solution using First-Fit-Decreasing Algorithm.

- **Max-Rest Algorithm:** The pseudocode of the algorithm is shown in Algorithm 3. If a naive algorithm is used, determining the bin with maximum remaining capacity yields a factor of $O(n)$. Thus, the worst-case running-time of the algorithm is $O(n^2)$. A more detailed analysis shows that the bin can be determined by using a priority queue (i.e. a heap). In this case, the algorithm has a worst-case running-time of $\mathcal{O}(n \log n)$. The solution related to the example is depicted in Figure 2.6.

Algorithm 3 Max-Rest

```
1: for All items $i$=1,2,...,$n$ do
2:     Determine $k = \min\left\{i|c_i = \min_{j=1}^{j=m} c_j\right\}$, the index of the bin with maximum remaining capacity.
3:     if Item $i$ fits in bin $k$ then
4:         Pack item $i$ in bin $k$
5:     else
6:         Create a new bin and pack item $i$
7:     end if
8: end for
```

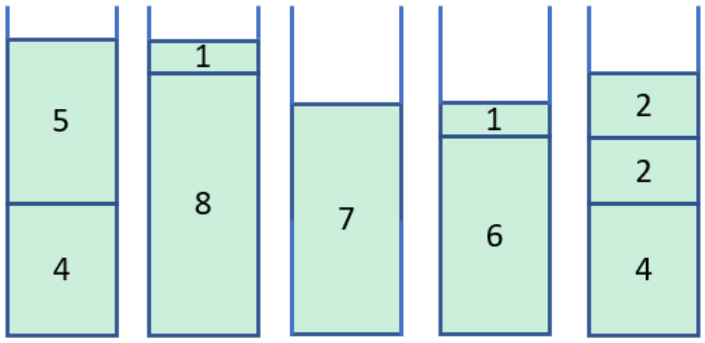


FIGURE 2.6: Solution using Max-Rest Algorithm.

- **Next-Fit Algorithm:** *"Place the items in the order in which they arrive. Place the next item into the current bin if it fits. If it does not, close that bin and start a new bin"* [53]. The pseudocode of the algorithm is shown in Algorithm 4. Since packing an object can be done in constant time, the algorithm is dominated by the loop, which has a running-time of $\Theta(n)$. The solution related to the example is depicted in Figure 2.7.

Algorithm 4 Next-Fit

```
1: for All items $i$=1,2,...,$n$ do
2:     if Item $i$ fits in current bin then
3:         Pack item $i$ in current bin
4:     else
5:         Create a new bin, flag it as current, and pack item $i$
6:     end if
7: end for
```

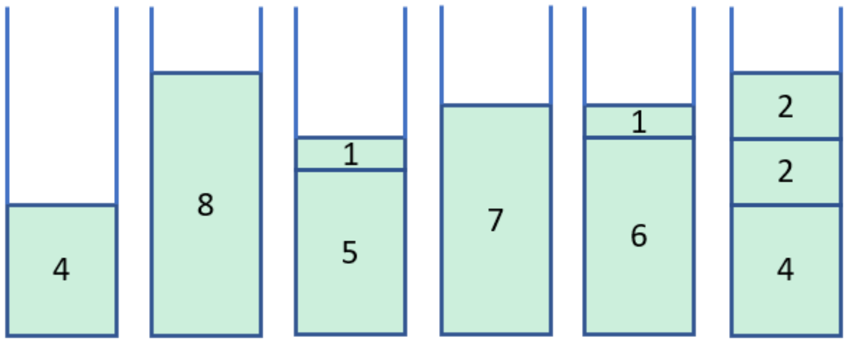


FIGURE 2.7: Solution using Next-Fit Algorithm.

- **Next-Fit-Decreasing Algorithm:** The pseudocode of the algorithm is shown in Algorithm 5. Since Next-Fit has a running time of $\Theta(n)$, the dominating factor is the Counting Sort algorithm, which has a running time of $\mathcal{O}(n+k)$, where k is the maximum weight of the problem. The solution related to the example is depicted in Figure 2.8.

Algorithm 5 Next-Fit-Decreasing

```
1: Sort objects in decreasing order using Counting Sort
2: Apply Next-Fit to the sorted list of objects
```

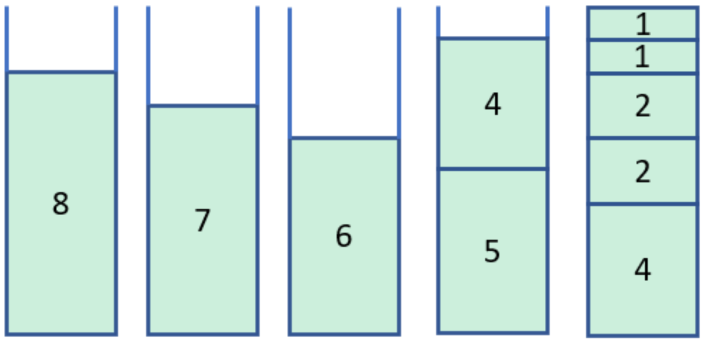


FIGURE 2.8: Solution using Next-Fit-Decreasing Algorithm.

- **Best-Fit Algorithm**: *"Place the items in the order in which they arrive. Place the next item into that bin which will leave the least room left over after the item is placed in the bin. If it does not fit in any bin, start a new bin"* [53]. The pseudocode of the algorithm is shown in Algorithm 6. Since all bins are examined in each step, the algorithm has an obvious running time of $\mathcal{O}(n^2)$. The solution related to the example is depicted in Figure 2.9.

Algorithm 6 Best-Fit

```
1: for All items $i$=1,2,...,$n$ do
2:     for All bins $j$=1,2,... do
3:         if Item $i$ fits in bin $j$ then
4:             Calculate remaining capacity after the object has been packed
5:         end if
6:     end for
7:     Pack item $i$ in bin $j$, where $j$ is the bin with minimum remaining capacity after adding the object (i.e. the object "fits best").
8:     If no such bin exists, open a new one and add the object
9: end for
```

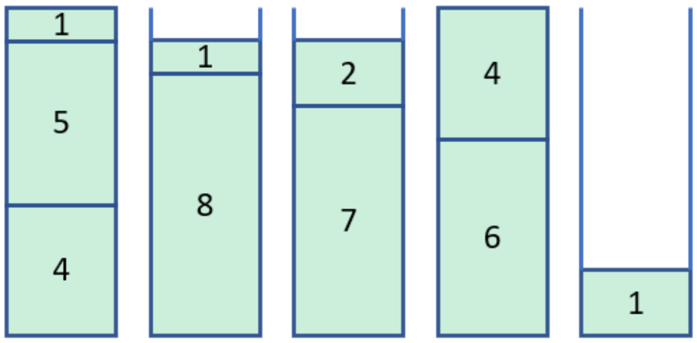


FIGURE 2.9: Solution using Best Fit Algorithm.

- **Best-Fit-Decreasing Algorithm:** The pseudocode of the algorithm is shown in Algorithm 7. Since Counting Sort has a complexity of $\mathcal{O}(n+k)$, where k is the largest weight, the algorithm is obviously dominated by the running time of First-Fit, which yields a factor of $\mathcal{O}(n^2)$. The solution related to the example is depicted in Figure 2.10 and it results equal to that generated by using First-Fit-Decreasing algorithm.

Algorithm 7 Best-Fit-Decreasing

```
1: Sort objects in decreasing order using Counting Sort
2: Apply Best-Fit to the sorted list of objects
```

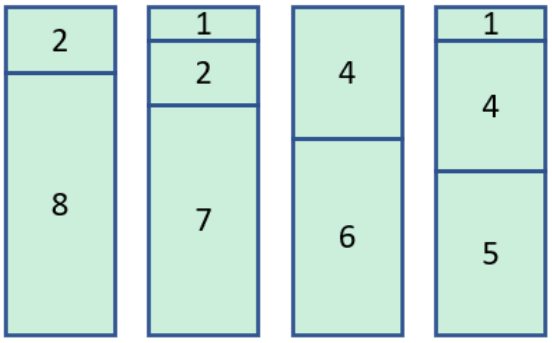


FIGURE 2.10: Solution using Best-Fit-Decreasing Algorithm.

2.3.1.2 Metaheuristics

In recent years, metaheuristic approaches to solve the 1D|MBS|BPP have become popular. This is largely due to these algorithm's ability to easily handle complex constraints that are present in real-life applications of optimization problems, and the fact that metaheuristics require less computational time to produce high-quality solutions [31]. Some of the most relevant metaheuristics are now briefly illustrated. In [2], the Whale Optimization Algorithm (WOA) is adopted. WOA is a swarm intelligent metaheuristic that imitates the peculiar hunting strategy of humpback whales, known as the 'bubble net strategy' [48]. This adaptation incorporated Lévy flights, an additional mutation phase, and a logistic chaotic map to enhance the exploration capabilities of the original algorithm. In [23], a modified version of the Squirrel Search Algorithm is adopted to solve the problem. This algorithm mimics the behaviour of flying squirrels and their use of gliding [33]. In [3], the Fitness Dependent Optimizer (FDO) is adapted to face the problem. FDO is a swarm intelligent algorithm that models the characteristics of the reproductive process of bee swarms, along with their collective decision-making behaviour. It calculates the velocity of a particle using the problem fitness function value to produce weights. These weights then guide the algorithm's search agents during the exploitation and exploration phases. An improved version of FDO is proposed in [58], where the original way of generating the first population is replaced with a random generation of the initial population, using an improved First-Fit heuristic and updating the operating strategies in the original algorithm to improve the exploration and exploitation phases. The advantages of this algorithm are that it is stable in both the two phases and that it requires fewer computations than other algorithms [52].

2.3.2 Two-Dimensional variant

The Two-Dimensional Single-Bin-Size Bin Packing Problem (2D|SBS|BPP) is the problem of packing a finite set of "small" rectangles, called items, into the minimum number of identical "large" rectangles, called bins, with the requirement that the items are packed without overlapping. In this problem, a set of n rectangular items $j \in J = \{1, ..., n\}$ is given, each having width w_j and height h_j, and an unlimited number of finite identical rectangular bins, having width W and height H. The problem is to allocate all the items to the minimum number of bins. Notice that, in the special case where $w_j = W$ where $(j = 1, ..., n)$, the one-dimensional variant is obtained. The problem has been classically addressed through heuristic techniques, whereas in the last years also metaheuristic approaches have been proposed. For details about exact methods refer to [47].

2.3.2.1 Heuristics

The heuristics in the following belong to the set of the off-line algorithms and they can be classified in two families:

- *One-phase algorithms* directly pack the items into the finite bins;
- *Two-phase algorithms* start by packing the items into a single strip of width W. In the second phase, the strip solution is used to construct a packing into finite $W \times H$ bins.

For a detailed survey on on-line heuristics, instead, refer to [20]. Before describing one and two-phase algorithms, let introduce three algorithms for packing the items into a strip, derived from the strategies for the one-dimensional case. As we will see, the they are also used as a first step in the two-phase algorithms.

Strip Packing

Firstly, in each case, the items are initially sorted by non-increasing height and packed in the corresponding sequence. Let j denote the current item and s the last created level:

- *Next-Fit Decreasing Height* (NFDH) strategy: item j is packed left justified on level s, if it fits. Otherwise, a new level ($s := s + 1$) is created, and j is packed left justified into it;
- *First-Fit Decreasing Height* (FFDH) strategy: item j is packed left justified on the first level where it fits, if any. If no level can accommodate j, a new level is initialized as in NFDH;
- *Best-Fit Decreasing Height* (BFDH) strategy: item j is packed left justified on that level, among those where it fits, for which the unused horizontal space is a minimum. If no level can accommodate j, a new level is initialized as in NFDH.

An example of these algorithm is depicted in Figure 2.11

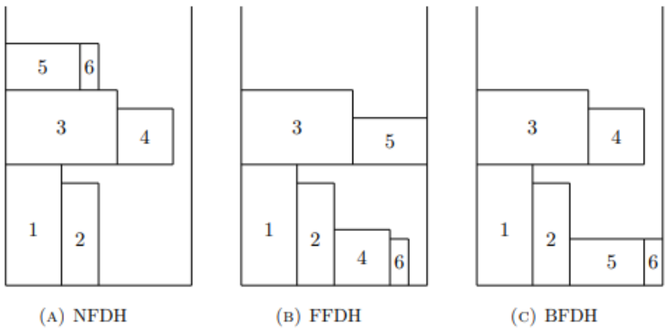


FIGURE 2.11: Three classical strategies for the strip packing.

Two-Phase Heuristics

A well-known two-phase algorithm, called *Hybrid First-Fit* (HFF), was proposed in [26]. In the first phase, the FFDH heuristic is used to obtain a strip packing. Let now $H_1, H_2, ...$ be the heights of the resulting levels. It results that $H_1 \geq H_2 \geq ...$. A solution for the original problem is obtained by solving a 1D|SBS|BPP with item sizes H_i and bin capacity H, through the *First-Fit Decreasing* algorithm described in Algorithm 2. Both phases can be implemented so as to require $\mathcal{O}(n \log n)$. An instance of the algorithm is shown in Figure 2.12.

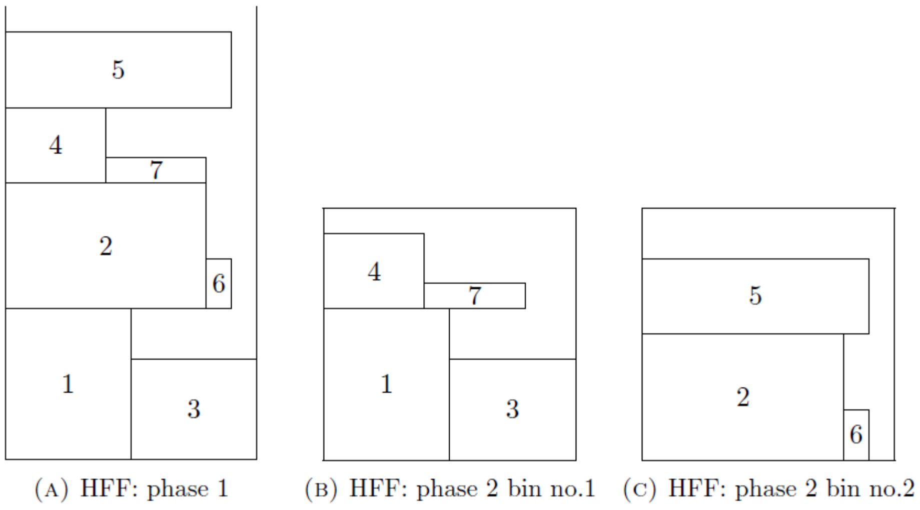
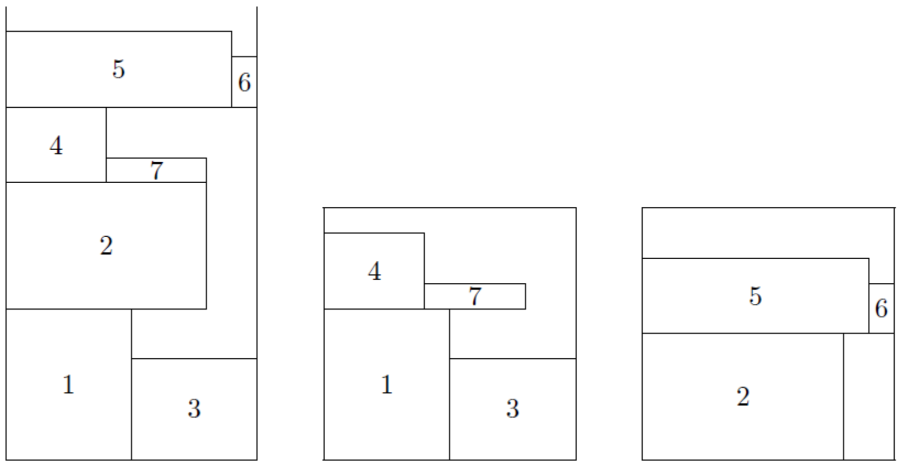


FIGURE 2.12: Algorithm HFF.

A variation of this algorithm was proposed in [9]. An initial strip packing is now obtained using the BFDH heuristic. Let now $H_1, H_2, ...$ be the heights of the resulting levels. Also in this case results that $H_1 \geq H_2 \geq ....$ A solution for the original problem is obtained by solving a 1D|SBS|BPP with item sizes H_i and bin capacity H, through the *Best-Fit-Decreasing* algorithm described in Algorithm 7. Both phases can be implemented so as to require $\mathcal{O}(n \log n)$. An instance of the algorithm is shown in Figure 2.13.



(A) FBS: phase 1 (B) FBS: phase 2 bin no.1 (C) FBS: phase 2 bin no.2

FIGURE 2.13: Variation of HFF algorithm.

Let us consider now another variation of HFF, denoted *Hybrid Next-Fit* (HNF), in which the NFDH strategy is adopted in the first phase, while the 1D|SBS|BPP is solved through the Next-Fit Decreasing algorithm. Due to the next-fit policy, this algorithm is equivalent to a one-phase algorithm in which the current item is packed on the current level of the current bin, if possible; otherwise, a new (current) level is initialized either in the current bin (if enough vertical space is available), or in a new (current) bin. Also this strategy can be implemented so as to require $O(n \log n)$ time complexity.

One-Phase Heuristics

Regarding one-phase heuristics, instead, they can be divided into 2 main categories: level and non-level heuristics.

- *Level heuristics*: two classical one-phase level algorithms were presented in [9] and [29] . Algorithm *Finite Next-Fit* (FNF) directly packs the items into finite bins exactly in the way algorithm HNF does. An example is shown in Figure 2.14.

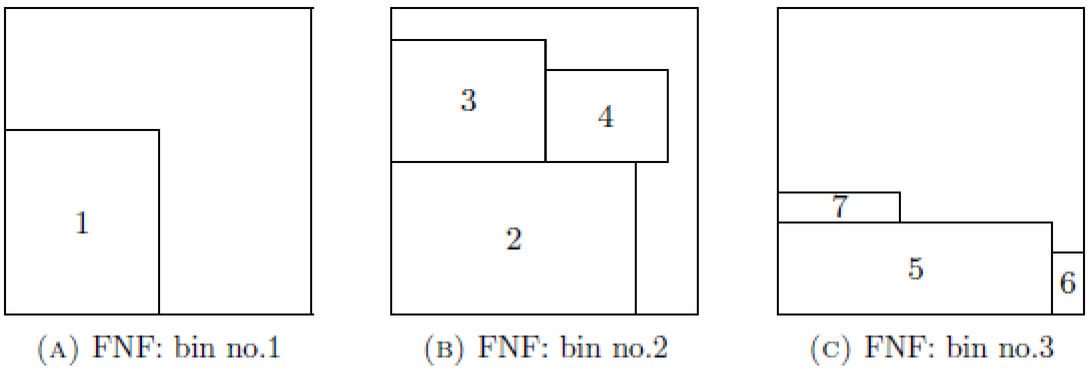


FIGURE 2.14: FNF algorithm.

Algorithm Finite First-Fit (FFF) adopts instead the FFDH strategy. The current item is packed on the lowest level of the first bin where it fits; if no level can accommodate it, a new level is created either in the first suitable bin, or by initializing a new bin (if no bin has enough vertical space available). An example is shown in Figure 2.15.

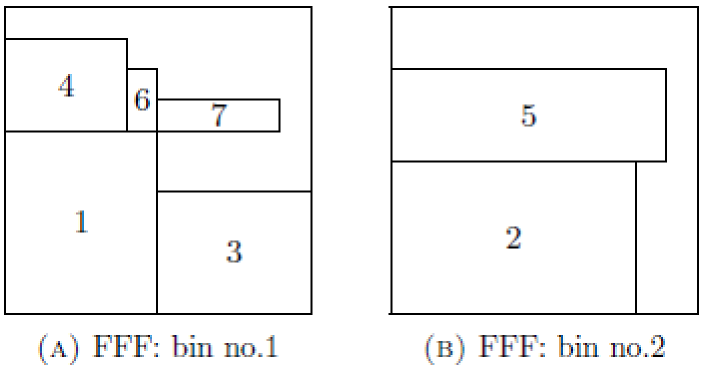


FIGURE 2.15: FFF algorithm.

Both algorithms can be implemented so as to require $O(n\log n)$ time.

- *Non-Level heuristics*: the main non-level strategy is known as Bottom-Left (BL) and consists in packing the current item in the lowest possible position, left justified. The BL approach for the finite bin case is proposed in [9]. Their algorithm, denoted *Finite Bottom-Left* (FBL), initially sorts the items by non-increasing width. The current item is then packed in the lowest position of any initialized bin, left justified; if no bin can allocate it, a new one is initialized. An example is shown in Figure 2.16. The computer implementation of algorithm BL was studied in [7], where a method for producing a packing in $O(n^2)$ time is given. The same approach was adopted in [9].

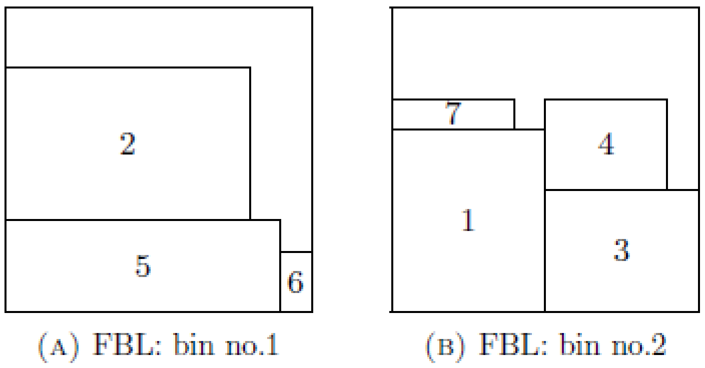


FIGURE 2.16: FBL algorithm.

In [9], it is also proposed the Next Bottom-left (NBL) algorithm which is similar to FBL but, in this case, the generation of a new bin for packing means that all the free spaces from the previous bin are discarded. Thus only one bin is active at a time. An example is shown in Figure 2.17.

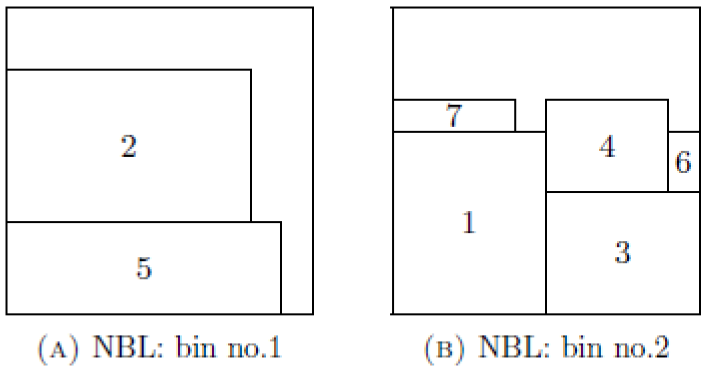


FIGURE 2.17: NBL algorithm.

2.3.2.2 Metaheuristics

In [42],[40],[41], Tabu Search algorithms for 2BP and for variants has been developed. In particular, these strategies involve the possibility of rotating the items by 90°or the additional constraint that the items may be obtained from the resulting patterns through guillotine cuts, i.e., straight bisecting lines going from one edge of a rectangle to the opposite edge. Guillotine cutting is particularly common in the glass industry, where glass sheets are scored along horizontal and vertical lines, and then broken along these lines to obtain smaller panels. It is also useful for cutting steel plates, cutting of wood sheets to make furniture, and cutting of cardboard into boxes [8]. The Unified Tabu Search framework proposed in [41], whose main characteristic is the adoption of a search scheme and a neighborhood which are independent of the specific packing problem to be solved, is now described. It can thus be used for virtually any variant of 2D|BPP, by simply changing the specific deterministic algorithm used for evaluating the moves within the neighborhood search. Given a current solution, the moves modify it by changing the packing of a subset S of items, trying to empty a specified target bin selected among those that currently pack a small area and a relatively large number of items. Subset S is defined so as to include one item, j, from the target bin and the current contents of k other bins, and the new packing is obtained by executing an appropriate heuristic algorithm on S. If the move packs the items of S into k (or less) bins, i.e., item j has been removed from the target bin, a new item is selected, a new set S is defined accordingly, and a new move is performed. Otherwise, S is changed by selecting a different set of k bins, or a different item j from the target bin.

The above framework above was suitably combined with a genetic algorithm in [32] so as to get a hybrid algorithm for 2D|BPP that can be easily adapted to other packing problems.

A different metaheuristic for 2D|SBS|BPP has been proposed in [25]. Their guided local search algorithm starts from a feasible solution, and randomly removes some bins by assigning the corresponding items to the other bins. The new solution is generally infeasible, leading to an optimization problem in which one is required to minimize an objective function that measures the pairwise overlapping area. The associated neighborhood is explored through object shifts, until a feasible solution is found.

Moreover, in [14] and [15], an effective randomized multi-start heuristic for 2BP has been proposed. The procedure is the following:

1. Assigns a score to each item;
2. Packs the items, one at a time, according to decreasing values of the corresponding scores;
3. Updates the scores by using a specified criterion, and
4. Iterates on 2 and 3 until an optimal solution is found or a maximum number of iterations has been performed.

The execution of the algorithm is repeated for a given set of different criteria used for the updating of the object scores.

In [49], a two-phase heuristic algorithm has been formulated. In the first phase (column generation), a large set of different feasible patterns is produced by using heuristic algorithms from the literature, while in the second phase (column optimization) a subset of patterns is selected by heuristically solving the associated set covering instance.

In [59], an hybrid genetic algorithm and a simulated annealing techinque is proposed for two-dimensional non-guillotine rectangular packing problems. In the paper, GA and SA are used separately to obtain permutation for placing the small pieces. Improved BL algorithm are employed to place rectangular pieces. The solution approach can be summarized as:

- GA and SA are used to find permutations with small trim loss.
- An improved BL algorithm is used to place rectangular pieces corresponding to a particular permutation.

The current literature on the bin packing problem mostly focuses on the minimization of wasted space. However, in most bin packing problems, both minimization of wasted space and balance of the bins needs to be achieved. For this purpose, a multiobjective two-dimensional bin packing model (MOBPP-2D) with minimum wasted space and balancing of loads has been formulated in [39].

2.3.3 Three-Dimensional variant

The Three-Dimensional Single-Bin-Size Bin Packing Problem (3D|SBS|BPP) in a real-industrial scenario is the main problem addressed in this thesis. As said above, it consists in packing a finite set of “small” cuboids, called items, into the minimum number of identical “large” rectangles, called bins, with the requirement that the items are packed without overlapping. Also this variant of the problem has been addressed through heuristic/metaheuristic approaches if a real-world point of view is considered. For details about exact methods, instead refer to [46].

2.3.3.1 Real-world constraints

In the following, the geometric and the specific constraints are distinguished and explained. Notice that, the latter type of constraints is not considered in the typology described in [64].

Geometric constraints

This typology of constraints include that all box must lie entirely within the bin. Moreover, they do not overlap and are assumed to be placed orthogonally, i.e., the edges of the boxes have to be only parallel or perpendicular to the bin edges.

Specific constraints

In [**Bortfeld_2013**], on the basis of [10], the authors presented a huge set of additional constraints that might be encountered in real-world packing situations. Each type of constraints is explained in the following, clarifying the selection of those taken into account in this work.

- **Bin weight limit**: typically, bins have a maximal weight limit, thus the total sum of the mass of the loaded items must not exceed this threshold. This type of constraint is common in literature (e.g., [60], [18], [17]) and it is particular relevant when there are some high density items.

  In this work, this constraint is not taken into account since the total weight of the products to be loaded is supposed to be always less than the bin weight limit. However, it is an easy task inserting also this constraint into the algorithm.

- **Load balance constraints**: they are also called weight distribution constraints. Satisfying these constraints ensure that the height of the centre of mass of the loaded bin is as much as possible across the floor, while the geometric centre of the related plane projection is as much as possible near to the geometric centre of the bin floor. In this way, a balanced bin is obtained, reducing the risk of shifting when the bin is moving and thus the risks of accidents. Moreover, some operations such as bin lifting may even become impossible if the weight is not well balanced (see, e.g., [18], [50], [30], [61]). This type of constraint can also be met in road transportation. Indeed, the loaded truck has to respect a precise distribution of the cargo over the axles of the vehicle ([56], [5]). Legislation about axle weight limits varies between countries and more details for European countries can be found in [27].

  Therefore, load balance constraints are crucial and they are taken into account in this work.

- **Loading priorities for the items**: Some of the available items can have higher priority than the loading of others [35].

However, since this thesis deals with a input minimization problem (i.e., all the boxes have to be loaded) this constraint is not considered.

- **Orientation constraints for the items**: Items are assumed to be placed orthogonally, i.e., the edges of the boxes have to be parallel or perpendicular to those of the bin. If the box can rotate and if each dimension can be in a vertical position, then six orientations are possible and they are shown in Figure 2.18. In practice, however, some boxes may not rotate in all directions because of their contents. These constraints are called *orientation constraints* (see, e.g., [28], [19], [17], [35]).

  In this thesis, it is considered that some dimensions may not be in a vertical position, hence reducing the admittable orientations.

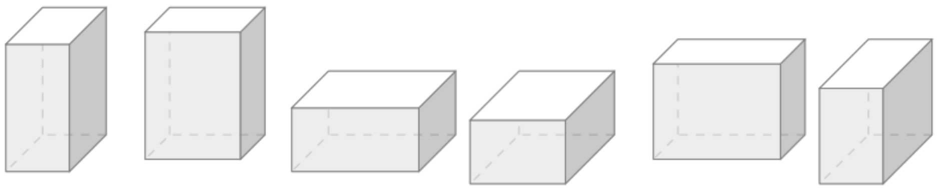


FIGURE 2.18: Six possible orientations.

- **Stacking constraints**: Also called load-bearing constraints, this type of constraint describes how many boxes can be placed on top of each other. More generally, load bearing strength refers to the maximum pressure that can be applied over the top face of a box without damaging it. How much pressure or weight box can hold depends on the material and the construction of the boxes. Different strategies have been developed to manage this feature (see, for example, [60], [38], [35], [17]). This constraint is quite important in practice because it prevents damage to products contained in a fragile box.

  In this work, a box is said to be fragile if only other fragile boxes can be placed on it.

- **Complete shipment constraints**: As far the loading priority constraints the complete shipment constraints appear only with the output maximisation objective. Since there is a selection of small items, some are left behind. This type of constraint states that if one item of a subset is loaded, then all other items of the same subset have to be packed as well. Inversely, if one box of a subset cannot be packed, then no item of the same subset can be loaded [24]. This kind of constraint may arise, for example, when parts of a piece of furniture are packed separately and have to be assembled on site at a customer's location.

This constraint is not considered in this thesis since all the boxes must be packed.

- **Allocation constrains**: This type of constraint arises when there are several bins in the problem [62], [60], [24], [4]. It is possible to distinguish:

  - *Connectivity constraints*: demand that items of a particular subset go into the same bin, for example because they go to the same destination;
  - *Separation constraints*: require tat some items are accommodated in different bins for safety reasons.

  These constraints are not considered in this thesis because they represent particular cases.

- **Positioning constraints**: These constraints limit the location of items on the bin either in absolute or in relative terms [60], [38]. On the one hand, absolute positioning constraints specify where items should or should not be located on the bin. For instance, volatile liquids or explosives should be packed near the opening of the bins so that they can be accessed and removed quickly if necessary. On the other hand, relative positioning constraints state whether items should or should not be located close to each other. For instance, items which alter the quality of other items (like food and petrol) must not be placed next to each other.

  This type of constraint is not considered here.

- **Stability constraints**: Load stability is one of the most important types of constraint. Indeed, unstable loads may result in a damaged loaded bin. Bin stability involves the vertical (or static) and the horizontal (or dynamic) stability [60], [34], [38], [18], [35], [17]. More in details:

  - *Vertical stability*: the bottom side of each box needs to be supported by the top face of other boxes or by the bin floor. This constraint is also called *static stabilty* as it deals with static bins. The vertical stability excludes floating boxes.

  - *Horizontal stability*: this constraint refers to the capacity of the box to withstand the inertia of its own body when being moved. The boxes remain in their position with respect to the main plane axis of the bin.

  This thesis only considers vertical stability because horizontal stability could be obtained by adding a special sheet increasing the friction between items.

- **Complexity constraints**: The proposed loading patterns sometimes have to be easy enough for the loading human operator to be able to visualize quickly. Moreover, more automatic packing technologies are not always suitable for complex bin packing. The most frequently constraint of this category is the guillotine cutting constraints, already discussed in the two-dimensional case. However, this kind of constraints is not always appropriate in practice since it may reduce the stability of the loaded bin when being transported.

  Then, for this reason, these will not taken into account in this thesis.

In order to measure how often the different constraints are considered, in [13] the authors listed 163 papers that are publicly available and published between 1980 and the end of 2011 in English in international journals, edited volumes and conference proceedings. Among these 163 papers, only 36 (i.e., 22%) addressed the 3D|SBS|BPP. The authors also present the number of papers in which constraint types have been addressed. The percentages are represented in Figure 2.19.

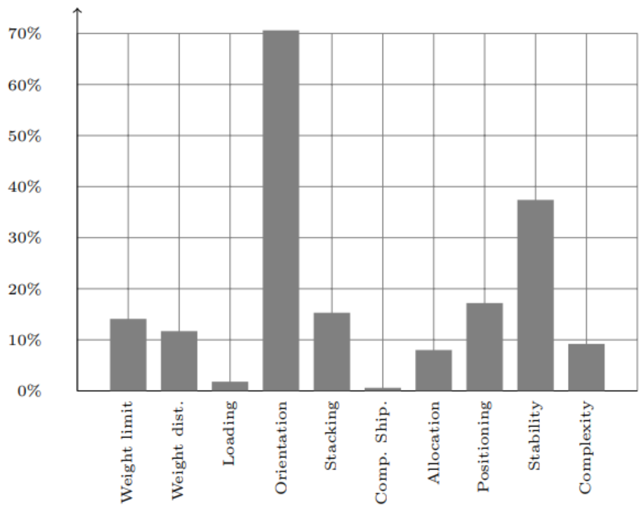


FIGURE 2.19: Percentage of papers in which constraint types have been addressed (population size = 163).

2.3.3.2 Heuristic/metaheuristics methods

To solve this problem, the best results in literature have been obtained combining together heuristic and metaheuristic techniques. The best performing ones are now briefly explained.
A Tabu Search algorithm for the 2D|SBS|BPP was proposed in [40]. The algorithm consists of two simple constructive heuristics to pack the items into bins, and a Tabu Search mechanisms to control the movement of items between bins. Two neighborhoods were considered to try to move an item from the weakest bin (i.e., the bin that appeared to be the easiest to empty) into another. Since the constructive heuristics produced guillotine packings, so did the overall algorithm. The authors generalized this approach to the 3D|SBS|BPP in [43]. In [25], a Guided Local Search (GLS) heuristic for the 3D|SBS|BPP is presented. Starting with an upper bound on the number of bins obtained by a greedy heuristic procedure, the algorithm iteratively decreased the number of bins, each time searching for a feasible packing using the GLS method. The process terminated when either a given time limit was reached or the current solution matched a precomputed lower bound. Moreover, two constructive heuristics have been developed and tested for the 3D|SBS|BPP in [46] . The first algorithm, called *S Pack*, was based on a layer-building principle. The second heuristic, denoted *MPV-BS*, repeatedly filled one bin after the other by means of the Branch & Bound algorithm for the single container presented by the authors in the same paper. To reduce the computational time of the algorithm, the Branch & Bound is truncated by limiting the width of the tree. In [1], a new shelf-based heuristic for the 3D|SBS|BPP, called *Height first - Area second* (HA) is presented. The algorithm was based on constructing two solutions and selecting the best. To obtain the first one, items were partitioned into clusters according to their height and a series of layers were obtained from each cluster. The layers were then packed into bins using the Branch & Bound algorithm developed in [45]. The second solution was obtained by ordering the items by non-increasing area of their base and building new layers. As previously, layers were packed into bins by solving a a set of 1D|SBS|BPP problem. HA is the constructive heuristic that currently obtains the best results on the benchmark test problem instances. Notice that, although the performances are good, none of the mentioned algorithm is able to provide as output also the optimal boxes input sequence, that is an essential requirement in a real-industrial scenario.

Considering this, the main contribute of the thesis is to develop an algorithm able to guarantee excellent solutions for each input, providing also the optimal boxes input sequence within a reasonable time and exploring other metaheuristic techniques.

Chapter 3

Preliminary mathematical tools

3.1 Feedforward Neural Networks

In this chapter, a brief summary of feedforward neural networks is shown, discussing how they can be adapted to our problem. An Artificial Neural Network (ANN) is an aspect of Artificial Intelligence (AI) that is focused on emulating the learning approach that humans use to gain certain types of knowledge. Like biological neurons, which are present in the brain, ANN also contains a number of artificial neurons, and uses them to identify and store information [21]. ANN consists of input and output layers, as well as (in most cases) one or more hidden layer(s) consisting of units that transform the input into something that the output layer can use. The role of each type of layers is the following:

- Input layer: this layer accepts input features. It provides information from the outside world to the network, no computation is performed at this layer, nodes here just pass on the information (features) to the hidden layer.
- Hidden layer: nodes of this layer are not exposed to the outer world. Hidden layer performs computations on the features entered through the input layer and transfer the result to the output layer.
- Output layer: this layer bring up the information learned by the network to the outer world.

In Figure 3.1, the complete structure of an artificial neuron is shown.

Except for the neurons of the input layer, each neuron uses a nonlinear activation function. Moreover, a bias node could be added in the input and hidden layers to increase the flexibility of the model to fit the data. The most used nonlinear activation functions are the Rectified Linear Unit (ReLU), Sigmoid and Tanh activation functions. In [54], the most common activation

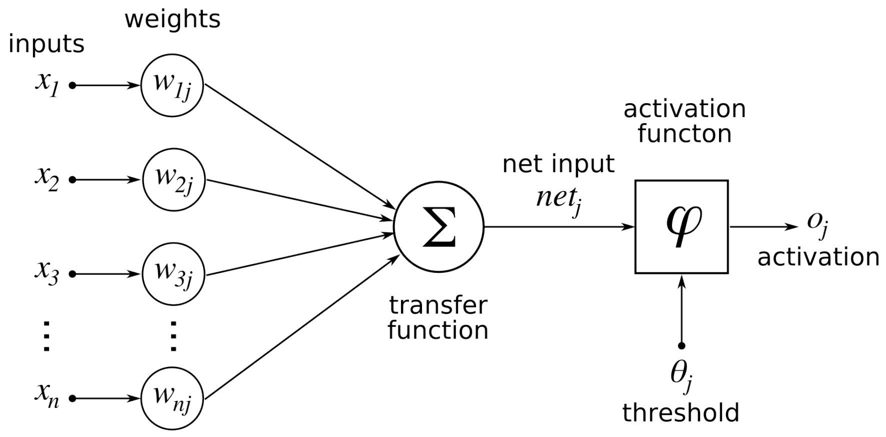


FIGURE 3.1: Structure of an artificial neuron.

functions are discussed and compared. To address our problem, a particular class of ANN, represented by Feedforward Neural Network (FNN) is adopted. It is a fully connected network moving only in forward direction (no loopback). An example of FFN with three hidden layers is shown in Figure 3.2.

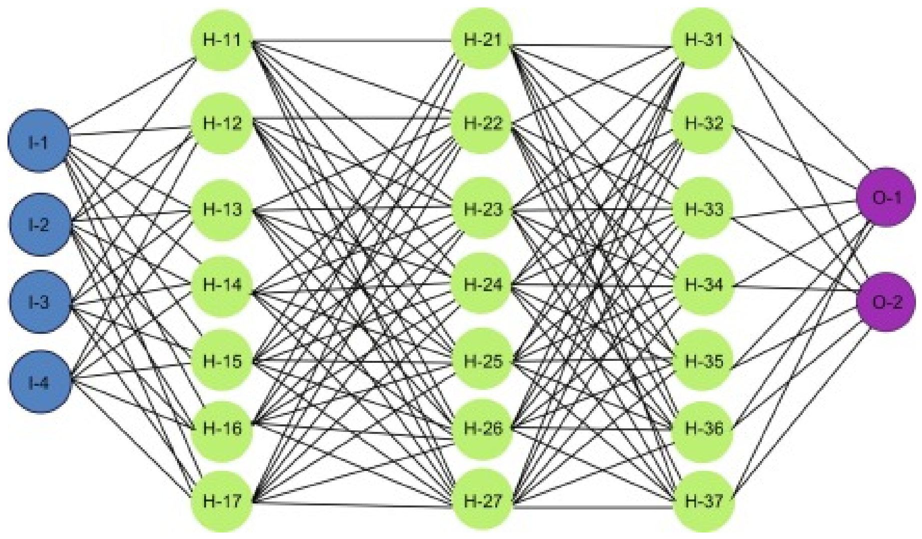


FIGURE 3.2: FNN with three hidden layers.

FNN is suitable for both classification and prediction problems. For the training process, a supervised learning technique is utilized [44], [57]. This algorithm uses a training set to teach network to yield the desired output, a validation set to avoid overfitting and a test set used to provide an unbiased evaluation of a final model fit on the training set. The union of these three sets forms the entire dataset. It must be created ad hoc, based on the problem under consideration. In particular, each element of the dataset is generated starting from a real order and consists of an input vector and a target vector. The input vector is created through a feature extraction process and it indicates the quantity of boxes present in the order, for each type. The target vector, on the other hand, represents the optimal weights associated to the sum-weighted objective functions of the optimization processes that generated the optimal solution. The dataset creation is discussed in details in Chapter 5.

3.2 Genetic Algorithms

Genetic Algorithms (GAs) are iterative optimization procedures that repeatedly apply GA operators (such as selection, crossover, and mutation) to a group of solutions until some criterion of convergence has been satisfied. In a GA, a search point (solution) is a setting in the search space with dimensions n and it is coded into a string, $x = (x_1, .., x_n)$, which is analogous to a chromosome in biological systems. The string/chromosome is composed of n characters, $x_1, ..., x_n$, which are analogous to the n genes. A set of chromosomes (or individuals) is called *population*. Each iterative step in which a new population is obtained is called *generation*. A GA hybridized with a heuristic is called Hybrid Genetic Algorithm (HGA). A basic HGA procedure has the following steps.

1. Define an objective/fitness function, and set the GA operators (such as population size, parent/offspring ratio, selection method, number of crossovers, and mutation rate);

2. Generate the initial population in a random way or using heuristics;

3. Evaluate the objective function for each individual (chromosome or solution) in the initial population;

4. Generate an offspring population by using GA operators (such as selection/mating, crossover, and mutation);

5. Evaluate the objective function of each individual in the offspring population using a heuristic procedure;

6. Decide which individuals to include in the next population;

7. If a stopping criterion is satisfied, then the procedure is halted. Otherwise, go to step 4.

In order to adapt this framework to the problem under consideration, n will indicate the number of boxes to be loaded and $x_k \in \{1, ..., n\}$ will represent the k-th box placed. Therefore, a chromosome $x = (x_1, ..., x_n)$ will be an input boxes sequence.

Chapter 4

A Two-Stage Algorithm for the Three-Dimensional Single-Bin-Size Bin Packing Problem

This chapter is organized as follows: in Section 4.1, a revisitation and a synthesis of the arguments already addressed in [6] is presented, in order to get the paper reading clearer. In Section 4.2, the complete structure of the algorithm is presented. Finally, in Section 4.3, simulation results are presented and discussed.

4.1 Hybrid Genetic Algorithm setup

4.1.1 Chromosome structure

Let $\mathcal{B} = \{b_1, b_2, ..., b_s\}$ be an ordered set containing all possible topology of boxes, where each element is characterized by a height, widht and depth value. Let $\mathcal{U} = \{u^1, u^2, ..., u^m\}$ be a high-cardinality set of orders. Considering an order u^i and defining u^i_c as the number of boxes of type b_c present in the order, it is possible to write $u^i = [u^i_1, u^i_2, ..., u^i_s]$. Moreover, let $n_i = \sum_{c=1}^{s} u^i_c$ representing the total number of boxes in the order. Given an order u^i, a HGA chromosome $c^i_d \in \{c^i_1, ..., c^i_l\}$ represents an input boxes sequence and it is composed of n_i genes, $x^i_{d,1}, ..., x^i_{d,n_i}$, where $x^i_{d,k} \in \{1, ..., n_i\}$ denotes the k-th box placed. As a direct consequence of this choice, the genetic algorithm will run only on this information (using the classical genetic operators such as selection, mutation and crossover), allowing boxes orientations and anchor points to be free optimization variables of the evaluation process. Moreover, in the special case of identical items, this choice allows us to use only one chromosome and one generation to find a suitable solution. In this way, the algorithm will turn into a non-genetic version, reducing calculation time even more.

4.1.2 Initial population

The chromosomes of the initial population are heuristically generated based on the following observations.

- The larger products, in most cases, should be packed first, so that the less bulky boxes can be placed into the small remaining spaces.
- In some cases, it is more convenient to insert boxes starting from the less voluminous ones, for example when many small and a few large boxes need to be loaded, so that the large boxes can be laid on the layers formed by the small ones.

Therefore, to be able to cover as many cases as possible, we divide the first population into three subsets: the main subset is the one composed of those chromosomes created in according to the first observation. The chromosomes belonging to the second subset will be created according to the second observation while those in the third subset will be randomly created. The number of chromosomes belonging to each subset is a parameter of the algorithm.

4.1.3 Evaluation process

The chromosome evaluation process is performed only if the input sequence coded in the chromosome is feasible, i.e., if it is a generic permutation of the sequence $(1, 2, ...n)$. In this case, for each box to place, a constrained minimization problem has to be solved, with respect to both anchor points and box orientations that represent the only free variables of the problem. Regarding orientations, since the boxes are assumed to rotate only orthogonally and since any of the four vertices of the lower face of the box can coincide with the candidate anchor point, there are 24 possible orientations (2 orientations along x-axis, 2 orientations along y-axis and 2 orientation along z-axis, for each vertex of the lower face). Notice that, in case of fragile box, some orientations may not be recommended. This is taken into account using a suitable fitness function. Regarding anchor points, they are generated using the following heuristic procedure.

4.1.3.1 A new heuristic procedure

From an analysis of heuristics proposed in the literature, one of the most performing is the one based on the concept of corner points ([46]). The proposed heuristic can be seen as its extension, allowing the algorithm to explore many more configurations within a reasonable computational time. Initially, when the bin is still empty, the only candidate anchor point for the first box will be the point at the centre of the bin. Once a box has been loaded (using both optimal anchor point and optimal orientation according to the fitness function), the algorithm will add the eight vertices of the placed box to the set of candidate anchor points for the following box in the input sequence. The chosen anchor points is not removed from the set, if it results usable again. Given a box to place, a feasibility check is required for all the possible pairs (anchor point, orientation) and, only among all the feasible pairs, the one that minimizes the fitness function is chosen.

4.1.3.2 Feasibility check

A pair (anchor point, orientation) is considered feasible if the related box satisfies these hard constraints:

- Geometric constraints:
  - The box must lie completely on the bin;
  - There must be no overlap between the placed boxes;
  - The height of the boxes must not exceed the maximum bin height threshold;
- Stability constraint:
  - At least s% and v vertices ($v \in \{1, 2, 3, 4\}$) of the lower surface of the box must lie on the bin floor or on other boxes. Choosing $s \geq 70$ and $v \geq 3$ ensures that its center of mass will always lie in the convex hull of all the contact points ([55]).

If all the pairs are infeasible, the fitness value of the chromosome is set to $+\infty$.

4.1.3.3 Fitness function

The aforementioned suitable fitness function can be written as:

$$f_k = w_1 f_{1,k} + w_2 f_{2,k} + w_3 f_{3,k} + w_4 f_{4,k} \tag{4.1}$$

where w_h are normalized positive weights ($h \in \{1, ..., 4\}$) and $f_{h,k}$ represents normalized fitness function. More in details, once the k-th box is placed:

- $f_{1,k}$ represents the maximum height of the boxes;
- $f_{2,k}$ denotes the height of the center of mass of the boxes;
- $f_{3,k}$ indicates the Euclidean norm between the chosen anchor point and the center of the bin floor;
- $f_{4,k}$ represents the number of fragile boxes on the bin.

In order to evaluate the whole chromosome, a fitness function $f^{i,d}$ is needed and obtained once all boxes are placed. It will be:

$$f^{i,d} = f_{n_i}. \tag{4.2}$$

4.1.4 Genetic Operators

4.1.4.1 Selection

A modified version of the Tournament Method for selection is adopted. Once all the chromosomes have been evaluated, n_{rs} chromosomes are randomly selected and, among these, the best chromosome is chosen for the next generation. This process is repeated until the 80% of the new population is created. Then, the best chromosome in the entire population is selected and replicated to complete the new population. In order to preserve the best chromosome, the crossover and mutation processes will not act on some of its clones.

4.1.4.2 Crossover and mutation

In the literature, crossover and mutation processes are conducted on each newly generated offspring with constant probability P_c and P_m, respectively. In this work, these probabilities are assumed to be generation-varying decreasing functions. By doing so, the algorithm tries to explore the solution spaces in a very exhaustive way at the begin and then it tries to converge to the optimal direction. The crossover is a one-point version: a random location is selected for two parents and the two parts after the crossover point of the two parents are switched over to form two children ([63]). If the newly formed children are not feasible due to some boxes appear in the children twice while some do not appear at all the chromosome is considered infeasible and its fitness values is set to $+\infty$. Regarding mutation, for each chromosome, two positions are randomly selected and the elements on these positions are swapped.

4.2 A new two-stage algorithm

The two-stage algorithm proposed in this work consists of a coupling between the previous HGA and a FNN. The latter is exploited to learn, as a function of the input instances, the optimal weight vector related to the local HGA fitness function and the optimal weight vector related to a new global weighted-sum objective function. The aim of the first stage is to create a rich dataset in order to train, validate and test the neural network. The HGA is run several times on each order within a large set of packing instances, using a different fitness weight vector at each iteration, with the aim of generating a rich set of solutions. Once the set is created, the best solution is chosen for each order, optimizing a new global weighted-sum objective function. Relying on a graphical user interface, it is possible to choose the optimal weight vector related to the optimized global objective function, showing in real-time the dependence of the optimal solution on the weight vector. Moreover, it is possible to keep track of the weights vector associated to the local HGA fitness that generated the chosen optimal solution.

4.2.1 First stage

4.2.1.1 Dataset creation

Let $\mathcal{W} = \{w_1, w_2, ..., w_p\}$ be a set of p local weight vectors related to the local weighted-sum fitness function optimized in the chromosome evaluation process of the HGA. In particular, the j-th local weight vector is $w_j = [w_{j,1}, w_{j,2}, w_{j,3}, w_{j,4}]$, where $j \in \{1, 2, ..., p\}$ and $w_{j,h} \in [0, 1], \forall h \in \{1, 2, 3, 4\}$. These vectors are found relying on trial-and-error techniques and they are able to cover a wide range of configurations during the boxes loading process. Opposite to what happened in our previous algorithm, in which only one solution (the optimal one) was generated as output, our aim now is to generate a rich set of solutions for each order in $\mathcal{U}$, running the HGA p times, varying w_j and considering the r_i best solutions found during HGA process at each iteration. Notice that, for each solution, we keep track of the local weight vector that generated it. Considering the order u^i, the HGA fitness function in (4.1) for the local weight vector w_j can be rewritten as:

$$f_{j,k} = w_{j,1}f_{1,k} + w_{j,2}f_{2,k} + w_{j,3}f_{3,k} + w_{j,4}f_{4,k}. \tag{4.3}$$

This extended version of (4.1) is optimized when the k-th box of a generic chromosome c_d^i has to be placed ($k \in \{1, 2, ..., n_i\}$), fixing w_j as local weight vector. In order to evaluate the whole chromosome, an extended version of (4.2) is needed and obtained once all boxes are placed. It will be:

$$f_j^{i,d} = f_{j,n_i}. \tag{4.4}$$

Once all p simulations related to the i-th order are completed, a set $\mathcal{S}_i = \{c^i_{\text{best},1}, c^i_{\text{best},2}, ..., c^i_{\text{best},b}\}$ containing the best $b = p \cdot r_i$ solutions is created. The optimal solution will be the one that minimizes a new global weighted fitness, calculated for each chromosome $c^i_{\text{best},z}$ belonging to S_i. It can be written as:

$$g^i = w^i_1 g_1 + w^i_2 g_2 + w^i_3 g_3 + w^i_4 g_4 + w^i_5 g_5 \tag{4.5}$$

where g_t are new normalized fitness functions and w^i_t are normalized positive global weights ($t \in \{1, ..., 5\}$). In particular:

- g_1 represents the ratio between the total volume of the loaded boxes and the volume of the associated convex hull;
- g_2 represents the number of fragile boxes;
- g_3 represents the stackability index of the bin. It is calculated as:

$$g_3 = \beta \cdot (1 - A_{\text{stack}}) \cdot D \tag{4.6}$$

  where:

  - β is a boolean value. It is set to 1 if all boxes at maximum height are not fragile. Otherwise, it is 0. In the case of $\beta = 1$, the fragility constraint ensures that no fragile boxes will be damaged by the stacking process.
  - A_{stack} indicates the normalized area of the convex hull C_{stack} of the set $\mathcal{S}_{\text{stack}}$, containing the vertices of the top face of each boxes at maximum height;
  - D represents the normalized Euclidean distance between the geometric center of the bin floor and the projection of the C_{stack} centroid on it.

- g_4 and g_5 represent the global strapping indices along the width and length directions of the loaded bin. Let x, y and z denote the three orthogonal axes related to the width, length and height directions of the loaded bin. Furthermore, let $\mathcal{S}_{\text{proj}}$ be a set containing the projections onto the xy-plane of the vertices of the placed boxes and $C_{xy} = (C_x, C_y)$ the centroid of its convex hull. The two indices are calculated, respectively, as:
$$\begin{cases} g_4 = 1 - \min(s_{x,1}, s_{x,2}, ..., s_{x,n_i}) \\ g_5 = 1 - \min(s_{y,1}, s_{y,2}, ..., s_{y,n_i}) \end{cases} \tag{4.7}$$
where $s_{x,k}$ and $s_{y,k}$ represent the normalized strapping indices along x-axis and y-axis of the k-th box of the sequence. Considering the projection $B_{xy,k} = (B_{x,k}, B_{y,k})$ onto the xy-plane of the centroid of the k-th box, the index $s_{x,k}$ ($s_{y,k}$) can be calculated as follows:
  - Case $B_{x,k} \neq C_x$ ($B_{y,k} \neq C_y$): some contact forces act along the x-axis (y-axis) during the strapping process. Let F_x (F_y) be the box face parallel to the yz-plane (xz-plane) and such that its centroid is a minimum distance from C_{xy}. Let A_x (A_y) and $A_{\text{touch},x}$ ($A_{\text{touch},y}$) be, respectively, the area of F_x (F_y) and the area of the portion of F_x (F_y) touching other boxes. It is possible to write: $s_{x,k} = \frac{A_{\text{touch},x}}{A_x}$ $\left(s_{y,k} = \frac{A_{\text{touch},y}}{A_y}\right)$.
  - Case $B_{x,k} = C_x$ ($B_{y,k} = C_y$): no contact forces act along the x-axis (y-axis) during the strapping process. Then, $s_{x,k}$ = 1 ($s_{y,k}$ = 1).

Relying on a graphical interface that permits to modify the global weights and to show in real-time the four best solutions in S_i, it is possible to create a target vector y_i related to the input vector u_i to be included into a dataset used for the training, validation and testing phase of the neural network. In particular, let $\bar{W}_g^i = [\bar{w}_1^i, \bar{w}_2^i, \bar{w}_3^i, \bar{w}_4^i, \bar{w}_5^i]$ be the selected global weight vector related to the global fitness g^i and $\bar{W}_l^i = [\bar{w}_{j,1}^i, \bar{w}_{j,2}^i, \bar{w}_{j,3}^i, \bar{w}_{j,4}^i]$ the local weight vector that generated the chosen optimal solution. The target vector for the input vector u_i will be $y^i = [\bar{W}_l^i, \bar{W}_g^i]$. Once all these tasks are completed for all orders in $\mathcal{U}$, the entire dataset can be created: $\mathcal{D} = [(u_1, y_1), (u_2, y_2), ..., (u_m, y_m)]$.

4.2.1.2 Training process

The dataset $\mathcal{D}$ is randomly shuffled and divided into three subsets, using the following percentages:

- Training set $\mathcal{D}_{\text{train}}$: 60%;

- Validation set $\mathcal{D}_{\text{valid}}$: 20%;
- Test set $\mathcal{D}_{\text{test}}$: 20%.

For the training process, the backpropagation algorithm is adopted, using the mean square error (MSE) as loss function to be minimized. In order to avoid overfitting, the early stopping criterion is adopted [51], calculating the MSE on the validation set after each training epoch and stopping the training process when it tends to increase. When the training process is completed, the generalization performance of the FNN is calculated on the test set. The results are reported in Section 4.3.

4.2.2 Second stage

Let $\mathcal{U}_{\text{test}} = \{u^1_{\text{test}}, u^2_{\text{test}}, ..., u^m_{\text{test}}\}$ be a set containing all the packing instances associated to the test set $\mathcal{D}_{\text{test}}$. Once the test phase is successfully completed, the trained FNN is now able to determine the optimal local weight vector $\bar{W}^i_l$ and the optimal global weight vectors $\bar{W}^i_g$, for each order $u^i_{\text{test}} \in \mathcal{U}_{\text{test}}$. By doing this, the HGA will run only once using the predicted local weight vector $\bar{W}^i_l$ and, once the solution set is generated, the optimal solution will be chosen using the predicted global weight vector $\bar{W}^i_g$.

4.3 Results

The above algorithm is coded in Python 3.7, running on a 3.6 GHz Intel Core i9 octa-core processor with 64 GB RAM. The set $\mathcal{U}$ of real packing instances is provided by an industrial company, as well as the box types and bin types databases. In order to fully exploit the processor power, $\mathcal{U}$ is split into eight subset to be simulated in parallel on each core.

4.3.1 HGA parameters

The HGA parameters are set as follows:

- Number of chromosomes = 30;
- Number of generations = 8;
- Number of chromosomes randomly selected in the selection process: $n_{rs} = 3$;

- Crossover probability:
  $P_c(g) = [0.3, 0.28, 0.25, 0.23, 0.2, 0.18, 0.15, 0.1]$;
- Mutation probability:
  $P_m(g) = [0.7, 0.65, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1]$;
- Local weight vectors: See Table 4.1.

TABLE 4.1: Local weight vectors.

j	$w_{j,1}$	$w_{j,2}$	$w_{j,3}$	$w_{j,4}$
1	1.0	0.7	0.5	0.9
2	1.0	1.0	0.1	0.1
3	0.9	0.9	0.3	0.1
4	0.8	0.6	0.9	0.7
5	0.9	0.2	0.8	0.5
6	0.5	0.1	0.4	0.5
7	0.8	0.7	0.6	0.5
8	0.4	0.4	0.6	1.0
9	0.6	0.8	0.2	0.8
10	0.2	0.2	1.0	1.0
11	0.2	0.4	0.2	0.3
12	0.8	0.7	0.7	1.0
13	0.9	0.7	0.4	0.4
14	1.0	0.5	0.2	0.3
15	0.9	0.6	0.1	0.5
16	1.0	1.0	0.0	0.1
17	0.2	0.2	0.0	1.0
18	0.8	0.7	0.0	0.5
19	0.5	0.5	0.0	0.5

4.3.2 FNN parameters

The FNN parameters are set as follows:

- Number of hidden layers: 1;
- Number of neurons in the input layer: 122 + bias;
- Number of neurons in the hidden layer: 500 + bias;
- Number of neurons in the output layer: 9;
- Activation function: ReLu;
- Loss function: Mean Squared Error;

- Stopping criterion: Early Stopping.

4.3.3 Input instances and graphical results

The set $\mathcal{U}$, containing real packing instances, is provided by an industrial company. The number of boxes of each instance can vary from 1 to 83 and, on average, 15 bin per hour should be arranged. Moreover, the frequency distribution of the different dimension box typologies of all instances is shown in Figure 4.1. The dataset $\mathcal{D}$, generated from $\mathcal{U}$, contains 675 samples. The related subsets, $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{valid}}$ and $\mathcal{D}_{\text{test}}$, contain 405, 135 and 135 samples, respectively.

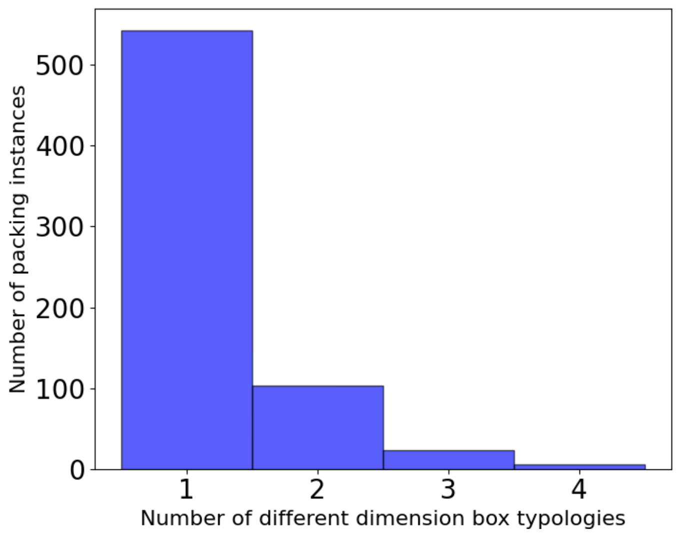


FIGURE 4.1: Frequency distribution of the different dimension box typologies of all packing instances.

4.3.3.1 Training and validation processes

The training process proves to be very efficient, resulting in a training time of 1.81 seconds. Moreover, it turns out to be robust with respect to the introduction of new typology of boxes in $\mathcal{B}$, that leads to an increment of the overall network complexity. The MSE evaluation on both $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{valid}}$ is reported in Figure 4.2, as function of the epochs number of the FNN training process. As expected, the error evaluated on $\mathcal{D}_{\text{train}}$ is a decreasing function, while the error evaluated on $\mathcal{D}_{\text{valid}}$, is decreasing until 17-th epoch and, then, it starts to increase, highlighting an overfitting phenomenon. The early stopping criterion will stop the training process once the 17-th epoch is reached.

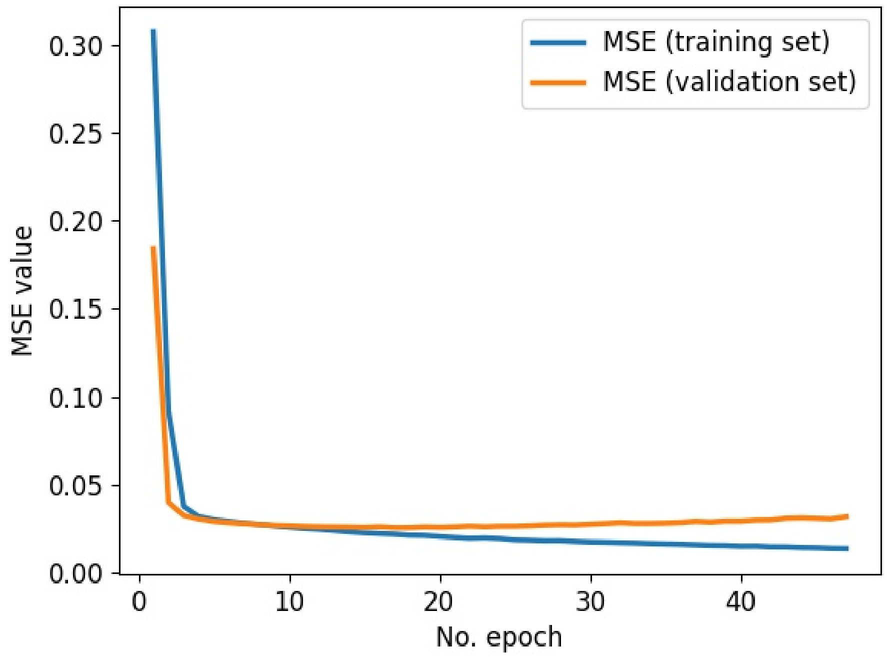


FIGURE 4.2: MSE evaluation on training set and validation set.

4.3.3.2 Test process

After the FNN training process is complete, it is possible to evaluate the MSE on the test set $\mathcal{D}_{\text{test}}$ and it results equal to $MSE_{\text{test}} = 0.0258$. The comparison between the target weight vector $y^i = [\bar{W}_l^i, \bar{W}_g^i]$ and the predicted weight vector $\hat{y}^i = [\hat{W}_l^i, \hat{W}_g^i]$ is reported in Table 4.2 and Table 4.3, for some relevant test instances. Notice that the target global weight vector is the same for each order in $\mathcal{U}$, meaning that it fits very well the type of packaging carried out by the company. The optimal solution of these instances is depicted in Figures 4.3a- 4.3f. They are found running the second stage of the algorithm directly on the relative predicted fitness weight vector. The red and blue box borders indicate, the fragility or non-fragility of the box, respectively. The related simulation time (S), the number of different dimension box typologies (M) and the total number of boxes (N) are reported in Table 4.4, while useful time simulation statistics, calculated on the whole subset $\mathcal{D}_{\text{test}}$, are shown in Table 4.5.

TABLE 4.2: Target vs. predicted local weight vector.

Order	Local weight vector	Type
1	[0.9, 0.2, 0.8, 0.5]	Target
1	[0.86, 0.18, 0.61, 0.55]	Predicted
2	[1.0, 0.7, 0.5, 0.9]	Target
2	[0.89, 0.63, 0.54, 0.86]	Predicted
3	[0.6, 0.8, 0.2, 0.8]	Target
3	[0.56, 0.71, 0.34, 0.71]	Predicted
4	[1.0, 0.5, 0.2, 0.3]	Target
4	[1.28, 0.57, 0.14, 0.26]	Predicted
5	[0.8, 0.7, 0.7, 0.1]	Target
5	[0.83, 0.75, 0.59, 0.86]	Predicted
6	[0.9, 0.9, 0.3, 0.1]	Target
6	[0.99, 0.82, 0.39, 0.17]	Predicted

TABLE 4.3: Target vs. predicted global weight vector.

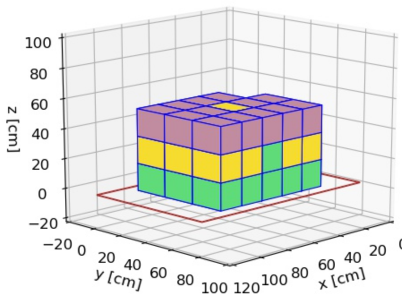
Order	Global weight vector	Type
1	[0.7, 0.5, 0.4, 0.4, 0.4]	Target
1	[0.68, 0.49, 0.42, 0.38, 0.38]	Predicted
2	[0.7, 0.5, 0.4, 0.4, 0.4]	Target
2	[0.69, 0.49, 0.39, 0.39, 0.39]	Predicted
3	[0.7, 0.5, 0.4, 0.4, 0.4]	Target
3	[0.71, 0.47, 0.41, 0.40, 0.38]	Predicted
4	[0.7, 0.5, 0.4, 0.4, 0.4]	Target
4	[0.67, 0.51, 0.37, 0.38, 0.38]	Predicted
5	[0.7, 0.5, 0.4, 0.4, 0.4]	Target
5	[0.73, 0.52, 0.39, 0.42, 0.43]	Predicted
6	[0.7, 0.5, 0.4, 0.4, 0.4]	Target
6	[0.81, 0.57, 0.46, 0.45, 0.45]	Predicted

TABLE 4.4: Simulation time (S), number of different dimension box typologies (M) and total number of boxes (N) reported for some test instances.

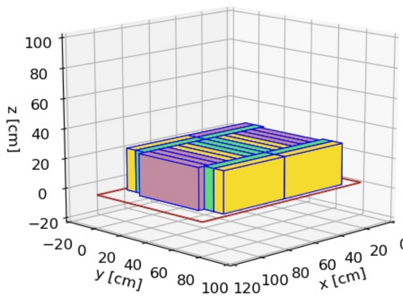
Order	M	N	S
1	1	44	1.529 min
2	1	21	0.187 min
3	2	28	13.183 min
4	2	27	15.190 min
5	2	50	14.540 min
6	2	27	14.797 min

TABLE 4.5: Time simulation statistics on subset $\mathcal{D}_{\text{test}}$.

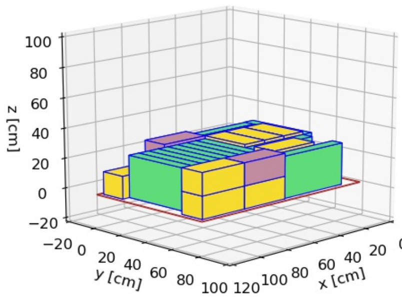
Mean value	1.654 min
Standard Deviation	4.176 min
Minimum value	0.004 min
Maximum value	20.632 min



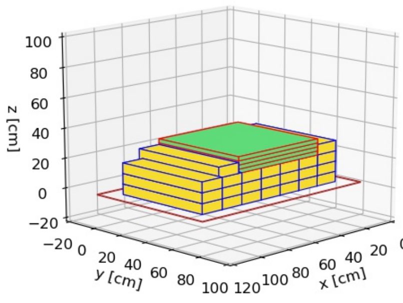
(A) Test Instance 1.



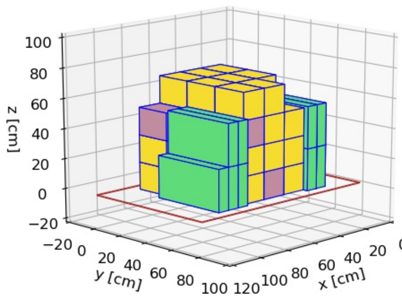
(B) Test Instance 2.



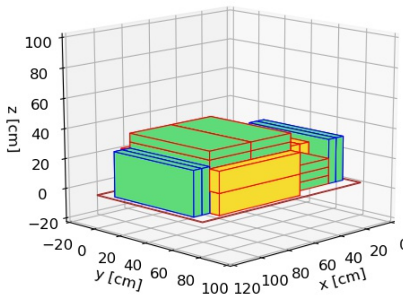
(C) Test Instance 3.



(D) Test Instance 4.



(E) Test Instance 5.



(F) Test Instance 6.

FIGURE 4.3: Best solutions found related to some relevant test instances.

Chapter 5

Conclusion and future works

5.0.1 Conclusion

In this relation, the Three-Dimensional Single-Bin-Size Bin Packing problem has been solved using an innovative and efficient algorithm. In Chapter 2, a literature review and a exhaustive classification about Cutting & Packing problems is presented, focusing on the Bin Packing problem and its dimensional variants. Continuing, a summary of genetic algorithm and feedforward neural networks is shown in Chapter 3, discussing how they can be adapted and combined to solve the 3D|BPP. Finally, In Chapter 4, the complete structure of the proposed algorithm is presented and simulation results are showed and discussed. In particular, a Hybrid Genetic Algorithm coupled with a Feedforward Neural Network has been used, taking into account geometric, stability and fragility constraints. One of the most important features of the new algorithm is its capability and efficiency to predict, for each input instance, the optimal weight vectors of all fitness functions that need to be optimized for solving the problem. Moreover, it is able to provide as output, the optimal boxes input sequence as well as the spatial coordinates of the the placed boxes vertices. The proposed algorithm proves to be very efficient and flexible for all the input instances as shown by the simulation results. Furthermore, the simulations demonstrate also that the system is easily adaptable to different users, thanks to an automatic adjustment of the fitness functions weights in order to avoid empirical tuning of the weights by the human operator.

5.0.2 Future works

The outcomes of the research work presented in this thesis open the possibility to several future improvements, taking as starting point the aquired stock of knowledge and the results obtained. Possible future works may involve the investigation of other machine learning techniques, in order to compare the various performance. Moreover, new practical constraints and new fitness functions could be added to the model, increasing the quality of the solutions even more.

Bibliography

[1] D. Vigo A. Lodi S. Martello. "Heuristic algorithms for the three-dimensional bin packing problem". In: *European Journal of Operational Research* 141 (2002), pp. 410–420.

[2] Mohamed Abdel-Basset et al. "An improved nature inspired meta-heuristic algorithm for 1-D bin packing problems". In: *Personal and Ubiquitous Computing* 22 (Oct. 2018). DOI: 10.1007/s00779-018-1132-7.

[3] Jaza Abdullah and Tarik Rashid. *Fitness Dependent Optimizer: Inspired by the Bee Swarming Reproductive Process*. Apr. 2019. DOI: 10.1109/ACCESS.2019.2907012.

[4] A. D. Almeida and M. B. Figueiredo. "A particular approach for the three-dimensional packing problem with additional constraints". In: *Computers & Operations Research* 37(11) (2010), pp. 1968–1976.

[5] M. Alonso et al. "Mathematical models for multicontainer loading problems". In: *Omega* 66 (2017), pp. 106–117.

[6] Gabriele Ancora, Gianluca Palli, and Claudio Melchiorri. "A Hybrid Genetic Algorithm for Pallet Loading in Real-World Applications". In: *IFAC-PapersOnLine* 53.2 (2020). 21th IFAC World Congress, pp. 10006–10010. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2020.12.2719. URL: https://www.sciencedirect.com/science/article/pii/S2405896320334820.

[7] Chazelle B. "The Bottomn-Left Bin-Packing Heuristic: An Efficient Implementation". In: *IEEE Transactions on Computers* C-32.8 (1983), pp. 697–707. DOI: 10.1109/TC.1983.1676307.

[8] J. E. Beasley. "Algorithms for Unconstrained Two-Dimensional Guillotine Cutting". In: *Journal of the Operational Research Society* 36.4 (1985), pp. 297–306. DOI: 10.1057/jors.1985.51. eprint: https://doi.org/10.1057/jors.1985.51. URL: https://doi.org/10.1057/jors.1985.51.

[9] J. O. Berkey and P. Y. Wang. "Two-Dimensional Finite Bin-Packing Algorithms". In: *The Journal of the Operational Research Society* 38.5 (1987), pp. 423–429. ISSN: 01605682, 14769360. URL: http://www.jstor.org/stable/2582731.

[10] E. E. Bischoff and M. S. W. Ratcliff. "Issues in the development of approaches to container loading". In: *Omega* 23(4) (1995), pp. 337–390.

[11] Eberhard E. Bischoff and Michael D. Marriott. "A comparative evaluation of heuristics for container loading". In: *European Journal of Operational Research* 44.2 (1990). Cutting and Packing, pp. 267–276. ISSN: 0377-2217. DOI: https://doi.org/10.1016/0377-2217(90)90362-F. URL: https://www.sciencedirect.com/science/article/pii/037722179090362F.

[12] E.E. Bischoff, F. Janetz, and M.S.W. Ratcliff. "Loading pallets with non-identical items". In: *European Journal of Operational Research* 84.3 (1995). Cutting and Packing, pp. 681–692. ISSN: 0377-2217. DOI: https://doi.org/10.1016/0377-2217(95)00031-K. URL: https://www.sciencedirect.com/science/article/pii/037722179500031K.

[13] A. Bortfeld and G. Wascher. "Constraints in container loading - A State-of-the-Art Review". In: *European Journal of Operational Research* 229 (2012), pp. 1–20.

[14] Marco Boschetti and Aristide Mingozzi. "The two-dimensional finite bin packing problem. Part I: New lower bounds for the oriented case". In: *Quarterly Journal of the Belgian, French and Italian Operations Research Societies* 1(1) (2003), pp. 27–42.

[15] Marco Boschetti and Aristide Mingozzi. "The two-dimensional finite bin packing problem. Part II: New lower and upper bounds". In: *Quarterly Journal of the Belgian, French and Italian Operations Research Societies* 1(2) (2003), pp. 135–147.

[16] José Carvalho. "LP models for bin packing and cutting stock problem". In: *European Journal of Operational Research* 141 (Sept. 2002), pp. 253–273. DOI: 10.1016/S0377-2217(02)00124-8.

[17] S. Ceschia and A. Schaerf. "Local search for a multi-drop multi-container loading problem". In: *Journal of Heuristics* 19(2) (2013), pp. 275–294.

[18] F.T.S. Chan et al. "Development of a decision support system for air-cargo pallets loading problem: A case study". In: *Expert Systems with Applications* 31 (2006), pp. 472–485.

[19] C. Chen, S. Lee, and Q. Shen. "An analytical model for the container loading problem". In: *European Journal of Operational Research* 80 (1995), pp. 68–76.

[20] J. Csirik and G. Woeginger. "On-line packing and covering problems". In: *Lecture Notes in Computer Science, Springer Berlin Heidelberg* 1442 (1998), pp. 144–177.

[21] Himanish Das and Pinki Roy. "A Deep Dive into Deep Learning Techniques for solving Spoken Language Identification Problems in Speech Signal processing". In: Dec. 2018, pp. 81–100. ISBN: 9780128181300. DOI: 10.1016/B978-0-12-818130-0.00005-2.

[22] Harald Dyckhoff. "A typology of cutting and packing problems". In: *European Journal of Operational Research* 44.2 (1990). Cutting and Packing, pp. 145–159. ISSN: 0377-2217. DOI: https://doi.org/10.1016/0377-2217(90)90350-K. URL: https://www.sciencedirect.com/science/article/pii/037722179090350K.

[23] Walaa El Ashmawi and Dr-Diaa Salama. "A modified squirrel search algorithm based on improved best fit heuristic and operator strategy for bin packing problem". In: *Applied Soft Computing* 82 (June 2019), p. 105565. DOI: 10.1016/j.asoc.2019.105565.

[24] M. Eley. "A bottleneck assignment approach to the multiple container loading problem". In: *OR Spectrum* 25(1) (2003), pp. 45–60.

[25] O. Faroe, D. Pisinger, and M. Zachariasen. "Guided local search for the three-dimensional bin packing problem". In: *Department of Computer Science, University of Copenhagen* (2003).

[26] Chung F.K.R., Garey M.R., and Johnson D.S. "On packing two-dimensional bins." In: *SIAM Journal on Algebraic Discrete Methods* 3(1) (1982), pp. 66–76.

[27] International Transport Forum. "Permissible maximum weights of lorries in europe". In: (2016).

[28] H. J. Fraser and J. A. George. "Integrated container loading software for pulp and paper industry". In: *European Journal of Operational Research* 77(3) (1994), pp. 466–474.

[29] Hans Frenk and Gábor Galambos. "Hybrid next-fit algorithm for the two-dimensional rectangle bin-packing problem". In: *Computing* 39 (Sept. 1987), pp. 201–217. DOI: 10.1007/BF02309555.

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

[31] Michel Gendreau. "Metaheuristics in combinatorial optimization". In: *Annals of Operations Research* 140 (2005), pp. 189–213. ISSN: 1572-9338. DOI: https://doi.org/10.1007/s10479-005-3971-7.

[32] M. Iori, S. Martello, and M. Monaci. "Metaheuristic algorithms for the strip packing problem". In: *Applied Optimization* 78 (2003), pp. 159–180.

[33] Mohit Jain, Vijander Singh, and Asha Rani. "A novel nature-inspired algorithm for optimization: Squirrel search algorithm". In: *Swarm and Evolutionary Computation* 44 (2019), pp. 148–175. ISSN: 2210-6502. DOI: https://doi.org/10.1016/j.swevo.2018.02.013. URL: https://www.sciencedirect.com/science/article/pii/S2210650217305229.

[34] Z. Jin, T. Ito, and K. Ohna. "A three-dimensional bin packing problem and its practical algorithm". In: *JSME International Journal Series C: Mechanical Systems, Machine Elements and Manifacturing* 46 (2003), pp. 60–66.

[35] L. Junqueira, R. Morabito, and D. S. Yamashita. "Three-dimensional container loading models with cargo stability and load bearing constraints". In: *Computers and Operations Research* 39 (2012), pp. 74–85.

[36] Leonardo Junqueira et al. "Optimization Models for the Three-Dimensional Container Loading Problem with Practical Constraints". In: *Modeling and Optimization in Space Engineering*. Ed. by Giorgio Fasano and János D. Pintér. Springer Optimization and Its Applications. Springer, 2012. Chap. 0, pp. 271–293. DOI: 10.1007/978-1-4614-4469-5. URL: https://ideas.repec.org/h/spr/spochp/978-1-4614-4469-5_12.html.

[37] L. V. Kantorovich. "Mathematical Methods of Organizing and Planning Production". In: *Management Science* 6.4 (1960), pp. 366–422. DOI: 10.1287/mnsc.6.4.366. URL: https://doi.org/10.1287%2Fmnsc.6.4.366.

[38] J. L. Lin, C. H. Chang, and J. Y. Yang. "A study of optimal system for multiple-constraint multiple-container packing problems". In: *Lecture Notes in Computer Science, Springer Berlin Heidelberg* 4031 (2006), pp. 1200–1210.

[39] D.S. Liu et al. “On solving multi-objective bin packing problems using evolutionary particle swarm optimization”. In: *European Journal of Operational Research* 190(2) (2008), pp. 357–382.

[40] A. Lodi, S. Martello, and D. Vigo. “Approximation algorithms for the oriented two-dimensional bin packing problem”. In: *European Journal of Operational Research* 112 (1999), pp. 158–166.

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

[42] A. Lodi, S. Martello, and D. Vigo. “Neighborhood search algorithm for the guillotine non-oriented two-dimensional bin packing problem”. In: *Meta-Heuristics, Springer US* 11 (1999), pp. 125–139.

[43] A. Lodi, S. Martello, and D. Vigo. “Tspack: A unified tabu search code for multi-dimensional bin packing problems”. In: *Annals of Operations Research* 131 (2004), pp. 203–213.

[44] Christoph von der Malsburg. “Frank Rosenblatt: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms”. In: *Brain Theory* (Jan. 1986), pp. 245–248. DOI: 10.1007/978-3-642-70911-1_20.

[45] S. Martello and P. Toth. “Knapsack Problems - Algorithms and computer implementations”. In: *John Wiley & Sons, Chichester, UK* (1990).

[46] Silvano Martello, David Pisinger, and Daniele Vigo. “The Three-Dimensional Bin Packing Problem”. In: *Operations Research* 48 (Feb. 1998). DOI: 10.1287/opre.48.2.256.12386.

[47] Silvano Martello and Daniele Vigo. “Exact Solution of the Two-Dimensional Finite Bin Packing Problem”. In: *Management Science* 44.3 (1998), pp. 388–399. ISSN: 00251909, 15265501. URL: http://www.jstor.org/stable/2634676.

[48] Seyedali Mirjalili and Andrew Lewis. “The Whale Optimization Algorithm”. In: *Advances in Engineering Software* 95 (2016), pp. 51–67. ISSN: 0965-9978. DOI: https://doi.org/10.1016/j.advengsoft.2016.01.008. URL: https://www.sciencedirect.com/science/article/pii/S0965997816300163.

[49] M. Monaci and P. Toth. “A set-covering based heuristic approach for bin-packing problems”. In: *INFORMS Journal on Computing* 18 (2006), pp. 71–85.

[50] I. Moon and T. Nguyen. “Container packing problem with balance constraints”. In: *OR Spectrum* 36 (2013), pp. 837–878.

[51] N. Morgan and H. Bourlard. “Generalization and Parameter Estimation in Feedforward Nets: Some Experiments”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 2. Morgan-Kaufmann, 1990. URL: https://proceedings.neurips.cc/paper/1989/file/63923f49e5241343aa7acb6a06a751e7-Paper.pdf.

[52] Chanaleä Munien and Absalom E. Ezugwu. “Metaheuristic algorithms for one-dimensional bin-packing problems: A survey of recent advances and applications”. In: *Journal of Intelligent Systems* 30.1 (2021), pp. 636–663. DOI: doi:10.1515/jisys-2020-0117. URL: https://doi.org/10.1515/jisys-2020-0117.

[53] Tugrul Nayraktar. “A memory-integrated artificial bee algorithm for heuristic optimisation”. In: *University of Bedfordshire* (2014). DOI: http://hdl.handle.net/10547/332794.

[54] Chigozie Nwankpa et al. “Activation Functions: Comparison of Trends in Practice and Research for Deep Learning”. In: Dec. 2020.

[55] C. Paquay, Michael Schyns, and Sabine Limbourg. “A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application”. In: *International Transactions in Operational Research* 23 (July 2014). DOI: 10.1111/itor.12111.

[56] H. Pollaris et al. “Vehicle routing problems with loading constraints: state-of-the-art and future directions”. In: *OR Spectrum* 37(2) (2015), pp. 297–330.

[57] David E. Rumelhart and James L. McClelland. “Learning Internal Representations by Error Propagation”. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. 1987, pp. 318–362.

[58] Dr-Diaa Salama et al. “An Adaptive Fitness-Dependent Optimizer for the One-Dimensional Bin Packing Problem”. In: *IEEE Access* (Apr. 2020), pp. 97959 –97974. DOI: 10.1109/ACCESS.2020.2985752.

[59] A. Soke and Z. Bingul. “Hybrid genetic algorithm and simulated annealing for two-dimensional non-guillotine rectangular packing problems”. In: *Engineering Applications of Artificial Intelligence* 19(5) (2006), pp. 557–567.

[60] J. Terno et al. “An efficient approach for the multi-pallet loading problem”. In: *European Journal of Operational Research* 123 (2000), pp. 372–281.

[61] A. Trivella and D. Pisinger. “The load-balanced multi-dimensional bin-packing problem”. In: *Computers & Operations Research* 74 (2016), pp. 152–164.

[62] R. Tsai, E. Malstrom, and W. Kuo. “Three dimensional palletization of mixed box sizes”. In: *IIE Transactions* 25(4) (1993), pp. 64–75.

[63] Yong Wu et al. “Three-dimensional bin packing problem with variable bin height”. In: *European Journal of Operational Research* 202.2 (2010), pp. 347–355. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2009.05.040. URL: https://www.sciencedirect.com/science/article/pii/S0377221709003919.

[64] Gerhard Wäscher, Heike Haußner, and Holger Schumann. “An improved typology of cutting and packing problems”. In: *European Journal of Operational Research* 183.3 (2007), pp. 1109–1130. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2005.12.047. URL: https://www.sciencedirect.com/science/article/pii/S037722170600292X.