Packing Problems in Industrial Environments: Application to the Expedition Problem at INDASA

JOSÉ FILIPE FREIRE BEIRÃO

Dissertação para obtenção do Grau de Mestre em
Engenharia Mecânica

Júri

Presidente: Prof. João Rogério Caldas Pinto
Orientador: Prof. João Miguel da Costa Sousa
Co-Orientador: Prof. Carlos Augusto Santos Silva
Vogal: Eng.º Rui Lopes dos Santos
Vogal: Prof. Alexandra Bento Moutinho

Outubro 2009
“Every great advance in science has issued from a new audacity of imagination.”

John Dewey

Este trabalho reflecte as ideias dos seus autores que, eventualmente, poderão não coincidir com as do Instituto Superior Técnico.
Abstract

This work proposes a tool to solve a specific problem related to the organization of expedition containers at INDASA. This tool aids the packing and picking at the company and by doing so, avoids unnecessary repacking and waiting time of products in line be packed.

To do so it was necessary to simulate the packing of the products. Different types of packing problems have been studied in order to decide the one that is better applied to the real problem. It was noticed that the problem could be divided into a Bin Packing Problem, and a Knapsack Problem. The first is related to the pallet loading with the products and the second is the responsible for the container loading with the pallets.

In order to solve these two problems, three different algorithms are proposed in this thesis: a Layer Building algorithm, an Extreme Points First Fit Decreasing algorithm and an Ant Colony Optimization applied to Extreme Points. These algorithms were able to meet the requirements and the last one allows solving both the Bin Packing Problem and the Knapsack Problem.

Hence, for the two INDASA instances it was possible reduce the number of necessary pallets from twenty six to twenty three in the first instance and twenty four to twenty in the second. As for one of the Bin Packing Problem instances, better results than the other algorithms were achieved. For the remaining benchmark instances similar results to the ones found in the literature were obtained.
Resumo

Este trabalho propõe uma ferramenta com o objectivo de optimizar a arrumação de produtos em paletes e estas em contentores. Esta ferramenta auxilia as tarefas de arrumação e recolha de produtos na empresa, evitando assim voltar a arrumar e períodos de espera dos produtos a arrumar.

De modo a conseguir este objectivo foi necessário simular a arrumação dos produtos. Este trabalho considerou diferentes tipos de problemas de modo a escolher o que melhor se adapta ao problema real. Chegou-se à conclusão que o problema podia ser dividido em dois, um problema de “Bin Packing” e um problema de “Knapsack”. O primeiro está relacionado com a arrumação dos produtos nas paletes, e o segundo das paletes nos contentores.

Assim nesta tese são propostos três algoritmos diferentes para resolver estes dois problemas: um algoritmo de “Layer Building”, um algoritmo de “Extreme Points First Fit Decreasing” e um algoritmo de “Ant Colony Optimization” aplicado a “Extreme Points”. Estes algoritmos conseguiram obedecer aos requisitos impostos, onde o último tem a capacidade de resolver os problemas de “Bin Packing” e “Knapsack”.

Os algoritmos desenvolvidos conseguiram resolver as duas instâncias fornecidas pela INDASA sendo possível reduzir o número de paletes de vinte e seis para vinte e três no primeiro caso, e de vinte e quatro para vinte e uma no segundo caso. Foi possível obter melhores resultados do que os obtidos pelos restantes algoritmos, presente na literatura, para uma das instâncias de “Bin Packing”. Para as restantes instâncias de “benchmark” foi possível obter resultados semelhantes.
Keywords:

- Bin Packing Problems
- Container Loading Problems
- Optimization
- Metaheuristics
- Hybrid optimization algorithms

Palavras chave:

- Bin Packing Problems
- Container Loading Problems
- Optimização
- Metaheurística
- Algoritmos híbridos de optimização
Acknowledgements

I would like to express my gratitude to Professor João Sousa for giving me the opportunity to do this thesis. Also thank the chance to discuss ideas, thus clarifying the path to follow.

To Prof. Carlos Silva I thank the help and optimism shown during this endeavor, also, the support and the availability for discussion.

To Eng. Rui Lopes dos Santos I would like to thank the explanation concerning the INDASA general problem. As well, thanking the swiftness in providing extra data whenever necessary.

All my colleagues at the GCAR I would like to thank the support during the endless hours spent at the lab, and their help in writing this thesis.

I am grateful to my friends for the hours spent listening to me talking about bins and boxes, and the help provided when it was needed.

I would also like to thank my family for the support shown whenever necessary.

Finally, and most important, the deepest appreciation to my Parents, they unconditionally helped me during these long years. To them I dedicate this thesis, and to them I simply say Obrigado.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>I</td>
</tr>
<tr>
<td>RESUMO</td>
<td>III</td>
</tr>
<tr>
<td>KEYWORDS:</td>
<td>V</td>
</tr>
<tr>
<td>PALAVRAS CHAVE:</td>
<td>V</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>VII</td>
</tr>
<tr>
<td>CONTENTS</td>
<td>IX</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>XIII</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>XV</td>
</tr>
</tbody>
</table>

## 1 INTRODUCTION

1.1 LOGISTICS | 1 |
1.2 EXPEDITION OPTIMIZATION IN INDASA | 3 |
1.3 APPLYING PACKING PROBLEMS | 4 |
1.4 OBJECTIVES AND OUTLINE OF THIS WORK | 4 |
1.5 WORK CONTRIBUTION | 5 |

## 2 PACKING PROBLEMS

2.1 PACKING PROBLEM DESCRIPTION | 7 |
2.1.1 Distinction between container and pallet | 10 |
2.1.2 Different types of packing problems | 10 |
2.1.3 Solving the packing problem | 11 |
2.2 PLACE REPRESENTATION OF THE BOXES INSIDE THE BIN | 12 |
2.3 CONSTRUCTIVE HEURISTICS | 15 |
2.4 STATE OF THE ART ALGORITHMS | 16 |
2.4.1 Bin packing problems | 16 |
2.4.1.1 Least waste first for the 2DBPP | 16 |
2.4.1.2 Guided local search for the 3DBPP | 17 |
2.4.1.3 Exact branch-and-bound for the 3DBPP | 17 |
2.4.1.4 Tabu-search for the 3DBPP | 17 |
2.4.1.5 Multiobjective in 2DBPP using swarm optimization ...........................................18
2.4.1.6 Extreme points based heuristic for 3DBPP ..........................................................18
2.4.2 Container loading problems ..................................................................................18
  2.4.2.1 Heuristics based on wall building algorithm ......................................................19
  2.4.2.2 Hybrid genetic algorithm for the CLP ...............................................................19
  2.4.2.3 Parallel algorithms to solve the CLP .................................................................20
  2.4.2.4 A tertiary-tree-based dynamic space decomposition approach .........................20

3 THE PACKING PROBLEM AT INDASA ........................................................................21
  3.1 DESCRIPTION OF THE PRODUCTS .......................................................................21
    3.1.1 Abrasive composition ..........................................................................................21
    3.1.2 Production and composition process .................................................................22
    3.1.3 Different types of products ..................................................................................23
  3.2 EXPEDITION LIST STRUCTURE ..............................................................................24
  3.3 PROBLEM DESCRIPTION .........................................................................................25
    3.3.1 Problem constraints ............................................................................................26
    3.3.2 Data processing ..................................................................................................27
  3.4 SOLVING THE INDASA EXPEDITION PROBLEM ..................................................29
    3.4.1 Pallet loading .....................................................................................................29
    3.4.2 Container loading ...............................................................................................29
    3.4.3 Packing process ..................................................................................................30

4 PROPOSED HEURISTIC ALGORITHMS .......................................................................33
  4.1.1 Orientation selection .............................................................................................34
  4.2 LAYER BUILDING ALGORITHM ..............................................................................34
    4.2.1 Filling multiple bins ............................................................................................37
  4.3 ADAPTED EXTREME POINTS ALGORITHM ............................................................38
    4.3.1 Extreme point definition and selection .................................................................39
    4.3.2 Extreme points available area .............................................................................40
    4.3.3 Packing sets of boxes ..........................................................................................42
    4.3.4 Stability constraint ..............................................................................................43
    4.3.5 Filling multiple bins ............................................................................................43

5 ANT COLONY OPTIMIZATION APPLIED TO PACKING PROBLEMS .........................45
  5.1 APPLYING THE ANT COLONY OPTIMIZATION TO THE PACKING PROBLEM ........46
    5.1.1 The zero node .....................................................................................................47
    5.1.2 The pheromone trail ...........................................................................................47
    5.1.3 The heuristic matrix ...........................................................................................48
  5.2 INTRODUCING THE CONSTRAINTS .......................................................................49
  5.3 THE ANT COLONY ALGORITHM APPLIED TO THE PACKING PROBLEMS ............50
APPENDIX A    TOOL DEVELOPED TO SOLVE THE INDASA PROBLEM - USER MANUAL ................. 85

A.1  MENU_INICIAL .................................................................................................................. 85
A.2  LISTA_ENCOMENDAS ....................................................................................................... 87
A.3  LISTA_PALETE .................................................................................................................. 88
A.4  RUNNING THE TOOL ......................................................................................................... 89
List of tables

Table 3.1 - Boxes and Bins Assignment ........................................................................................................... 29
Table 5.1 - ACO Example .................................................................................................................................. 46
Table 6.1 - Container Loading Problem Instances [Ratcliff, n.d.] ................................................................. 55
Table 6.2 - Container and pallets dimensions .................................................................................................. 55
Table 6.3 - INDASAUSA Instances .................................................................................................................. 56
Table 6.4 - INDASAUSA2 Instance .................................................................................................................. 56
Table 6.5 - INDASA Results ............................................................................................................................ 56
Table 6.6 - Parameters tuning, step 1 - Class I, 50 boxes .................................................................................... 58
Table 6.7 - Parameters tuning, step 2 - Class I, 50 boxes ................................................................................... 58
Table 6.8 - Parameters tuning, step 3 - Class I, 50 boxes ................................................................................... 59
Table 6.9 - Parameters tuning, step 4 - Class I, 50 boxes ................................................................................... 60
Table 6.10 - Parameters tuning, step 5 - Class I, 50 boxes ............................................................................... 61
Table 6.11 - Parameters tuning - Class VI, 50 boxes ......................................................................................... 62
Table 6.12 - Parameters tuning - Class I, 200 boxes ......................................................................................... 63
Table 6.13 - Parameters tuning - Class VI, 200 boxes ...................................................................................... 63
Table 6.14 - Ant Colony Optimization tuned parameters .................................................................................. 63
Table 6.15 - Number of bins necessary for each instance of Class I with 50 boxes .......................................... 64
Table 6.16 - Number of bins necessary for each instance of Class I with 200 boxes ....................................... 64
Table 6.17 - Number of bins necessary for each instance of Class VI with 50 boxes ....................................... 65
Table 6.18 - Number of bins necessary for each instance of Class VI with 200 boxes ..................................... 66
Table 6.19 - State of the art versus ACO-EP .................................................................................................. 66
Table 6.20 - Container Loading Problem results with stability constraint ...................................................... 68
Table 6.21 - Container Loading Problem results with stability constraint ...................................................... 69
Table 6.22 - Container Loading Problem results without stability constraint ................................................ 69
Table 6.23 State of the art versus ACO-EP ......................................................................................................... 70
Table 6.24 - Layer Building Heuristic results .................................................................................................. 71
Table 6.25 - Layer Building Heuristic results without agglomeration constraint ............................................ 71
Table 6.26 - Extreme Points Heuristic results ................................................................................................ 72
Table 6.27 Ant Colony Optimization - Extreme Points result .......................................................................... 74
Table 6.28 - Time comparison .......................................................................................................................... 75
## List of figures

**FIGURE 1.1** - LOGISTICS RESPONSIBILITIES ........................................................................................................... 2

**FIGURE 2.1** - BIN .................................................................................................................................................... 7

**FIGURE 2.2** - Box ...................................................................................................................................................... 7

**FIGURE 2.3** - STRONGLY HETEROGENEOUS ...................................................................................................... 9

**FIGURE 2.4** - WEAKLY HETEROGENEOUS ............................................................................................................ 9

**FIGURE 2.5** - HOMOGENEOUS ............................................................................................................................... 9

**FIGURE 2.6** - NON GUILLOTINABLE .................................................................................................................... 9

**FIGURE 2.7** - GUILLOTINABLE ............................................................................................................................. 9

**FIGURE 2.8** - CONTAINER ....................................................................................................................................... 10

**FIGURE 2.9** - PALLET ................................................................................................................................................. 10

**FIGURE 2.10** - PACKING PROBLEM SOLUTION ................................................................................................ 12

**FIGURE 2.11** - SPACE DISCRETIZATION REPRESENTATION .................................................................................... 12

**FIGURE 2.12** - CORNER POINTS REPRESENTATION ............................................................................................... 13

**FIGURE 2.13** - EXTREME POINTS REPRESENTATION ............................................................................................ 14

**FIGURE 2.14** - GRAPH REPRESENTATION ........................................................................................................ 14

**FIGURE 3.1** - ABRASIVE COMPOSITION ................................................................................................................ 22

**FIGURE 3.2** - PRODUCTION PROCESS .................................................................................................................. 22

**FIGURE 3.3** - TRANSFORMATION PROCESS ......................................................................................................... 23

**FIGURE 3.4** - FORM TYPES ................................................................................................................................... 24

**FIGURE 3.5** - EXPEDITION LIST STRUCTURE ..................................................................................................... 24

**FIGURE 3.6** - GENERIC ALGORITHM ................................................................................................................... 26

**FIGURE 3.7** - MAIN PROCESS ............................................................................................................................... 28

**FIGURE 3.8** - WEIGHT AND DIMENSIONS ATTRIBUTION ...................................................................................... 28

**FIGURE 3.9** - PACKING PROCESS .......................................................................................................................... 31

**FIGURE 4.1** - LAYER SCHEME ............................................................................................................................. 34

**FIGURE 4.2** - ROW SCHEME ........................................................................................................................................ 34

**FIGURE 4.3** - SINGLE BIN FILLING ........................................................................................................................ 35

**FIGURE 4.4** - LAYER FILLING ALGORITHM ........................................................................................................... 36

**FIGURE 4.5** - MULTIPLE BINS FILLING ................................................................................................................ 37

**FIGURE 4.6** - SINGLE BIN FILLING ........................................................................................................................ 39
FIGURE 4.7 - EXTREME POINTS PROJECTION .......................................................................................... 40
FIGURE 4.8 - EXTREME POINTS AVAILABLE AREA ........................................................................ 41
FIGURE 4.9 - EXAMPLES OF AVAILABLE AREAS ........................................................................... 42
FIGURE 4.10 - EXTREME POINTS AVAILABLE AREA DECOMPOSITION ........................................ 42
FIGURE 4.11 - MULTIPLE BINS FILLING USING FFD..................................................................... 44
FIGURE 5.1 - UNCONSTRAINED EXAMPLE......................................................................................... 46
FIGURE 5.2 - ZERO NODE INSERTION ............................................................................................. 47
FIGURE 5.3 - CONSTRAINED EXAMPLE ............................................................................................ 49
FIGURE 5.4 - ANT COLONY OPTIMIZATION APPLIED TO PACKING ALGORITHMS .................... 52
FIGURE 6.1 PARAMETERS TUNING, STEP 2 - CLASS I, 50 BOXES .................................................. 59
FIGURE 6.2 - PARAMETERS TUNING, STEP 3 - CLASS I, 50 BOXES ................................................ 60
FIGURE 6.3 - PARAMETERS TUNING, STEP 4 - CLASS I, 50 BOXES ................................................ 61
FIGURE 6.4 - COST FUNCTION EVOLUTION FOR CLASS I WITH 200 BOXES INSTANCES ........... 65
FIGURE 6.5 - STATE OF THE ART VERSUS ACO-EP ..................................................................... 67
FIGURE 6.6 - INSTANCE LN2 EVOLUTION WITH 10% EVAPORATION COEFFICIENT ................. 68
FIGURE 6.7 - LAYER BUILDING ALGORITHMS RESULTS ............................................................... 72
FIGURE 6.8 - EXTREME POINTS FIRST FIT DECREASING HEURISTIC ........................................... 73
FIGURE 6.9 - METHODS COMPARISON ........................................................................................... 74
FIGURE A.1 – MENU_INICIAL .......................................................................................................... 86
FIGURE A.2 – LISTA_ENCOMENDAS ................................................................................................. 87
FIGURE A.3 – LISTA_PALETE ........................................................................................................... 88
FIGURE A.4 – DIMENSOES .............................................................................................................. 89
FIGURE A.5 – PALETES ..................................................................................................................... 90
FIGURE A.6 – DIMENŠOES ............................................................................................................. 90
FIGURE A.7 – PERGUNTA ............................................................................................................... 90
1 Introduction

In shop floor/factories, the optimization of the space available for products expedition, being in the form of containers or pallets, is an important issue. There may be different parameters to optimize, e.g. the percentage of volume occupied, the total profit obtained or the weight distribution. The problem is an important issue for company logistics, as the optimization of the number of products packed leads to a direct decrease in warehousing and expedition costs. Also most of the references in the literature concern small instance problems. In order to solve problems with large instances, which represent most of the practical problems, it is necessary to develop or adapt the existing solutions.

Since logistics is so important in a factory, any tool that helps to improve it is of great help. In this case INDASA needed this specific tool concerning the expedition problem optimization. Thus, it was necessary to construct a model that could simulate packing of the necessary orders. The solution found to construct this model was to study the packing problems and necessary adaptations to this real case.

1.1 Logistics

Logistics management is defined by the Council of Logistic Management [CSCMP, n.d.] as “the process of planning, implementing and controlling the efficient, effective flow and storage of goods, services and related information from point of origin to point of consumption for the purpose of conforming to customer requirements”. As a study area, it became important in the beginning of the XXth century with the distribution of horticultural goods [Lambert et al., 1998].

Logistics have been used, however, since ancient times by the military leaders. As the wars were distant and long, large and constant shift of resources were necessary. To transport the troops, armory and goods to the war field to plan, organize and execute logistic tasks were necessary. Thus the closest route was not always the shortest, as it was necessary to have a near water source, transportation, warehousing and distribution of equipment and supplies. In ancient Greece, Rome or Byzantine Empire the militaries with the title of “Logistikas” were the responsible for assuring the supplies and resources for war [Carvalho, 2005].
The contribution of logistics toward the Allied victory in World War II was an important factor. Logistics began then receiving increasingly recognition and emphasis [Pagonis, 1992]. The first dedicated logistic texts began to appear in the early 1960s, which is also the time when Peter Ducker stated that “logistics was one of the last real frontiers of opportunity for organizations wishing to improve their corporate efficiency” [Ducker, 1962].

There has been a proliferation of technological developments in areas that support logistics. Technology is having a profound effect on the way that logistics personnel interface with other functional areas, creating the ability to access more timely accurate information. For instance, combining information technology with automated warehousing reduces inherent human variability, creating an opportunity to improve customer service [Lambert et al., 1998].

In Figure 1.1 [Lambert et al., 1998] it is possible to observe the different functions that compose logistics and the percentage of time involved and effort in each of those activities. This data is based on the Logistic Career Patterns Study conducted annually by The Ohio State University. From this picture it is possible to understand the complexity involving logistics. As it can be seen, inventory control and warehousing are activities that require a lot of time and effort, and any optimization in these activities, can have a great impact on the overall logistics.
1.2 Expedition optimization in INDASA

Founded in 1979, INDASA is actually one of the European leaders in the production of abrasives, for many applications, e.g. car repainting or construction. INDASA has done several investments in cutting-edge technology along the years, and it is today one of the most modern manufacturers in Europe. Nevertheless, INDASA managers recognize areas where further improvements can be achieved. One of these areas is the orders expedition.

Currently, INDASA does not use any software tool to optimize the expedition problem. This leads to a poor utilization of the container space, and sometimes it is necessary to the reload the orders so as to make a better use of the available space. The current available knowledge is purely empirical and based in previous attempts to pack the products. The ordered products are stored first inside pallets, and then, those loaded pallets are packed inside the expedition containers. The main problem is the packing of products into pallets which is more complex therefore leading to poor utilization of the space. Hence the poor utilization of the container space is a direct consequence of the poor utilization of the pallet space.

In this work a tool was developed which gives an idea of the number of containers/pallets necessary to fulfill a given set of orders and arrange those orders. This allows INDASA to choose the orders which are to be dispatched if they do not fit into a given container before the loading is started.

The output of the tool is a list of information relating containers, pallets and different abrasives. It is possible to know which pallets are to go in which container and all their information, such as weight, both liquid and gross, and the pallets position in the container. It is also possible to know the abrasives that are stored in each pallet, knowing their exact position inside the pallet and relevant data for expedition.

The developed tool allows a previous negotiation with the client, based on the previous knowledge of the products that are possible to dispatch. This may be a way of preventing fines for not dispatching products for lack of space.

This tool becomes a powerful aid in two complementary fields: the picking and the packing. The picking is done by previously knowing which products are packed first. This knowledge allows to pick the products from the warehouse orderly, thus prevents having a large area with waiting products. The efficiency of the packing is therefore increased, instead of having a great number of products waiting to be packed, and sometimes repacked; there is available information about the destination pallet.

Therefore this tool makes it possible to transform a given set of orders into an expedition list with all the information relevant to all the activities that are in between the warehouse and dispatching the required abrasives.
1.3 Applying packing problems

In order to develop the tool for INDASA, it was necessary to develop a model that allows simulating the arrangement of the products inside a pallet, and of those pallets inside the container. To do so it was necessary to use packing problems. Packing problems comprehend the exercise of storing boxes inside of bins. By doing so, the packing problems can have many goals, such as optimizing the minimum number of necessary bins to pack a given set of boxes, maximize the volume of boxes that can fit into a given number of bins or the minimal dimension of a bin in order to store a set of boxes, etc.

As this problem asks for packing a diversity of three dimensional boxes, it became apparent that the packing problem to be implemented must be a three dimensional problem, which is the most complex exercise related to packing problems.

Packing problems are a good alternative to solve this practical problem becoming the foundation of this model. In this way a tool to deal with large instances that may be applied to other similar problems was developed.

1.4 Objectives and outline of this work

The main objective of this thesis is to optimize the expedition problem at INDASA. This is done by creating a model that simulates the packing process and by doing so, can give an idea of what are the order lines available for expedition. However, the packing problems found in literature do not satisfy the constraints imposed by INDASA. Thus it becomes necessary to construct or adapt the existing problems. Since the INDASA problem consists of two different packing problems, the container loading and the pallet loading, the necessary algorithms must adapt to this reality. Given that the algorithms are able to solve different problems, they were tested in benchmark instances to compare to state of the art results.

Based on different packing problem approaches, it was built a model that could simulate the packing of products into pallets and then the pallets into containers. As there are packing problems involved to develop this tool, these problems must be tested with benchmark instances related to the specific problems.

The steps that were taken to achieve this objective are described in the following chapters:

In Chapter 2, a vision of what are packing algorithms, the definition and constraints is presented. The steps which allow a packing algorithm to run are explained and the state of art to these problems is presented.

Chapter 3 describes the expedition problem of INDASA, and the main abrasive products that the company sells. Since the main goal is to find a model that satisfies the expedition process of INDASA, it is presented the packing problem adapted to this case. The restrictions and necessary data to solve the given
problem are presented, by doing so the INDASA problem is formulated. Also it is explained the way the packing problems are adapted to this specific problem.

In chapter 4, two different heuristics are described, and the algorithms behind them are explained. The first one, a layer building heuristic, the second a heuristic based on Extreme Points. Both heuristics are adapted to large instances due to the nature of the problem.

In chapter 5, it is explored the possibility of applying Ant Colony Optimization to the Extreme Points. Because the resemblances between the different packing problems, it is presented an algorithm that can solve both Knapsack Problems and Bin Packing Problems.

In chapter 6, the different developed algorithms are tested with two INDASA instances, and the Ant Colony Optimization is also compared with the state of the art using Benchmark instances.

Finally, in chapter 7, the conclusions are drawn, and suggestions for future work are presented.

In Appendix A, it is presented an user manual for the developed tool for INDASA. The different menus are explained and the functioning of the tool is described.

1.5 Work Contribution

In order to solve the proposed problem, three different algorithms for packing problems were developed to solve large instance packing problems. Also they were adapted so that they could respect the constraints imposed by INDASA. The proposed algorithms are the following:

- A Layer Building Algorithm based in the Wall Building Algorithm idea;
- An Extreme Points using First Fit Decreasing Heuristic adapted to large instances;
- An Ant Colony Optimization using Extreme Points flexible so it could solve both the Knapsack Problem and the Bin Packing Problem.

Having these algorithms built and tested it was constructed the software tool that can optimize the expedition of containers by simulating the packing process, and therefore create a viable expedition list.

Additionally the Ant Colony Optimization applied to Extreme Points was tested in Benchmark instances for both the Bin Packing Problem and the Container Loading Problem, as a specification of the Knapsack Problem. For both the problems the results provided viable solutions when compared to state of the art algorithms, even if the algorithm was not built specifically to solve those instances.

The paper is going to be submitted to the 2010 IEEE World Congress on Computational Intelligence (IEEE WCCI 2010), to the conference: 2010 IEEE Congress on Evolutionary Computation (IEEE CEC 2010).
2 Packing problems

Packing problems are known to be NP-hard, non-deterministic polynomial-time hard hence it is not known any algorithm that solves the problem in polynomial time, [Scheithauer, 1991]. In general these problems are too large and complex to be solved exactly and heuristics become the only viable algorithms to find a solution. Thus it becomes important to study the different packing problems in order to choose the one that better adapts to one given problem.

2.1 Packing problem description

In this study the objective is to deal with three dimensional packing problems, which ask for an orthogonal packing of a given set of rectangular shaped boxes into three dimensional rectangular bins. Each box, see Figure 2.2, j (j=1,…,n) is characterized by a width $w_j$, depth $d_j$ and height $h_j$. The three dimensional bins, see Figure 2.1, can be either containers or pallets with width $W$, depth $D$ and height $H$. The coordinates origin is located at the bottom-left-behind of the bin and $(x_j, y_j, z_j)$ is the point where the bottom-left-behind of the box $j$ is positioned. The two dimensional problems are a simplification of the three dimensional problems. They consider the depth of the boxes is equal to the depth of the bins [Martello et al., 2000]. For further analysis the box upper area will be represented by $u_j$ and the bottom area by $b_j$.

![Figure 2.1 - Bin](image1)

![Figure 2.2 - Box](image2)
Packing problems main objective is to determine a feasible arrangement of a subset of boxes, which maximizes a cost function that most times is related to the volume occupation.

\[ f = \frac{\sum_{j=1}^{n} w_j d_j h_j}{n_{\text{bins}} \times WD_f} \]  

(2.1)

The coordinates \((x_j, y_j, z_j)\), which define the position of a box \(j\) inside a bin, become the decision variables.

An arrangement is called feasible when it meets the given constraints:

- Each box is placed completely within the bin (in the case of pallets there can be some previously defined tolerance);
  
  \[
  0 \leq x_j \leq D - d_j, \quad j = 1, \ldots, n \\
  0 \leq y_j \leq W - w_j, \quad j = 1, \ldots, n \\
  0 \leq z_j \leq H - h_j, \quad j = 1, \ldots, n
  \]  

(2.2) (2.3) (2.4)

- A box \(i\) may not overlap another box \(j\);
  
  \[
  x_i \geq x_j + d_j \lor y_i \geq y_j + w_j \lor z_i \geq z_j + h_j \lor x_j \geq x_i + d_i \lor y_j \geq y_i + w_i \lor z_j \geq z_i + h_i, \quad i, j = 1, \ldots, n
  \]

(2.5)

- Each box is placed parallel to the side walls of the container, which comes from the orthogonal packing.

Additional constraints can be added in order to make the packing problems correspond to the intended case study, for the INDASA problem see section 3.3.1.

Another kind of differentiation is based on the type of boxes. Two boxes are considered being of the same type if, given a suitable space orientation, all their dimensions are equal. This way we have problems with:

- Strongly heterogeneous set of boxes, Figure 2.3, many boxes with few of the same type;
- Weakly heterogeneous set of boxes, Figure 2.4, few types of boxes with many of the same type;
- Homogeneous set of boxes, Figure 2.5, in which there is only one type of boxes.
There may be also what is called a guillotine constraint, which requires the pattern to be such that the boxes can be obtained by face-to-face cuts parallel to the walls of the bin [Martello et al., 2000]. Imposing guillotine cuts simplifies the solution approach. However this generally decreases the solution quality, the loading presented in Figure 2.6 would not be possible if a guillotinable constraint was imposed. Hence a loading, Figure 2.6 and Figure 2.7, can be classified as non guillotinable or guillotinable [Pureza & Morabito, 2006].
2.1.1 Distinction between container and pallet

Containers, Figure 2.8, and pallets, Figure 2.9, are usually a physical extension of the bin concept. The main difference between pallets and containers is the presence of walls and ceiling in the container. In the container the bin space is found inside it, in the pallet the theoretical bin is presented on top of it.

In the case of the containers the walls provide a point of support to the boxes. The presence of a ceiling allows containers to be piled without concerns about the way the boxes are loaded inside.

Due to the non-existence of walls and ceiling on pallets the three dimensions referred above may sometimes be slightly exceeded. When different pallets are required to be piled it is important that the top presents a regular area in order to allow other pallets to be piled on top without losing stability.

2.1.2 Different types of packing problems

There are many ramifications of the packing problems [Dyckhoff & Finke, 1992]. Nevertheless, accordingly to the main goal of this work and the available literature, only four will be described in here. In the state of the art section it is possible to view some examples of bin packing problems and container loading problems. In the state of the art section it is possible to view some examples of bin packing problems and container loading problems.

1. Bin packing problem – it is probably the most explored problem in the literature. In general it is treated as specific problem without associating, either pallet or container, as the type of bin. The boxes have to be completely stowed in a minimal number of bins. This problem aims, e.g., at minimizing the required bin costs;

2. Knapsack problem – although it is not so explored as a specific problem as the bin packing problem, usually it is associated to the container loading problem or the pallet loading problem. Differently to the bin packing problem it tolerates some boxes to be unpacked. This problem goal is, e.g., to maximize the stowed volume or the unit contribution of freight.
3. **Container loading problem** – after the bin packing problem, this is most likely the one with more references in the literature. This problem is usually a specification of the knapsack problem with the objective of filling only one container, even if some cases are treated as multi container or as a Bin Packing Problem. Other authors use a container with infinite depth and their objective is to minimize the depth occupied by the boxes.

4. **Pallet loading problem** – This problem usually only differs from the container loading problem in one point: it is treated as a two dimensional loading problem in which only boxes with similar dimension are allowed to be piled on top of each other.

### 2.1.3 Solving the packing problem

Due to the complexity of the packing problem, it is hard to solve it as a single problem. Therefore the problem can be divided in three different sub-problems:

1. **The place representation of the boxes inside the bin** – This is the base of the packing problem. In the one-dimension problem this is not an issue, but with the increasing number of dimensions the difficulty increases. So in the three-dimensional problem this definition can have a big influence in the final result. The number of feasible solutions for the positioning of each of box is directly influenced by the method chosen. Some cases are presented on section 2.2.

2. **The constructive heuristics** – After having a correct definition of the boxes position, the next step is the definition of how to stow the boxes inside the bin. Given a set of boxes, the main goal is how to introduce them in the bin or check the feasibility of the solution. Often, the constructive heuristics are based on one of the first sub-problems and then the selection of the boxes position among feasible points is done resorting to some criteria. If more than one box is available, a simple heuristic, such as first fit decreasing or best fit decreasing, is used to provide an initial loading of the boxes around the bins (sometimes equal to the best solution available). Some constructive heuristics are presented on section 2.3.

3. **The searching heuristic** – The goal of this heuristic is to find the best boxes distribution inside the bins. Depending on the addressed packing problem different approaches may be required. The searching heuristic may be based either on a simple heuristic or on metaheuristics. For the same heuristic, different approaches may be required for different packing problems. The searching heuristic usually uses a constructive heuristic to verify the feasibility of the packing. Some of the searching heuristics are described in the state of the art in section 2.4, both for the container loading problem and the bin packing problem.

However the third sub-problem may be overrided and, even so, obtain good solutions. Nevertheless the combinations of different types of these three sub-problems are the key to a good solution. The arrangement of the three sub-problems can be done in two different ways, see Figure 2.10:
1. The constructive heuristic provides an initial solution and the searching heuristic tries to improve it;
2. The searching heuristic provides a solution by itself using only the constructive heuristic to check the feasibility.

2.2 Place representation of the boxes inside the bin

On packing problems, the search of an efficient and accurate definition of the points where to place the boxes inside the bin, is an important issue since the used method may strongly influence the overall solution.

A possible approach is presented by Beasley [Beasley, 1985], to solve a 2D packing problem. It is based on the discretization of the area into rectangles, as shown in Figure 2.11. In this way, the bottom-left corner of a box is positioned into the bottom-left corner of a rectangle. This method has a disadvantage the number of variables increases with the accuracy of the discretization, therefore increasing the computational time. This approach becomes a good solution to calculate upper bounds through Lagrangian relaxation and sub gradient optimization.
More recently, a methodology called Corner Points, in Figure 2.12, was introduced [Martello et al., 2000]. The Corner Points are defined as the non-dominated-locations where a box can be placed into an existing packing. In two dimensions, Corner Points are defined where the envelope of the boxes in the bin changes from horizontal to vertical. In three dimensions the envelope can be found applying the two dimensional algorithm to each distinct value of the height of the bin defined by the lower and upper lines of each box. Constructive heuristics using corner points may be inefficient in terms of bin occupation because the definition of Corner Points depends on the sequence of the accommodation of the boxes inside the bin.

Figure 2.12 - Corner Points representation

A new method, that extends the Corner Points concept, was introduced as the Extreme Points, present in Figure 2.13, by [Crainic et al., 2008]. This idea provides the means to exploit the free space defined inside a packing by the shapes of the boxes already inside the bin. Thus the Extreme Points are obtained by the projection of the vertices of a box in the surrounding environment, already packed boxes or the walls of the bin. All the Corner Points are Extreme Points, but the second definition allows to explore feasible points outside the envelope, other way holes in the packing that can be occupied by other boxes.
Another method based on a graph theoretical approach, as seen in Figure 2.14, for the characterization of multi dimensional packing was proposed by [Fekete & Schepers, 2004]. The authors considered the relative position of the items in a feasible packing and defined a graph describing the item overlapping according to the projection of the items in each orthogonal axis.
2.3 Constructive heuristics

After finding a good way to represent the position of the boxes inside the bin another problem arises, this is how to pack the boxes inside the bin. There are several methods available, depending on the problem or the constraints.

A typical approach is the extrapolation of the shelf approach used by many authors to the two dimensional packing problem (e.g. [Berkey & Wang, 1987]). In such approaches the boxes are ordered by non increasing height and packed from left to right in rows forming shelves, the selves height corresponds to the height of the tallest box. The approach is extended to the three dimensional problem by creating shelves, with width \( W \) and Height \( H \) and depth equal to the box with biggest depth on the shelf, using a two dimensional algorithm to create the shelves, [George & Robinson, 1980] used this algorithm to minimize the depth of the bin. More recently this algorithm has been adapted to the one-dimensional problem in order to pack the boxes into one or more bins of limited depth [Pisinger, 2002]. This algorithm can be also adapted to the building of layers as an alternative to walls instead of different depths the layers have different heights, with width \( W \) and depth \( D \).

Another approach is to find the available space inside the bins, using the known position of the boxes inside it. This method usually asks for a predefined boxes order, e.g. volume-height or area-height. After that the next unpacked box is chosen and according to the free space inside the bin; a list of feasible points to store the box are chosen. If more than one point is possible then different methods are available to choose among them, e.g. choose the one with inferior \( x,y,z \) coordinates (in this order) [Crainic et al., 2008], or compute a parameter to choose the position as the goodness number [Wei et al., 2009]. The drawback of the shelf approach is that it introduces guillotine cuts leading to the underutilization of the bin.

An exact solution for the three dimensional problem is given by a branch-and-bound algorithm [Martello et al., 2000]. This method uses the corner points to find feasible positions for the different boxes. The best performances are obtained when the boxes are pre-ordered according to non increasing volumes. Although this method considerably limits the enumeration compared to a naïve technique trying to place all boxes, practical experiments show that the algorithm is very time consuming.
2.4 State of the art algorithms

After describing how to solve a packing algorithm and the phases involved, state of the art algorithms for these problems are presented in this section. Two types of problems are reviewed: the Bin Packing Problem and the Container Loading Problem. As stated in section 2.1.2 the Bin Packing Problem is the most used in the literature and can be adapted to the current problem. Secondly, as the Container Loading Problem is based on the Knapsack Problem, by reviewing the first, it is possible to understand both. Additionally the Pallet Loading Problem is also a simplification of the Container Loading Problem, since it only optimizes the two-dimensional space, and the Container Loading Problem offers a three-dimensional solution. So by reviewing these two different problems it is possible to have a general idea of four different problems.

2.4.1 Bin packing problems

The Bin Packing Problem (BPP) consists of orthogonally packing all the boxes into the minimum number of bins with identical dimensions. Multi dimensional bin packing problems have been studied mainly in their two dimensional versions. Three dimensional versions are quite recent [Martello et al., 2000][Pisinger, 2002].

2.4.1.1 Least waste first for the 2DBPP

A recent approach to the 2D problem is based on a least waste first strategy [Wei et al., 2009], allowing non-guillotine packing. It uses a methodology similar to Corner Points to find the position of the boxes inside the bin. Wasted area is defined as a gap between a Corner Point and the border of the bin that is inferior to the smallest edge of the unpacked. A variable called goodness numbers is defined in order to classify the way in which the available space is used.

This method selects a box and Corner Point such that the wasted area will be minimal. If the number of Corner Points is greater than one it selects the one with greater goodness number. In case there is more than one box that satisfies such conditions, the first in the list is the one selected.

As this method depends on the order in which the boxes are introduced, the boxes are first ordered by area and the height and width are swapped so that the width is no shorter than the height. Then a fast local search is introduced by randomly swapping two boxes in order to find a better solution. If a better solution is found, that order is saved and a fast local search is applied again. This procedure is used until an optimal solution is found or a number of iterations is reached.
2.4.1.2 Guided local search for the 3DBPP

A Guided Local Search (GLS) [Faroe et al., 2003] was used to solve the 3D bin packing problem. This method has proven to be a powerful metaheuristic to solve hard combinatorial problems. The GLS algorithm starts with an upper bound on the number of bins obtained by a greedy heuristic. Then whenever a feasible solution is found the upper bound is reduced by one bin until is impossible to find a feasible solution or the lower bound, theoretical minimum number of necessary bins, is reached. To speed the local search, a Fast Local Search algorithm is used. Since the neighborhood is quite large the FLS speeds up the search for a local minimum.

2.4.1.3 Exact branch-and-bound for the 3DBPP

The optimal solution for the 3D problem was found using a branch-and-bound algorithm [Martello et al., 2000]. That approach was later improved to introduce robot packing [Martello et al., 2007]. This algorithm is based on a two level decomposition principle for the two dimensional bin packing problem [Martello & Vigo, 1998].

A main branching tree assigns the boxes to the bins without verifying the feasibility of the package. When a box is assigned to a non empty bin, a lower bound, checking if it is theoretically possible to stow the box in that bin, is computed. If the lower bound is higher than a given constant, the node is killed. Otherwise, two heuristics are executed for the sub-instance and, if a solution requiring a single bin is obtained, the assignment is accepted. In case neither the node is killed nor the assignment accepted, an exact single bin filling algorithm is used to test the feasibility of the packing. In order to obtain a good upper bound at the root node of the branching tree, two heuristics, described in [Martello et al., 2000], are used in order to limit the number of uses of the exact single bin filling algorithm.

2.4.1.4 Tabu-search for the 3DBPP

In order to solve the 3D bin packing problem a two level tabu-search was used by [Crainic et al., 2009]. A first solution is computed using a heuristic derived from the well-known first fit decreasing algorithm. Given the initial solution the meta-heuristic algorithm iteratively discards the bin with worst value of fitness function defined in [Lodi et al., 2004].

The first level tabu-search works with a set of boxes and a fixed number of bins. Its goal is to find a set of boxes-to-bins assignments able to produce a packing for each bin such that it fits within the dimensions of the bin and the boxes are not overlapping.
In order to verify the feasibility of the packing defined by the first level heuristic a second level tabu search-based local search, which uses the implicit solution representation given by an Interval Graph proposed by [Fekete & Schepers, 2004] reducing the search space.

### 2.4.1.5 Multiobjective in 2DBPP using swarm optimization

In the literature is hard to find bin-packing problems with multi objectives, Liu et al. [Liu et al., 2008] use swarm optimization to find a solution for both the minimization of the number of bins and to optimize a desired position for the center of gravity. To achieve this purpose it’s used an evolutionary approach allowing the use of a mutation operator as a source of diversity. The accommodation of the boxes inside the bins is done by a bottom left fill heuristic.

### 2.4.1.6 Extreme points based heuristic for 3DBPP

Using the second approach referred in the constructive heuristics, using Extreme Points, applied to a First Fit Decreasing (FFD) and Best Fit Decreasing (BFD) heuristic, it was possible to obtain a simple heuristic to solve the 3DBPP [Crainic et al., 2008].

FFD heuristics load the ordered boxes one after another in the first bin they fit. The BFD heuristics tries to load each box in the best bin, the bin which after loading the box has the more free space. In both cases when there is no bin available a new one is created.

For both these methods, it is necessary to sort the boxes previously and therefore different ordering rules were tested. In the case of FFD, the items are placed on the first bin into the feasible extreme point with inferior z, y, x coordinates (in this order). In BFD case, different variables were tested to quantify the best fit. The results obtained with this approach, which requires negligible computational effort, yields better results not only to existing constructive heuristics, but also to more complex methods as time limited branch-and-bound.

### 2.4.2 Container loading problems

The container loading problem is usually associated to the knapsack problem. The objective is to select a subset of boxes that fit into a single container (bin) in order to maximize the profit, assuming that each box has an associated profit.

In the container loading problem it is assumed that the profit of each box equals its volume. The problem becomes filling the container the most possible.
2.4.2.1 Heuristics based on wall building algorithm

An extension of the wall building algorithm [George & Robinson, 1980], described in section 2.3, is proposed by using a tree-search algorithm to find the set of layer depths and strip widths which results in the best overall filling [Egblad & Pisinger, 2009]. The algorithm differs from previous wall-building algorithms for the knapsack container loading problem in several respects: by using a m-cut enumeration scheme for choosing depths and widths/heights, the algorithm has a possibility of backtracking in order to reach depth and width/height combinations that fit well together. Each strip filling problem is solved to optimality by use of an efficient knapsack algorithm, and pairings of boxes are allowed to form layers of uniform depth.

Several ranking rules for the selection of layer depths and strip widths have been investigated, pointing out that a compromise between the largest and most frequent dimensions leads to the best solution quality.

Also a greedy randomized adaptive search procedure (GRASP) is presented by [Moura & Oliveira, 2005], adapted to a modified George and Robinson heuristic (GRMod) wall building algorithm [George & Robinson, 1980]. In this modified algorithm the rejected spaces are amalgamated in order to improve the space use. Also in order to improve the amalgamation the layer width is flexible in order to optimize the free space.

After a first solution is found it is improved by means of local-search algorithm. This local search is based both on a greedy and random algorithm. A greedy algorithm would lead to the choice of the best box type, and a completely random strategy would draw from the entire list. The GRASP chooses from a restricted candidate list.

2.4.2.2 Hybrid genetic algorithm for the CLP

A genetic approach for this kind of problem is presented by [Bortfeldt & Gehring, 2001]. A hybrid genetic algorithm is used to generate stowage plans with a wall building structure. Stowage plans are generated by means of a subordinated heuristic [Gehring et al., 1990].

This heuristic is used to create the initial solution, a first set of walls as starting individuals for the evolution. For the generation of subsequent generations are used problem-specific operators, mutation and crossover. These transfer some unchanged walls from the parents to the offspring and then the heuristic is used to complete the offspring. The basic heuristic and the operators were designed so the walls order was initially of no concern.

The sequence of the walls is then chosen for the best stowage plan. While the sequence can be determined arbitrarily for constraint free problems, in case of a balance constraint a special heuristic is used to order the walls.
2.4.2.3  Parallel algorithms to solve the CLP

A parallel tabu search algorithm for the container loading problem associated with a weakly heterogeneous load was presented by [Bortfeldt et al., 2003]. The parallel searches are carried out by differently configured instances of a tabu search algorithm, which cooperate by exchanging of solutions at the end of defined search phases. The algorithm used is hierarchically divided in three modules: the lowest module is based on a simple heurist who serves the complete loading of a container, the middle module is a tabu search algorithm search, and within the uppermost module several differently configured instances of the tabu search algorithm evolve independent search paths. The instances cooperate through the exchange of best solutions. The exchange always takes place at the end of defined search phases and exerts an influence on the further search of the individual instances.

In the application of SA and TS [Mack et al., 2004], simulated annealing clearly wins over tabu search, especially for problems with larger solution spaces. However, the fine-tuning of the steering parameters of SA was long and complicated process compared to the parameterization of TS. Hybridization proved to be a powerful extension of SA and TS. In the parallel version, the improvements due to parallelization were comparable to those found with hybridization.

2.4.2.4  A tertiary-tree-based dynamic space decomposition approach

Instead of using a static method to partition the free space, a dynamic space partition is proposed, and the usage of a tertiary-tree-based heuristic is used by [Wei et al., 2009]. The tertiary-tree representation comes from the fact that whenever a box or cuboid group of boxes is packed, three different spaces are made available.

The essence of the proposed dynamic space decomposition approach is to minimize the size of most likely unusable spaces and in opposition maximize the size of most likely usable spaces. It is also proposed a optimal-fitting rule to rank boxes, which integrates the aggregate volume of homogeneous boxes and the orientation of the boxes together to determine a group of boxes that make the best use of the free space.
3 The packing problem at INDASA

At the beginning of the expedition of products, the order lines must be distributed among the available space. This sometimes leads to the reloading of pallets and modifications on the expedition list.

With the current packing method, which is basically a trial and error approach, it is hard to guarantee that a given order line is not split among different pallets, which is not desired. Further, some packing rules are not met, such as having the heaviest products on the bottom, in order to avoid deformation of the products that are below. Therefore INDASA has a great need of a tool that optimizes the packing of expedition containers.

In this section, the packing problem at INDASA and a model of the problem that will be used to develop the packing algorithms are presented.

3.1 Description of the products

The product variety in INDASA is quite large (over twenty thousand different products). However the great diversity of products does not correspond to the same dimension variety, as some products only differ in their features, not in dimension.

3.1.1 Abrasive composition

Generally an abrasive is composed by many layers of different materials, as represented in Figure 3.1, where the grain, glue and backing are the essential and indispensable materials. There are also the coatings and velcro (in some cases), which is one of the main responsible for the weight differences between products. Different materials are used depending on the purpose of the abrasive. The used grain has influence in the goal of the work in which the abrasive is used.
3.1.2 Production and composition process

The production of abrasives passes through a series of steps until the final product is conceived. This covers all the necessary stages to the production of the semi-finished product to be used on the next stage of transformation, from where the final product comes.

Abrasives are produced from the combination of the materials described on the previous section. These materials are processed in order to form abrasive rolls, denominated jumbos, which are characterized by the type of abrasive, grain, brand and dimension. The jumbos can suffer some other alteration such as humidification or cure. After these operations the rolls denominated virgin jumbos will enter in the transformation phase [IDMEC, 2007], as it is depicted in Figure 3.2.
The transformation process consists of all the required tasks over the virgin jumbo, in order to finalize the abrasives to be put on the market. This process is mainly based on jumbo’s cutting operations, so as to define the different shapes and group them on the adequate boxes, shown in Figure 3.3.

![Transformation Process Diagram](image)

**Figure 3.3 - Transformation Process**

### 3.1.3 Different types of products

To identify the products a Stock Keeping Unit (SKU) is used, which consists of a numeric code which defines each kind of product. The great variety of SKU corresponds mainly to differences in grain, brand or kind of abrasive. Some products with different forms are presented below.

The great diversity of SKU may be explained by all the important information regarding a packed set of products:

- Type;
- Form, see Figure 3.4;
- Dimension;
- Grain dimension;
- Units per pack;
- Primary/Secondary pack;
- Location in the warehouse.
3.2 Expedition list structure

The objective of the model is to create an expedition list. Each expedition is the result of the combination of different orders related to one given client. Each order comprehends a set of order lines which correspond to different SKU.

The expedition list, whose structure can be seen in Figure 3.5, corresponds to the set of order lines that are available for expedition. However the expedition list may not include all the order lines present in the different orders, issued by the client, due to the lack of space.

![Figure 3.4 - Form types](image)

![Figure 3.5 - Expedition List Structure](image)
Each order line corresponds to the information needed to create an expedition list, such as:

- Line number;
- SKU;
- Product description
- Due date
- Quantity available
- Quantity needed

When an order line is added to the expedition list some parameters are added, e.g. the weight of the line and the number of the pallet in which the order line is stowed.

3.3 Problem description

The expedition of orders to a large number of clients is done resorting to containers of variable dimensions. In this way it is fundamental to optimize the occupied volume. The products are stowed into pallets of variable known dimensions which are then packed into containers.

The main goal is to develop a model that, given the available order lines to expedition, identifies:

- The order lines that may be dispatched to maximize the volume used in the expedition container;
- The sequence of the order lines in each pallet, and the pallets on the container;
- The weight each pallet and total weight of the container.

For the order lines that are not guaranteed to fit the expedition container, the model should return a warning. The problem can be approached in two ways, as it can be seen in Figure 3.6:

- The model waits for the user to select a new set of order lines that may or not fit the container;
- The model automatically selects the set of order lines which maximize the containers occupation.
3.3.1 Problem constraints

This specific problem has some additional constraints that are not defined in the general packing problem, Chapter 2. These constraints are of physical nature, to prevent damaging the products, and of practical nature, to guarantee the clients satisfaction.

1. Stability constraint
   - The bottom area of each product must be partially supported by the upper area of the products bellow or completely supported by the floor of the pallet. This prevents the upper products from falling down. Being \( \alpha \) the percentage of the box that must be supported:
   \[
   \text{if } z_j > 0 \text{ then } \sum_{i=1}^{n} u_i \cap b_j \geq \alpha \times b_j, \quad j = 1, ..., n \quad (3.1)
   \]

2. Rotation constraint
   - Most products only allow rotation around the vertical axis. This avoid that the products are placed in positions that can make damage them. In this way the products can only take two positions, the original position or rotated \( 90^\circ \) around the vertical axis due to the orthogonality of the packing.
3. Weight constraint
   - Products with bigger weight must be on the bottom. This prevents the bottom products from being damaged by excessive weight. Considering $\epsilon$ a weight margin which prevents a heavier upper product from damaging the other products:

   \[
   \text{if } z_i + h_i = z_j \text{ and } u_i \cap b_j > \emptyset \text{ then } \omega_i \geq \omega_j + \epsilon, \quad i, j = 1, \ldots, n \tag{3.2}
   \]

   \( \omega_j \) - Product \( j \) weight

4. Different products constraint
   - Rolls must be on different pallets than the other types of products, since the rolls are packed in telescopic pallets.

5. Product agglomeration constraint
   - Products with the same SKU must be on the same pallet, unless one pallet is not enough to stow all of them. In that case the products with the same SKU must be stowed around the minimal number of pallets possible. This is important to assure the client’s satisfaction who does not want to be unloading many pallets to get only one kind of product.

3.3.2 Data processing

The required data to solve the problem do not resume to the data available in the order lines. So in order to solve the problem it is necessary, in addition to section 3.3, to acquire the dimensional and weight values of each product and the dimensional values of the pallets and containers involved. Also it is important to know that some data are collected before the packing process itself, products data, and some data are inserted in the model itself, pallets and containers dimensional data. In order to do so, the generalized algorithm described in Figure 3.7 is used.

The algorithm used to determine the dimensions of the products is based on previous work [Vieira, 2008]. The first step consists of loading the dimensions and description of the different products according to the code bar. Secondly using the code bar again the product weight is loaded from a different file.
As for the expedition list it must contain not only the data related to the products (bar code, SKU, corresponding order line, weight, dimensions) but also all the information related to the pallets and containers. For the pallets the products inside them must be known, making possible to know their weight, as the containers in which they are. For the containers the pallets which are inside must be relevant as they give all the information of what is inside them as well as the total weight of the container, important for the expedition, as shown in Figure 3.8.
3.4 Solving the INDASA expedition problem

This specific problem asks for two different packing problems. The distinction between the first one which is to be solved depends on the followed approach. One of the problems is the packing of the products along a number of pallets. The other is the packing of the pallets in the containers.

It is important to emphasize that in the pallet loading the pallet is considered as a bin and the products the boxes, while in the container loading the pallets become the boxes and the container the bin, as in Table 3.1.

<table>
<thead>
<tr>
<th>Packing Problem</th>
<th>Container Loading</th>
<th>Pallet Loading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bin</td>
<td>Container</td>
<td>Pallet</td>
</tr>
<tr>
<td>Box</td>
<td>Pallets</td>
<td>Products</td>
</tr>
</tbody>
</table>

Table 3.1 - Boxes and Bins assignment

3.4.1 Pallet loading

The optimization of the pallet volume is clearly a three dimensional and complex problem. There may be more than one approach, but here it is used a bin packing approach.

The problem can be considered a heterogeneous problem without distinction between weak or strong. There are up to one hundred different types of products which make the problem heterogeneous, but if some types of products have hundreds of items, other types are limited to less than ten items. In this way, assuming only one type of pallets, the goal would be to pack the products into the minimal number of pallets. The main difference to the literature is the number of products which are to be stowed, in addition to the additional constrains.

3.4.2 Container loading

Regarding the loading of the container with the pallets, the problem can be approached as a pallet loading problem. Even if it is a three dimensional problem, only pallets with identical or inferior dimensions are piled on top of each other. This way the problem resumes to a two-dimensional problem in which the maximization of the used container bottom area is the main goal.

This is clearly a weakly homogeneous problem, in which there are typically only two types of boxes (pallets), and even each type of boxes regards a low number of items. Therefore this is a simpler problem than the pallet loading.
The main difference to the typical pallet loading problem is the fact that in addition to the dimensions of the pallets, the algorithm has to also take into account the pallets that have more space used, in order to maximize the volume of products in the container. Due to the fact that the pallet loading problem is based on a knapsack problem, so from here on the problem will be defined as Knapsack Problem.

### 3.4.3 Packing process

The packing process is the core of the work as it is in here that the requirements described in section 3.3 are applied, and the pallet loading and the container loading are inserted.

This packing process, schematized in Figure 3.9, allows determining the necessary number of pallets and containers and the distribution of the products inside each pallet. Additionally, it also permits the user to change the products list if the results do not satisfy the number of containers which are to be used. In this way the algorithm should work according to:

- The products list is divided into rolls and non rolls as they have to been packed in different pallets;
- A bin packing algorithm is used to stow the products, rolls and non rolls, into a minimal number of pallets;
- A knapsack algorithm is used to fit the pallets into a container according to the products volume in each of the pallets. If after a container is full there are pallets left a new container is created. In this way if there are more containers than the ones allowed only the last containers to be filled have to be eliminated. This prevents another repacking in order to achieve the number of containers pretended, as the first container is always the one with better volume occupation;
  - It is given the user the opportunity to choose the products to be eliminated in order to decrease the number of containers.
Order Lines

Divides between rolls and other products

Rolls

Use Bin Packing algorithm to compute the number of pallets necessary

Rolls pallet dimension

Rolls pallets

Non rolls

Use Bin Packing algorithm to compute the number of pallets necessary

Non rolls pallet dimension

Non rolls pallets

Use Knapsack Packing Problem to compute the number of containers needed

Container dimension

Number of containers is allowed?

Yes

Expedition List

No

Products are to be eliminated in order to have less containers

User manual selection of the products to be deleted

Automatic selection of the products to be deleted

Figure 3.9 - Packing process
4 Proposed heuristic algorithms

This chapter introduces new heuristics specially designed to solve the INDASA problem. After reviewing the available algorithms present in the literature, the main problem was finding the one that fits the problem. Most algorithms are usually used for problems with few boxes, one hundred items, while this problem asks for a larger number of boxes, up to five thousand different items. This difference in the number of boxes would ask for an adaptation of any chosen algorithm in order to adjust. This problem with the number of boxes would be regarding the pallet loading, in the case of the container loading it would be a simplification of the first problem.

Most importantly, concerning the INDASA problem, it would be necessary to take into account the constraints described in section 3.3.1 as most algorithms present on the literature only respect the basic constraints described in the definition of packing problems in section 2.1.

This way the first step was constructing a simple layer building algorithm, as described in the constructive algorithms section, which would take into account the constraints and size problem referred above. This algorithm should solve the given INDASA problem, nevertheless some simplifications were made that eventually would not satisfy the stability constraint. In addition to this, the algorithm was not built so a spatial representation of the boxes in the bin was given as an output. Although this was not a requirement, nonetheless was a very useful tool.

After a larger study on the available literature it was decided to adopt the Extreme Points [Crainic et al., 2008] methodology, and adapt it in order to obey all the constraints previously referred and the problem data size. These modifications were made so that the algorithm was able to solve even problems that did not ask for the given constraints resulting in a more generic algorithm than the Layer building. In addition the structure of this algorithm also gives the spatial representation of the boxes in the bin.

It is important to refer that both algorithms can easily be adapted to either a bin packing problem or a knapsack problem by choosing the correct search heuristic, or by adapting a simple constructive heuristic like First Fit Decreasing.
4.1.1 Orientation selection

For both the algorithms referred above it is possible to have a rotation of the boxes. To choose the orientation of a given box is used an algorithm referred in [Vieira, 2008]. This algorithm computes the number of boxes that fit into a given free volume for each orientation. The orientation which allows the maximum number of boxes is the one chosen.

As there is an orientation constraint that only allows rotation along the vertical axis, only two different positions are allowed for this specific problem. Nevertheless this algorithm is prepared to allow the six possible positions if no rotation constraints are imposed.

4.2 Layer building algorithm

As referred above the structure of this algorithm is generically described in the constructive algorithms section. The non existence of a place representation of the boxes inside the bin was solved by dividing each bin into a set of layers, in Figure 4.1, and each layer into a set of rows, see Figure 4.2.

Since one existing restriction is about the weight, it is considered that the heaviest boxes must be packed first. If this restriction was non-existent it would be possible to use other sorts of ordering such as area, volume, order line volume, etc. Another possibility would be to make clusters of weight and inside each cluster order the boxes by the previously said orders.
As the problem is dependant of each box height to define and fill the layers, as described in Figure 4.3, a correspondence between products with the same height and products with height that are sub-multiples of the main height is made. In this way, if a type of products is not enough to fill a layer, another type of products with similar height is used to keep the layers with the same height.

Having the order defined and the relations between heights computed it is time to start building layers, as described in Figure 4.4. Each layer height is defined by the first available box in the ordered list.
Having a layer’s height defined, then the layer is filled by rows. Each row height it is equal to the correspondent layer’s height. To do so, two different paths are defined, thus providing two different algorithms. In the first path whenever possible it is started a row with similar height than the last one. In the second path it is started a new row with the first possible box. Hence for the second algorithm the stability constraint may not be met. However by using the two algorithms it is possible to obtain better percentages of occupied volume.

Figure 4.4 - Layer filling algorithm
4.2.1 Filling multiple bins

Using the layer building algorithm described above it is possible to fill a single bin. However the objective is to stow all the boxes into a minimum number of bins, which may ask for more than one bin.

In the case of the container loading, the logical step would be to fill successive bins until there was no box left, which would be a good approach. Nevertheless for the pallet loading, as the algorithm is structured, this eventually would not meet the product agglomeration constraint, see section 3.3.1. This way it was projected an iterative algorithm that succeeds in meeting the given constraint, as explained in Figure 4.5.

As presented before, the layer filling algorithm allows two different paths to solve the problem. In addition considering the bin with two possible orientations, it also gives two different solutions. In this ways the algorithm solves four different packing for each bin and only the one with better percentage of occupied volume is chosen.
At this point two problems arise after removing the products that do not follow the product agglomeration constraint;

1. Is the stability constraint kept?
2. Are some bins not under packed? Some bins may have almost all the products removed by applying the agglomeration constraint.

To avoid the second problem, the bins with lowest percentage of occupied volume are repacked in order to improve it. This becomes part of the solution to the first problem, as partially maintaining the original packing solution it is assumed that the packing stability is also kept.

4.3 Adapted extreme points algorithm

As previously said this algorithm is based on the work developed by [Crainic et al., 2008] and in the pseudo code presented in that article. However the methodology took some changes in order to adapt to larger problems as the one explored in this work. Instead of packing only single boxes, it packs sets of boxes.

The Extreme Points algorithm is based on the concept that all feasible points must be considered for the packing so that there are not spaces inside the bin that are not used. So for each extreme point there is a related area on which a given box can be packed. Only if no box can be packed on that area is the Extreme Point deleted. This algorithm is schematized in Figure 4.6 for a single bin case.
4.3.1 Extreme point definition and selection

Each extreme point is the projection of three points of a given box on the surrounding environment. Considering the coordinate system defined in Chapter 2, the three points projected are \( \{(x_j + d_j, y_j, z_j), (x_j, y_j + w_j, z_j), (x_j, y_j, z_j + h_j)\} \), and each point is then project in all directions except the one in which the origin of the box \( j \) is, Figure 4.7. In this way, for each box there is a maximum of six extreme points, and if two projections are overlapping only one extreme point is considered. Each extreme point \( i \) will have their coordinates given as \((EPx_i, EPy_i, EPz_i)\).

One of the problems in this method is choosing the Extreme Point on which to place the box. This problem becomes even more complicated as the number of boxes packed increases and so the number of Extreme points.
The better solution would be to base the selection in some sort of parameter that would measure the quality of each Extreme Point for a given box. However the increasing number of available points turns that task quiet heavy and so not very appealing.

Therefore, the choice fell on a more simple approach that is sorting the extreme point by their coordinates and so choosing the first one of the list that presents a feasible solution.

### 4.3.2 Extreme points available area

As there is a stability constraint, the volume available for packing corresponding to each extreme point is different. If the extreme point is on the bottom of the bin, this area depends only on the boxes that are at the same level than the given extreme point. However, the extreme points that are not at the bottom of the bin, have their available area constrained by the boxes which upper area is on the same level as the extreme point.
In this way the two sets of boxes are defined for each extreme point $i$, box already packed $j$ and box to be packed $k$

1. **Floor boxes**
   
   $\text{if } EPz_i = z_j + h_j \text{ and } EPx_i < x_j + d_i \text{ and } EPy_i < y_j + w_i \tag{4.1}$

2. **Obstacle boxes**
   
   $\text{if } z_j - w_k < EPz_i < z_j + h_j \text{ and } EPx_i < x_j + d_i \text{ and } EPy_i < y_j + w_i \tag{4.2}$

The floor boxes represent the area on which the box to be packed will be placed and the obstacle boxes are the ones that, as the name says, become obstacles for the box to be packed. This way for each box to be loaded the available area associated to a given extreme point will vary as the boxes height is a factor to select the obstacle boxes.

With these two sets of boxes, it is possible to compute the associated available area for packing for each extreme point, as depicted Figure 4.8.

![Figure 4.8 - Extreme Points available area](image)

In Figure 4.9 it is possible to watch two different Extreme Points with the respective available areas in red and orange. The available area does not have to be mandatorily quadrangular as shown by the available area presented in Figure 4.9. One of the problems would be then how to define the distribution of the products among it. However the algorithm only places quadrangular bottom area boxes in quadrangular areas.
So the available area may have to be decomposed into smaller areas and one of those areas must be the one chosen. Those decompositions are defined by the extreme point and one of the available area vertices as shown in Figure 4.10. If two of those areas are available, the one which can stow more boxes is chosen.

4.3.3 Packing sets of boxes

The main difference to the previously referred algorithm is the fact that it packs multiple boxes instead of single boxes. This adds to the algorithm the possibility of packing larger instances.

The addition of multiple boxes is considered using a set of boxes of the same type. In this way the set of boxes is defined by multiple boxes aligned in the direction of the x or y axis, with all the boxes having the same orientation.
So for each iteration, a row of boxes is packed and the volume defined by the row is the one used in the update of the Extreme Points. In this way the number of iterations is reduced, as in each iteration multiple boxes are stored, and the number of extreme points is reduced.

4.3.4 Stability constraint

The stability constraint, section 3.3.1, asks for a percentage of the box bottom area to be supported, by other box or the bin floor. In this way a method of guaranteeing the stability is having the geometrical center, admitting this is coincident to the mass center, positioned on top of the box bellow.

This constraint was added admitting that it was possible to extend the floor area in both the x and y axis dimensions with a certain value. This value was defined by the minimal horizontal dimension of the box to be packed. The extension of the floor area would be limited by the bin area itself. In this way there are three different scenarios depending on the percentage of the minimal dimension:

1. The percentage is zero – all the products are fully supported
2. The percentage is between zero and fifty – the stability constraint is maintained but the boxes may not be fully supported
3. The percentage is more than fifty – it is not possible to know if the constrain is met

In the extreme case of having this percentage defined as infinite, the floor areas are defined by the bin bottom area. By doing so the stability constraint is completely disregarded.

4.3.5 Filling multiple bins

The agglomeration constraint, referred in section 3.3.1, does not allow to sequentially use the single bin filling algorithm to fill multiple bins. That would result in having some types of boxes divided between different bins. This was an issue for the pallet loading, in the case of the container loading this would be a good approach.

Thus it was decided that each type of boxes should be defined as a block that should be packed in the same bin. The exception would be the types of boxes who would not fit all in the same bin, and in that case those types should be divided in blocks that obey the constraint. To define these blocks it would be used the single bin filling algorithm.
For those blocks, a First Fit Decreasing heuristic was applied in order to distribute them along a minimal set of bins, as described in Figure 4.11. A Best Fit Decreasing algorithm was another possibility, but the computational weight of testing all Extreme Points of multiple bins become an issue, which led to not using that method.

In the next chapter an Ant Colony Optimization algorithm is proposed due to the large search space. This should help by giving extra flexibility to the algorithm, by having the possibility of defining different cost functions.
5 Ant colony optimization applied to packing problems

Ants appeared on earth some one hundred million years ago, and have a current total population of $10^{16}$ individuals. Most of these ants are social insects, living in colonies of thirty to millions of individuals. The complex behaviors that emerge from insect swarms have intrigued humans, and there have been many studies aimed at a better comprehension of these behaviors [Engelbrecht, 2005].

The main idea is that the self-organizing principles which allow the highly coordinated behavior of real ants can be exploited to coordinate populations of artificial agents that collaborate to solve computational problems. Several different aspects of the behavior of ant colonies have inspired different kinds of ant algorithms. Examples are foraging, division of labor, brood sorting, and cooperative transport [Dorigo & Stützle, 2004].

One of the most successful examples of ant algorithms is known as “ant colony optimization” and it is inspired by the foraging behavior of ant colonies. The visual perceptive faculty of many ant species is only rudimentarily developed and there are ant species that are completely blind. In fact, an important insight of early research on ants’ behavior was that most of the communication among individuals, or between individuals and the environment, is based on the use of chemicals produced by the ants, the pheromones.

Based on the double bridge experiment [Deneubourg et al., 1990] it was possible to notice that ants have a built-in optimization capability. By using probabilistic rules based on local information they can find the shortest path between two points in their environment. Taking inspiration on that experiment it was possible to design artificial ants, using a graph approach, simulating the behavior shown on that double bridge experiment.

Using this modeling it was possible to progressively solve different problems with Ant Colony Optimization algorithms as is shown in [Dorigo et al., 2006]. Examples of such problems are: the traveling salesman [Stützle & Hoos, 1997], the vehicle routing [Reimann et al., 2004], the sequential ordering [Gambarella & Dorigo, 2004], the course timetabling [Socha et al., 2003], the graph coloring [Costa & Hertz, 1997] or classification rules [Martens et al., 2006]. For the Bin Packing Problem, [Levine & Ducatelle, 2001] found an approach for the one dimensional case.
5.1 Applying the ant colony optimization to the packing problem

One of the issues in packing problems is the order in which the boxes are stowed. There are many alternatives to finding this order as shown in chapter two. Here we present another alternative, by combining the Ant Colony Optimization with the Extreme Points algorithm referred in the previous chapter, in order to find a viable solution to solve both the Bin Packing Problem and the Knapsack Problem.

As previously stated the main goal of the Ant Colony Optimization is finding a good order in which the boxes are packed.

The basis of this type of optimization is to model the problem in question as a graph problem. In this case the nodes are considered to be boxes or blocks of boxes and the paths are a representation of the order in which each box or block of boxes enters the bin.

In order to have a better understanding of the given problem a simple example with only four boxes, described in Table 5.1, is used in order to obtain a graph representation.

<table>
<thead>
<tr>
<th>Box</th>
<th>Weight</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>48</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 5.1 - ACO example

Considering that in this example there is no restriction of the order in which the boxes are packed, all the paths are bidirectional. In the next sections some paths will only have one direction result of the problems constraints, Figure 5.1.
5.1.1 The zero node

Since the order in which the bins are filled is not important, only the order in which the boxes are stored inside the bins is relevant, a zero node is created. The zero node is the starting point of each ant, when a bin is starting to be filled, and act as the ants nest. In this way the path followed by the ant, from the zero node to the last box to be packed, is the path between the nest and the food source. The addition to the previous example results in Figure 5.2.

![Image of a graph showing the zero node insertion](image)

Figure 5.2 - Zero node insertion

5.1.2 The pheromone trail

The pheromone trail is a way of inserting the efficiency of each ant into the problem. Since the goal is to maximize the percentage of occupied volume and for multiple bins to minimize the number of bins used, the pheromone trail becomes a measure of these two parameters. So the pheromone trail given by packing box $j$ after box $i$ is:

$$\Delta r_{ij} = \frac{(\phi_j^k)^q}{n_{bins}^p}$$  \hspace{1cm} (5.1)

$\phi_j^k$ – percentage of occupied volume in bin $k$ where the box $j$ is found.

$q, p$ – parameters to weight the importance of the percentage of occupied volume and the number of bins necessary respectively.
The update of the pheromone trail should be done using an evaporation coefficient $\rho$. So for an iteration $t+1$ it is possible to write:

$$\tau_{t+1} = \tau_t (1 - \rho) + \Delta \tau$$  \hspace{1cm} (5.2)

Another problem is defining the value for the initial pheromone trail. So it will be considered a value expressing a simple lower bound and using a bin without any free space. This lower bound is given by dividing the total volume of boxes by the bin volume, thus obtaining a theoretical minimum number of necessary bins. Having this optimal case the initial pheromone trail can be multiplied by a factor $\gamma$, allowing to tune the initial pheromone trail. So for this case the initial pheromone trail should be:

$$\tau_0 = \gamma \times \frac{1}{\left( \sum_{j=1}^{n} w_j h_j d_j \right)^{\beta}}$$  \hspace{1cm} (5.3)

### 5.1.3 The heuristic matrix

The heuristic matrix is a way to introduce some heuristic information into the choosing of a path. In this way the ants “see” the boxes that should be packed first. In the case of the pallet loading, the boxes with biggest weight should have a greater probability of being packed first, in the container loading for example boxes with more volume should be considered first. So the heuristic matrix parameter from packing box $j$ after box $i$ should be for example the weight or volume of the box $j$.

$$\eta_{ij} = \text{weight}_j \text{ or } \eta_{ij} = w_j h_j d_j$$  \hspace{1cm} (5.4)

In addition as a node must not be considered more than once, the heuristic matrix must suffer an update each time a given node is selected. This way if a node $j$ as already been chosen it must be updated:

$$\eta_{ij} = 0, \quad i = 0, \ldots, n$$  \hspace{1cm} (5.5)

Also as the zero node is always the starting point of all the paths, it will not be possible to have it in the middle of a given trail. This way the heuristic matrix from all the nodes to the zero node shall be zero:

$$\eta_{i0} = 0, \quad i = 0, \ldots, n$$  \hspace{1cm} (5.6)
5.2 Introducing the constraints

There are two constraints that are not built into the Extreme Points algorithm: the weight constraint and the product agglomeration constraint, the second only important for the pallet loading.

The product agglomeration constraint is treated, as described in section 4.5.3, by constructing products blocks with the single bin filling algorithm.

The weight constraint is introduced by defining a matrix relating the different products’ weight, in which the value $\xi$ represents a margin that allows heavier products to be piled on top of a lightest one without damaging them:

$$\phi_{ij} = \begin{cases} 1, & \omega_i - \omega_j \geq -\xi \\ 0, & \text{otherwise} \end{cases}$$

(5.7)

Admitting a value of $\xi = 1$ for the section 5.1 example it is possible to represent the feasible paths as shown in Figure 5.3.

So this new matrix becomes even one more update to the heuristic matrix, as it artificially excludes some of the paths from the overall solution. This way the new heuristic matrix becomes:

$$\eta_{ij} = \eta_{ij} \times \phi_{ij}, \quad i, j = 0, ..., n$$

(5.8)
5.3 The ant colony algorithm applied to the packing problems

The Bin Packing Problem and the Knapsack Problem have many similarities. If the number of bins available to a Knapsack Problem is more than the one necessary to pack all the boxes, then the goal becomes minimizing the number of bins used, which is the aim of a Bin Packing Problem.

In this section it is presented an Ant Colony Optimization algorithm that can easily be adapted to the Bin Packing Problem or the Knapsack problem by having an unlimited number of bins or a predefined number of bins.

5.3.1 The cost function

The cost function ought to be somehow related to the pheromones trail. Sometimes in ant colony optimization the cost function is equal, or inverse, to the factor used to update the pheromones trail. However in the exposed problem, an ant leaves a different trail of pheromones depending on the different bins which the nodes are associated to. So the cost function is given by the relation between the sum of bin percentage of occupied volume and the necessary number of bins.

\[ f = \frac{n_{\text{bins}}^p}{\sum_{k=1}^{n} \varphi^k} \]

\( \varphi^k \) – percentage of occupied volume in bin.

q,p – parameters to weight the importance of the percentage of occupied volume and the number of bins necessary respectively.

As the bins have all the same dimension, the sum of percentage of occupied volume is dependant from the number of used bins. Hence for this specific case the cost function could be reduced to the number of bins.
5.3.2 Adding an available space matrix

By filling the bin, the available space decreases, thus some blocks will not be able to be packed anymore.

In order to avoid so, another parameter was added as a matrix comparing directly the bin available space with the overall volume of the unpacked block of boxes. This would prevent having some blocks of boxes from running a heavy algorithm to try to stow them in an already filled bin, and also to jump to another empty bin when there was still available space for some smaller blocks of boxes.

In order to apply this it is considered the possibility of a block of boxes \( j \) being packed after a block of boxes \( i \) the available space matrix is defined by:

\[
\sigma_{ij}^k = \begin{cases} 
1, & F_i^k \geq V_j \\
0, & \text{otherwise}
\end{cases}
\]  

\( \sigma_{ij}^k \) – Available space matrix for bin \( k \);

\( F_i^k \) – Available volume in the bin \( k \) where the group of boxes \( i \) is;

\( V_j \) – Total volume of the block of boxes \( j \).

5.3.3 The path probability

The path probability becomes a conjunction of all the factors defined before; the pheromone trail, the heuristic matrix and the available space matrix. The available space matrix is not inserted on the heuristic matrix because it varies with the different bins, as the heuristic matrix varies with the ant. So the probability of a block of boxes \( j \) being chosen after a block of boxes \( i \) is:

\[
p_{ij}^k = \frac{\tau_{ij}^a \eta_{ij}^b \sigma_{ij}^k}{\sum_{j=1}^n \tau_{ij}^a \eta_{ij}^b \sigma_{ij}^k}, \quad j = 1, ..., n
\]  

\( p_{ij}^k \) – Probability of ant \( k \) going from node \( i \) to node \( j \);

In this way it is finally possible to describe the final algorithm using all the concepts previously enumerated as schematized in Figure 5.4.
Figure 5.4 - Ant Colony Optimization applied to Packing Algorithms
6 Results

The results presented on this chapter are divided into two phases, the benchmark problems, Bin Packing Problem and Container Loading Problem, and the INDASA problem.

The results obtained from the Benchmark tests were used to tune the parameters considered in the Ant Colony Optimization, mainly in the case of the Bin Packing Problem. The INDASA problem was used to validate the models algorithms already presented, as the real solution was available.

6.1 Introduction

In section 6.2 in order to be able to test the Ant Colony Optimization applied to the Extreme Points, it is necessary to perform a tuning of the many existing parameters for this algorithm. This was done using a Bin Packing Problem instance with fifty different boxes. All the problems were solved using the parameters obtained at this stage.

Having the set the parameters, the benchmark problems were the next stage, concerning the Bin Packing Problem, in section 6.3 and the Container Loading Problem, in section 6.4. For the Bin Packing Problem two classes were tested, each with two different numbers of boxes, being the goal of this problem packing the boxes into a minimal number of bins. As for the Container Loading Problem fifteen different instances were used. In this problem the objective is to maximize the percentage of occupied volume of a single available bin. Both problems results are then compared to other algorithm results.

Finally in section 6.5, the three developed algorithms are validated for the INDASA problem. To do this it is verified if it is possible to use the same number of pallets, or preferably even less, to stow all the products, concerning the pallet loading. Hence not only the Ant Colony Optimization Algorithm was used but also the First Fit Decreasing algorithm associated to the Extreme Points and the Layer Building algorithm were validated. It is also tested the simpler container loading, also using all three algorithms. As this problem is already optimized, the objective is to obtain the same packing as the real case.

In order to do all these tests it is necessary to define all the necessary instances, this is done in the three following sections.
6.1.1 Bin packing problem instances

In the case of the Bin Packing Problem the chosen instances are described on [Martello et al., 2000]. That work describes eight different classes, each one with different numbers of boxes. Due to the number of classes available, only two were chosen, the first and sixth class presented in the referred document, considering a number of boxes equal to 50 and 200 for each case. Thus to generate the ten instances for each class, the following definitions were used:

- **Class I:** \( W = H = D = 100 \), type 1 box with uniform distribution probability of 60%, type 2,3,4,5 boxes with uniform probability distribution of 10% each, where:
  - Type 1: \( w_j \) uniformly random in \( \left[ 1, \frac{1}{2} W \right] \), \( h_j \) uniformly random in \( \left[ \frac{2}{3} H, H \right] \) and \( d_j \) uniformly random in \( \left[ \frac{2}{3} D, D \right] \);
  - Type 2: \( w_j \) uniformly random in \( \left[ \frac{2}{3} W, W \right] \), \( h_j \) uniformly random in \( \left[ 1, \frac{1}{2} H \right] \) and \( d_j \) uniformly random in \( \left[ \frac{2}{3} D, D \right] \);
  - Type 3: \( w_j \) uniformly random in \( \left[ \frac{2}{3} W, W \right] \), \( h_j \) uniformly random in \( \left[ \frac{2}{3} H, H \right] \) and \( d_j \) uniformly random in \( \left[ 1, \frac{1}{2} D \right] \);
  - Type 4: \( w_j \) uniformly random in \( \left[ \frac{1}{2} W, W \right] \), \( h_j \) uniformly random in \( \left[ \frac{1}{2} H, H \right] \) and \( d_j \) uniformly random in \( \left[ \frac{1}{2} D, D \right] \);
  - Type 5: \( w_j \) uniformly random in \( \left[ 1, \frac{1}{2} W \right] \), \( h_j \) uniformly random in \( \left[ 1, \frac{1}{2} H \right] \) and \( d_j \) uniformly random in \( \left[ 1, \frac{1}{2} D \right] \);
- **Class VI:** \( w_j, h_j \) and \( d_j \) uniformly distributed between [1,10], and \( W = H = D = 10 \)

6.1.2 Container loading problem instances

The instances used to test the algorithm in the Container Loading Problem are fifteen problems from [Loh & Nee, 1992]. These problems vary in the size of the bins and in the number of types of boxes. These instances are clearly weakly heterogeneous and therefore similar to the model developed for INDASA. Some of the data regarding the Loh and Nee instances are presented in Table 6.1.
6.1.3 INDASA instances

The instances used are complete expedition lists and are previously divided between rolls products and non rolls products. In this way for each instance it is possible to have two sub instances. Hence the focus on each instance is not the overall number of pallets, but the number of pallets of rolls and non rolls.

For each instance there are also two different problems. The first problem and most complex is the pallet loading in which the pallets are considered to be the bin and the products. The second, the container loading, is the one in which the pallets are considered to be the boxes and the container the bin.

The first instance is based on the expedition list INDASAUSA, which is transported in a shipping container ISO 40 and the pallets for rolls and non rolls each have a specific dimension, indicated in Table 6.2. For the pallets case, the presented dimension is the external dimension. To determine the internal dimension, available space for the products, the physical pallets height has to be subtracted, approximately by 150mm.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Number of boxes</th>
<th>Number of different types of boxes</th>
</tr>
</thead>
<tbody>
<tr>
<td>LN1</td>
<td>100</td>
<td>7</td>
</tr>
<tr>
<td>LN2</td>
<td>200</td>
<td>8</td>
</tr>
<tr>
<td>LN3</td>
<td>200</td>
<td>8</td>
</tr>
<tr>
<td>LN4</td>
<td>100</td>
<td>7</td>
</tr>
<tr>
<td>LN5</td>
<td>120</td>
<td>6</td>
</tr>
<tr>
<td>LN6</td>
<td>200</td>
<td>8</td>
</tr>
<tr>
<td>LN7</td>
<td>200</td>
<td>8</td>
</tr>
<tr>
<td>LN8</td>
<td>130</td>
<td>6</td>
</tr>
<tr>
<td>LN9</td>
<td>200</td>
<td>8</td>
</tr>
<tr>
<td>LN10</td>
<td>250</td>
<td>8</td>
</tr>
<tr>
<td>LN11</td>
<td>100</td>
<td>6</td>
</tr>
<tr>
<td>LN12</td>
<td>120</td>
<td>6</td>
</tr>
<tr>
<td>LN13</td>
<td>130</td>
<td>7</td>
</tr>
<tr>
<td>LN14</td>
<td>120</td>
<td>6</td>
</tr>
<tr>
<td>LN15</td>
<td>250</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 6.1 - Container Loading Problem instances [Ratcliff, n.d.]

<table>
<thead>
<tr>
<th>Dimensions [depth width height] [mm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shipping container ISO 40</td>
</tr>
<tr>
<td>Rolls pallets</td>
</tr>
<tr>
<td>Non rolls pallets</td>
</tr>
<tr>
<td>[12020 2330 2280]</td>
</tr>
<tr>
<td>[1000 800 2250]</td>
</tr>
<tr>
<td>[1200 800 2250]</td>
</tr>
</tbody>
</table>

Table 6.2 – Container and pallets dimensions
The data related to the quantities and dimensions of products is defined in Table 6.3.

<table>
<thead>
<tr>
<th></th>
<th>Number of types of products</th>
<th>Number of products</th>
<th>Minimum volume dimensions [depth width height] [mm]</th>
<th>Maximum volume dimensions [depth width height] [mm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rolls</td>
<td>122</td>
<td>2194</td>
<td>[135 135 210]</td>
<td>[480 170 310]</td>
</tr>
<tr>
<td>Non rolls</td>
<td>27</td>
<td>2955</td>
<td>[110 110 70]</td>
<td>[500 500 115]</td>
</tr>
<tr>
<td>Total</td>
<td>149</td>
<td>5149</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6.3 - INDASAUSA instances

The second instance is named INDASAUSA2. The container and pallets dimensions are the same as in INDASAUSA and consequently the same as the ones referred in Table 6.3. The main differences are the type of products, which in this case are only non rolls and the quantities involved, which is described in Table 6.4, which makes the container loading simpler.

<table>
<thead>
<tr>
<th></th>
<th>Number of types of products</th>
<th>Number of products</th>
<th>Minimum volume dimensions [depth width height] [mm]</th>
<th>Maximum volume dimensions [depth width height] [mm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non rolls</td>
<td>165</td>
<td>3386</td>
<td>[335 175 152]</td>
<td>[300 225 270]</td>
</tr>
</tbody>
</table>

Table 6.4 - INDASAUSA2 instance

In order to validate the results for the proposed algorithms, it is necessary to know the exact number of pallets that was used for each instance both for the rolls and the non rolls, which is presented in Table 6.5. In the pallet loading case, these numbers are the maximum number of bins to be used and in the container loading case is the number of boxes to be loaded into one bin.

<table>
<thead>
<tr>
<th></th>
<th>Number of pallets</th>
<th>INDASAUSA</th>
<th>INDASAUSA2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rolls</td>
<td>6</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Non rolls</td>
<td>20</td>
<td>24</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.5 - INDASA results
6.2 Parameter tuning for the ant colony optimization

Since the Ant Colony Optimization applied to the Extreme Points has a great number of parameters it is necessary to tune them. In a first step Class I and Class VI were used, both with fifty boxes, allowing in this way a faster simulation than with two hundred boxes. After having all the parameters tuned for those two instances it was decided to test the number of iterations and ants necessary to run the two hundred boxes instances. In this way a second test was run changing only these two parameters. The procedure consists of the following steps:

1. Consider $\alpha, \beta, q, p = 1, \rho = 0.1$ and five ants. Then run simulations starting with twenty iterations and increase this number until the cost function became stable, considering this the number of iterations to be used;

2. Consider $\alpha, \beta, q, p = 1, \rho = 0.1$. Then test it with ten and twenty ants and evaluate if there is an improvement on the result or in the iteration in which the cost function becomes stable. If that happens consider the new number of ants and a new number of iterations;

3. Consider $\alpha, \beta, q, p = 1$. Change the value of $\rho$ to 0.2 and 0.3 and look for improvements. If that happens consider the new value of $\rho$ and possibly in the number of iterations;

4. Consider $q, p = 1$. Test the pair $\alpha, \beta$ with values of $[1, 2], [2, 1], [1, 5]$ and $[5, 1]$. Choose the better pair of values and eventually change the number of iterations;

5. Test the pair $q, p$ with values of $[1, 2], [2, 1], [1, 5]$ and $[5, 1]$. Choose the better pair of values and eventually change the number of iterations.

The necessary number of bins is used to choose the best parameter in each step. If in one step the number of bins is always the same, it is chosen the parameter that allows reducing the number of iterations.

Additionally it was necessary to define the Extreme Points sorting, so they were ordered first by lower z value, then x value and then y value. Also the rows of identical products were always built along the x axis. These two factors are not only used in the tuning but in the problems in which the Extreme Points are involved.
6.2.1 Class I with fifty boxes - Step 1: number of iterations

In this section the parameters are tuned for Class I with fifty boxes using the previously described protocol in section 6.2. The tests are represented by two digits; the first represents the step and the second the test number concerning that test.

<table>
<thead>
<tr>
<th>Test</th>
<th>Number of ants</th>
<th>Number of iterations</th>
<th>$\rho$</th>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$P$</th>
<th>$q$</th>
<th>Number of bins</th>
<th>Stabilization iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>5</td>
<td>20</td>
<td>0,1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>1.2</td>
<td>5</td>
<td>40</td>
<td>0,1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>14</td>
<td>40</td>
</tr>
<tr>
<td>1.3</td>
<td>5</td>
<td>60</td>
<td>0,1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>14</td>
<td>60</td>
</tr>
<tr>
<td>1.4</td>
<td>5</td>
<td>80</td>
<td>0,1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>14</td>
<td>68</td>
</tr>
<tr>
<td>1.5</td>
<td>5</td>
<td>100</td>
<td>0,1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>14</td>
<td>78</td>
</tr>
</tbody>
</table>

Table 6.6 - Parameters tuning, step 1 - Class I, 50 boxes

So in the first step, for the Class I with fifty boxes, Table 6.6, is possible to notice that the number of bins is the same for four tests. In this case it is chosen the one that takes less time to stabilize, which happens around iteration sixty eight for test 1.4.

6.2.2 Class I with fifty boxes - Step 2: number of ants

In this step were tested ten and twenty ants. Initially eighty were the number of iterations used.

<table>
<thead>
<tr>
<th>Test</th>
<th>Number of ants</th>
<th>Number of iterations</th>
<th>$\rho$</th>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$P$</th>
<th>$q$</th>
<th>Number of bins</th>
<th>Stabilization iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.4</td>
<td>5</td>
<td>80</td>
<td>0,1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>14</td>
<td>68</td>
</tr>
<tr>
<td>2.1</td>
<td>10</td>
<td>80</td>
<td>0,1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>60</td>
</tr>
<tr>
<td>2.2</td>
<td>10</td>
<td>100</td>
<td>0,1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>76</td>
</tr>
<tr>
<td>2.3</td>
<td>20</td>
<td>80</td>
<td>0,1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>73</td>
</tr>
<tr>
<td>2.4</td>
<td>20</td>
<td>100</td>
<td>0,1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>86</td>
</tr>
</tbody>
</table>

Table 6.7 - Parameters tuning, step 2 - Class I, 50 boxes

Evaluating Table 6.7 it is possible to notice, in test 2.1, that the improvement in the number of ants provides better results. Nevertheless increasing from ten to twenty ants does not improve the results. Tests 2.2 and 2.4 were implemented since the stabilization iteration was near the number of iterations. From Figure 6.1, it is possible to watch that the cost function oscillates between two values, result of the metaheuristic algorithm, trying to find a better solution through a worst one. In that case it was considered the stabilization iteration as the one in which the lower value for the cost function was reached the second time.
6.2.3 Class I with fifty boxes - Step 3: evaporation factor

In this step was considered the number of ants to be 10, and the number of iterations 80. It was tested an evaporation factor of 0.2 and 0.3.

<table>
<thead>
<tr>
<th>Test</th>
<th>Number of ants</th>
<th>Number of iterations</th>
<th>( \rho )</th>
<th>( \alpha )</th>
<th>( \beta )</th>
<th>( \rho )</th>
<th>( q )</th>
<th>Number of bins</th>
<th>Stabilization iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>10</td>
<td>80</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>60</td>
</tr>
<tr>
<td>3.1</td>
<td>10</td>
<td>80</td>
<td>0.2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>47</td>
</tr>
<tr>
<td>3.2</td>
<td>10</td>
<td>80</td>
<td>0.3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>52</td>
</tr>
</tbody>
</table>

Table 6.8 - Parameters tuning, step 3 - Class I, 50 boxes

Having completed step three it is possible to notice the importance of the evaporation factor. The quality of the result is the same even with the decrease of the number of necessary iterations when comparing test 3.1 with test 2.1. So from here on it will be considered only sixty iterations with an evaporation factor of 0.2. When testing the 0.3 evaporation factor more than one time it proved to sometimes stabilize in worse values.
6.2.4 Class I with fifty boxes - Step 4: alfa and beta parameters

In this section, having set the first three parameters, it is tested the relation between $\alpha$ and $\beta$ by setting values described in section 6.2.

<table>
<thead>
<tr>
<th>Test</th>
<th>Number of ants</th>
<th>Number of iterations</th>
<th>$\rho$</th>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$p$</th>
<th>$q$</th>
<th>Number of bins</th>
<th>Stabilization iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>10</td>
<td>80</td>
<td>0.2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>47</td>
</tr>
<tr>
<td>4.1</td>
<td>10</td>
<td>60</td>
<td>0.2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>34</td>
</tr>
<tr>
<td>4.2</td>
<td>10</td>
<td>60</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>20</td>
</tr>
<tr>
<td>4.3</td>
<td>10</td>
<td>60</td>
<td>0.2</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>54</td>
</tr>
<tr>
<td>4.4</td>
<td>10</td>
<td>60</td>
<td>0.2</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>14</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 6.9 - Parameters tuning, step 4 - Class I, 50 boxes

Analyzing the Table 6.9, it is possible to notice the importance of the factors $\alpha, \beta$. It becomes apparent that for high values of $\beta$ the solution quickly converges to a minimal value that it is not always the best. With multiple tests it was possible to notice that many times it converged to higher values, even if in Figure 6.3 the minimal is reached. So the best pair of values seems to be $\alpha = 1, \beta = 2$, which means that the pheromone trail is more important than the heuristic matrix. This also allows decreasing the number of necessary iterations to 40.
By having decreased the number of necessary iterations to forty and all the other parameters tuned, it is finally time to tune the parameters \( q \) and \( p \), according to the procedure in section 6.2.

In Table 6.10, shows that the values of the pair \( p,q \) on the result is negligible. The decision of choosing the pair is to be done after running the same tuning for an instance of class VI with fifty boxes. In this case it is not shown a graph since this pair directly affects the cost function and so it is not possible to compare all the results in a graph.

### Table 6.10 - Parameters tuning, step 5 - Class I, 50 boxes

<table>
<thead>
<tr>
<th>Test</th>
<th>Number of ants</th>
<th>Number of iterations</th>
<th>( \rho )</th>
<th>( \alpha )</th>
<th>( \beta )</th>
<th>( p )</th>
<th>( q )</th>
<th>Number of bins</th>
<th>Stabilization iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>10</td>
<td>60</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>20</td>
</tr>
<tr>
<td>5.1</td>
<td>10</td>
<td>40</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>13</td>
<td>23</td>
</tr>
<tr>
<td>5.2</td>
<td>10</td>
<td>40</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>13</td>
<td>25</td>
</tr>
<tr>
<td>5.3</td>
<td>10</td>
<td>40</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>13</td>
<td>21</td>
</tr>
<tr>
<td>5.4</td>
<td>10</td>
<td>40</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>13</td>
<td>25</td>
</tr>
</tbody>
</table>

Figure 6.3 - Parameters tuning, step 4 - Class I, 50 boxes

6.2.5 Class I with fifty boxes - Step 5: \( q \) and \( p \) parameters
6.2.6 Class VI with fifty boxes – Complete procedure

In this case the class VI with fifty boxes instance is tested. Instead of doing a step by step evaluation as for the previous case, it is only evaluated if the test matches the previous tests done to class I.

<table>
<thead>
<tr>
<th>Test</th>
<th>Number of ants</th>
<th>Number of iterations</th>
<th>$\rho$</th>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\rho$</th>
<th>$q$</th>
<th>Number of bins</th>
<th>Stabilization iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>5</td>
<td>20</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>14</td>
<td>20</td>
</tr>
<tr>
<td>1.2</td>
<td>5</td>
<td>40</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>14</td>
<td>40</td>
</tr>
<tr>
<td>1.3</td>
<td>5</td>
<td>60</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>69</td>
</tr>
<tr>
<td>1.4</td>
<td>5</td>
<td>80</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>63</td>
</tr>
<tr>
<td>1.5</td>
<td>5</td>
<td>100</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>62</td>
</tr>
<tr>
<td>2.2</td>
<td>10</td>
<td>100</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>52</td>
</tr>
<tr>
<td>2.4</td>
<td>20</td>
<td>100</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>60</td>
</tr>
<tr>
<td>3.1</td>
<td>10</td>
<td>80</td>
<td>0.2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>26</td>
</tr>
<tr>
<td>3.2</td>
<td>10</td>
<td>80</td>
<td>0.3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>23</td>
</tr>
<tr>
<td>4.1</td>
<td>10</td>
<td>60</td>
<td>0.2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>34</td>
</tr>
<tr>
<td>4.2</td>
<td>10</td>
<td>60</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>4.3</td>
<td>10</td>
<td>60</td>
<td>0.2</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>56</td>
</tr>
<tr>
<td>4.4</td>
<td>10</td>
<td>60</td>
<td>0.2</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td>5.1</td>
<td>10</td>
<td>40</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>12</td>
<td>19</td>
</tr>
<tr>
<td>5.2</td>
<td>10</td>
<td>40</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>12</td>
<td>18</td>
</tr>
<tr>
<td>5.3</td>
<td>10</td>
<td>40</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>12</td>
<td>28</td>
</tr>
<tr>
<td>5.4</td>
<td>10</td>
<td>40</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>12</td>
<td>23</td>
</tr>
</tbody>
</table>

Table 6.11 - Parameters tuning - Class VI, 50 boxes

With the analysis of Table 6.11, it is possible to support the results obtained previously with the analysis of class I. To point out the fact that for higher values of $\beta$ the solution quickly converges to a minimal, nevertheless the possibility of converging to a worse value is kept. In addition combining the two sets of tests allows to define a pair of values $\rho = 1$ and $q = 2$. This difference between results becomes from the fact that the number of necessary bins and the sum of percentage of occupied volume are dependant.
6.2.7 Class I and VI with two hundred boxes – Number of ants and iterations tuning

This section compares the results obtained with larger instances, in this case Class I and Class VI with two hundred boxes. In this way it is possible to estimate the necessary number iterations and ants, considering that the remaining parameters are kept.

<table>
<thead>
<tr>
<th>Test</th>
<th>Number of ants</th>
<th>Number of iterations</th>
<th>( \rho )</th>
<th>( \alpha )</th>
<th>( \beta )</th>
<th>( \rho )</th>
<th>( q )</th>
<th>Number of bins</th>
<th>Stabilization iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>10</td>
<td>40</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>57</td>
<td>20</td>
</tr>
<tr>
<td>1.2</td>
<td>10</td>
<td>80</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>56</td>
<td>49</td>
</tr>
<tr>
<td>1.3</td>
<td>10</td>
<td>100</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>56</td>
<td>51</td>
</tr>
<tr>
<td>2.1</td>
<td>20</td>
<td>60</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>56</td>
<td>54</td>
</tr>
</tbody>
</table>

Table 6.12 - Parameters tuning - Class I, 200 boxes

From the results of Table 6.12, it is possible to understand that are necessary less than sixty iterations to have a minimal. It is also noticeable that increasing the number of ants from ten to twenty does not improve the solution.

<table>
<thead>
<tr>
<th>Test</th>
<th>Number of ants</th>
<th>Number of iterations</th>
<th>( \rho )</th>
<th>( \alpha )</th>
<th>( \beta )</th>
<th>( \rho )</th>
<th>( q )</th>
<th>Number of bins</th>
<th>Stabilization iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>10</td>
<td>60</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>38</td>
<td>56</td>
</tr>
<tr>
<td>1.2</td>
<td>10</td>
<td>80</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>37</td>
<td>70</td>
</tr>
<tr>
<td>1.3</td>
<td>10</td>
<td>100</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>37</td>
<td>93</td>
</tr>
</tbody>
</table>

Table 6.13 - Parameters tuning - Class VI, 200 boxes

From class VI, Table 6.13, it is possible to notice that sixty iterations may not be enough to have a good solution. For an instance with two hundred boxes, in this case represent two hundred nodes, the ten ants are kept but the number of iterations increases from forty, in the fifty boxes case, to one hundred.

The tuned parameters are presented in Table 6.14.

<table>
<thead>
<tr>
<th>Number of ants</th>
<th>Number of iterations</th>
<th>( \rho )</th>
<th>( \alpha )</th>
<th>( \beta )</th>
<th>( \rho )</th>
<th>( q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>variable</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 6.14 - Ant Colony Optimization tuned parameters

63
6.3 Bin packing problem

In order to compare the values obtained with other algorithms, three runs were done for each instance. The standard deviation is also presented in order to reflect the difference of the instances that are generated randomly, and those results are presented in Table 6.15, Table 6.16, Table 6.17 and Table 6.18. The average is computed in order to compare to the remaining algorithms presented in the literature and between parentheses is shown the minimal value.

Clearly both Classes are strongly heterogeneous sets of boxes, since all boxes have different dimensions. This way the adaptations, regarding the existence of blocks of boxes of the same type for the agglomeration constraint, are not used. Thus those adaptations become additional computational effort, hence increasing the time necessary to find the solution.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>11</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>12</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>15</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

Time [s] 69,1
Standard Deviation 1,48
Average 13,2 (11)

Table 6.15 - Number of bins necessary for each instance of Class I with 50 boxes

<table>
<thead>
<tr>
<th>Instance</th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>52</td>
<td>53</td>
<td>53</td>
<td>52</td>
</tr>
<tr>
<td>3</td>
<td>48</td>
<td>48</td>
<td>49</td>
<td>48</td>
</tr>
<tr>
<td>4</td>
<td>54</td>
<td>54</td>
<td>53</td>
<td>53</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>53</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>6</td>
<td>56</td>
<td>55</td>
<td>56</td>
<td>55</td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>49</td>
<td>50</td>
<td>49</td>
</tr>
<tr>
<td>8</td>
<td>52</td>
<td>52</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>9</td>
<td>53</td>
<td>53</td>
<td>54</td>
<td>53</td>
</tr>
<tr>
<td>10</td>
<td>53</td>
<td>53</td>
<td>53</td>
<td>53</td>
</tr>
</tbody>
</table>

Time [s] 1045
Standard Deviation 2,25
Average 52,2 (48)

Table 6.16 - Number of bins necessary for each instance of Class I with 200 boxes
The results presented in Table 6.16 are a problem as the process time exceeds the 1000 seconds allowed in the literature. However considering the average value per iteration, 104.5 seconds, and the value of an iteration in which a minimum was reached, iteration 95 in Figure 6.4, it is possible to say that the last minimum was reached around 993 seconds guaranteeing the limit defined for algorithms used for comparison.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>11</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>12</td>
<td>13</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>13</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 6.17 - Number of bins necessary for each instance of Class VI with 50 boxes
### Table 6.18 - Number of bins necessary for each instance of Class VI with 200 boxes

<table>
<thead>
<tr>
<th>Instance</th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Minimum</th>
<th>Time [s]</th>
<th>Standard Deviation</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>42</td>
<td>42</td>
<td>41</td>
<td>41</td>
<td>42</td>
<td>978</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>43</td>
<td>43</td>
<td>44</td>
<td>43</td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>41</td>
<td>42</td>
<td>41</td>
<td>41</td>
<td>41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>38</td>
<td>38</td>
<td>38</td>
<td>38</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>46</td>
<td>47</td>
<td>45</td>
<td>45</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>39</td>
<td>39</td>
<td>40</td>
<td>39</td>
<td>39</td>
<td>3,21</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>40</td>
<td>38</td>
<td>39</td>
<td>38</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>48</td>
<td>50</td>
<td>48</td>
<td>48</td>
<td>48</td>
<td></td>
<td>41,9 (38)</td>
</tr>
</tbody>
</table>

#### 6.3.1 Comparison with other algorithms

In order to compare the proposed algorithm with other algorithms, six different cases were used:

- a two-level Branch and Bound (MPV) [Martello et al., 2000];
- the third constructive heuristic described in section 2.3 (MPV-BS) [Martello et al., 2000];
- a Guided Local Search (GLS) [Faroe et al., 2003];
- a theoretical Lower Bound (LB) [Boschetti & Mingozzi, 2004];
- an Extreme Points Heuristic (C-EPBFD) [Crainic et al., 2008];
- a two-level tabu-search (TS-Pack) [Crainic et al., 2009].

<table>
<thead>
<tr>
<th>Class</th>
<th>Number of boxes</th>
<th>MPV (1000 s)</th>
<th>MPV-BS</th>
<th>GLS (1000s)</th>
<th>LB</th>
<th>C-EPBFD</th>
<th>TS-Pack</th>
<th>ACO-EP</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>50</td>
<td>13,6</td>
<td>13,5</td>
<td>13,4</td>
<td>12,9</td>
<td>13,7</td>
<td>13,4</td>
<td>13,2</td>
</tr>
<tr>
<td>I</td>
<td>200</td>
<td>52,3</td>
<td>52,3</td>
<td>51,2</td>
<td>49,7</td>
<td>51,9</td>
<td>51,1</td>
<td>52,3</td>
</tr>
<tr>
<td>VI</td>
<td>50</td>
<td><strong>9.8</strong></td>
<td>11</td>
<td><strong>9.8</strong></td>
<td>9,4</td>
<td>10,1</td>
<td><strong>9.8</strong></td>
<td>11,2</td>
</tr>
<tr>
<td>VI</td>
<td>200</td>
<td>38,2</td>
<td>40,8</td>
<td><strong>37.7</strong></td>
<td>36,7</td>
<td>38,5</td>
<td><strong>37.7</strong></td>
<td>41,9</td>
</tr>
</tbody>
</table>

Table 6.19 - State of the art versus ACO-EP
As it is possible to evaluate from Table 6.19 and Figure 6.5 the Ant Colony Optimization applied to the Extreme Points algorithm presents a tool that can compete with the state of the art in terms of results in class I. However in class VI the results are not so good, maybe for lack of iterations, or because the main tuning was done for a Class I instance. Nevertheless these results are only a guideline, as the instances generated for this problem are different from the ones used for the state of the art tests. Additionally it would be interesting to compare all the eight classes described in [Martello et al., 2000]. The Lower Bound is not considered as the best result as it is only a theoretical result.

![Figure 6.5 - State of the Art versus ACO-EP](image)

6.4 Container Loading Problem

For this problem, there may be types of boxes that are not completely stowed, which led to slight changes in the model to allow it. As the LN instances are supposed to fill a single container, the variables $p$ and $q$ were discarded as there it is not necessary to compare the number of filled bins, only one, and so the percentage of occupied volume becomes the only important data.

Admitting that the tuning done for the bin packing problem may be used in this case, those parameters are kept. As in this case the number of nodes does not depend on the number of boxes, but on the number of types of boxes, the number of necessary iterations must be less than twenty. As no test was done to backup this information, or the assumption that the parameters can be kept, the evolution of the cost function is followed to watch if it really stabilizes.
The first set of tests was done in order to keep the stability constraint. For the cases in which it was not possible to pack all the boxes it was done another set of tests without the stability constraint. For those instances where the solution was found before the number of iterations limit, the iteration in which the solution was found is the one shown. In most cases when that happens are simple packing problems and so independently from the order, a feasible packing containing all the boxes is achieved. Doing so most times only one iteration is used to complete the packing. However analyzing the Table 6.21, it is possible to notice that the instance LN7 which usually is complex [Wang et al., 2008], was solved in one instance. This can happen due to the stochastic nature of the Ant Colony Optimization procedure.

Rotating the bin 90°, switch the width and depth, can provide a different solution. Thus for complex problems, and since only one bins is considered, the bin is rotated to look for a better solution.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Number of iterations</th>
<th>Rotated bin</th>
<th>Unpacked boxes</th>
<th>Percentage of occupied volume</th>
<th>Time [s]</th>
<th>Time per iteration [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>LN1</td>
<td>1</td>
<td>no</td>
<td>0</td>
<td>62,5%</td>
<td>1,1</td>
<td>1,1</td>
</tr>
<tr>
<td>LN2</td>
<td>20</td>
<td>no</td>
<td>50</td>
<td>87,5%</td>
<td>260,1</td>
<td>13,0</td>
</tr>
<tr>
<td>LN2</td>
<td>20</td>
<td>yes</td>
<td>56</td>
<td>88,9%</td>
<td>387,9</td>
<td>19,4</td>
</tr>
</tbody>
</table>

Table 6.20 - Container Loading Problem results with stability constraint

Analyzing the evolution of the LN2 instances, Figure 6.6, it is possible to notice that the cost function convergence was very fast, and by doing information is being lost. In order to prevent it, the evaporation coefficient was changed to 0.1. As it can be seen by comparing Table 6.20 and Table 6.21, it was possible obtain better results for the instance LN2.
Having all the instances tested with the stability constraint, it was decided to test the non optimal cases by removing this constraint, since most algorithms in the literature don’t use it, thus trying to obtain a better solution. The results are shown in Table 6.22.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Number of iterations</th>
<th>Rotated bin</th>
<th>Unpacked boxes</th>
<th>Percentage of occupied volume</th>
<th>Time [s]</th>
<th>Time per iteration [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>LN2</td>
<td>20</td>
<td>no</td>
<td>30</td>
<td>89,5%</td>
<td>248,3</td>
<td>12,4</td>
</tr>
<tr>
<td>LN2</td>
<td>20</td>
<td>yes</td>
<td>49</td>
<td>89,0%</td>
<td>607,6</td>
<td>30,4</td>
</tr>
<tr>
<td>LN3</td>
<td>1</td>
<td>no</td>
<td>0</td>
<td>53,4%</td>
<td>1,9</td>
<td>1,9</td>
</tr>
<tr>
<td>LN4</td>
<td>1</td>
<td>no</td>
<td>0</td>
<td>55,0%</td>
<td>0,6</td>
<td>0,6</td>
</tr>
<tr>
<td>LN5</td>
<td>2</td>
<td>no</td>
<td>0</td>
<td>77,2%</td>
<td>2,1</td>
<td>2,1</td>
</tr>
<tr>
<td>LN6</td>
<td>20</td>
<td>no</td>
<td>64</td>
<td>88,1%</td>
<td>268,3</td>
<td>13,4</td>
</tr>
<tr>
<td>LN6</td>
<td>20</td>
<td>yes</td>
<td>48</td>
<td>87,7%</td>
<td>428,3</td>
<td>21,4</td>
</tr>
<tr>
<td>LN7</td>
<td>1</td>
<td>no</td>
<td>0</td>
<td>87,7%</td>
<td>3,2</td>
<td>3,2</td>
</tr>
<tr>
<td>LN8</td>
<td>1</td>
<td>no</td>
<td>0</td>
<td>59,4%</td>
<td>0,6</td>
<td>0,6</td>
</tr>
<tr>
<td>LN9</td>
<td>1</td>
<td>no</td>
<td>0</td>
<td>61,9%</td>
<td>7,2</td>
<td>7,2</td>
</tr>
<tr>
<td>LN10</td>
<td>1</td>
<td>no</td>
<td>0</td>
<td>67,3%</td>
<td>1,6</td>
<td>1,6</td>
</tr>
<tr>
<td>LN11</td>
<td>1</td>
<td>no</td>
<td>0</td>
<td>62,2%</td>
<td>0,6</td>
<td>0,6</td>
</tr>
<tr>
<td>LN12</td>
<td>4</td>
<td>no</td>
<td>0</td>
<td>78,5%</td>
<td>22,3</td>
<td>5,6</td>
</tr>
<tr>
<td>LN13</td>
<td>20</td>
<td>no</td>
<td>15</td>
<td>80,1%</td>
<td>153,6</td>
<td>7,7</td>
</tr>
<tr>
<td>LN13</td>
<td>20</td>
<td>yes</td>
<td>14</td>
<td>81,7%</td>
<td>390,3</td>
<td>19,5</td>
</tr>
<tr>
<td>LN14</td>
<td>1</td>
<td>no</td>
<td>0</td>
<td>62,8%</td>
<td>0,7</td>
<td>0,7</td>
</tr>
<tr>
<td>LN15</td>
<td>1</td>
<td>no</td>
<td>0</td>
<td>59,5%</td>
<td>1,6</td>
<td>1,6</td>
</tr>
</tbody>
</table>

Table 6.21 - Container Loading Problem results with stability constraint

<table>
<thead>
<tr>
<th>Instance</th>
<th>Number of iterations</th>
<th>Rotated bin</th>
<th>Unpacked boxes</th>
<th>Percentage of occupied volume</th>
<th>Time [s]</th>
<th>Time per iteration [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>LN2</td>
<td>20</td>
<td>no</td>
<td>45</td>
<td>91,4%</td>
<td>220,0</td>
<td>11,0</td>
</tr>
<tr>
<td>LN2</td>
<td>20</td>
<td>yes</td>
<td>52</td>
<td>93,3%</td>
<td>219,9</td>
<td>11,0</td>
</tr>
<tr>
<td>LN6</td>
<td>20</td>
<td>no</td>
<td>32</td>
<td>88,1%</td>
<td>264,7</td>
<td>13,2</td>
</tr>
<tr>
<td>LN6</td>
<td>20</td>
<td>yes</td>
<td>33</td>
<td>89,8%</td>
<td>262,7</td>
<td>13,1</td>
</tr>
<tr>
<td>LN13</td>
<td>20</td>
<td>no</td>
<td>7</td>
<td>80,9%</td>
<td>147,6</td>
<td>7,4</td>
</tr>
<tr>
<td>LN13</td>
<td>20</td>
<td>yes</td>
<td>14</td>
<td>82,0%</td>
<td>118,8</td>
<td>5,9</td>
</tr>
</tbody>
</table>

Table 6.22 - Container Loading Problem results without stability constraint
6.4.1 Comparison with other algorithms

In order to compare with other algorithms, four different methods were used in the comparison:

- a tertiary-tree-based dynamic space decomposition (TTTDSD) [Wang et al., 2008];
- a Grasp approach (GRModGrasp) [Moura & Oliveira, 2005];
- a hybrid genetic algorithm (CBGAS) [Bortfeldt & Gehring, 2001];
- a parallel tabu search (CBUSE) [Bortfeldt et al., 2003].

Analyzing Table 6.23, it is possible to notice, by the average value, that the developed algorithm values are not much inferior to other algorithms results. The worst results for LN2, LN6 and LN13 can be explained by the algorithm structure, since it tries to pack groups of boxes instead of box by box. That leads to under packing and the solution would be modify the algorithm so it could exist mixing of different types of boxes. The values between parentheses represent the number of unpacked boxes, this value shows that not always having a maximum number of packed boxes presents the better solution.

<table>
<thead>
<tr>
<th>Instance</th>
<th>TTTDSD</th>
<th>GRModGrasp</th>
<th>CBGAS</th>
<th>CBUSE</th>
<th>ACO-EP</th>
</tr>
</thead>
<tbody>
<tr>
<td>LN1</td>
<td>62,5</td>
<td>62,5</td>
<td>62,5</td>
<td>62,5</td>
<td>62,5</td>
</tr>
<tr>
<td>LN2</td>
<td>90,7(35)</td>
<td>92,6(19)</td>
<td>89,8(51)</td>
<td>96,7(28)</td>
<td>93,3(52)</td>
</tr>
<tr>
<td>LN3</td>
<td>53,4</td>
<td>53,4</td>
<td>53,4</td>
<td>53,4</td>
<td>53,4</td>
</tr>
<tr>
<td>LN4</td>
<td>55,0</td>
<td>55,0</td>
<td>55,0</td>
<td>55,0</td>
<td>55,0</td>
</tr>
<tr>
<td>LN5</td>
<td>77,2</td>
<td>77,2</td>
<td>77,2</td>
<td>77,2</td>
<td>77,2</td>
</tr>
<tr>
<td>LN6</td>
<td>92,9(37)</td>
<td>91,7(28)</td>
<td>92,4(45)</td>
<td>96,3(40)</td>
<td>89,8(33)</td>
</tr>
<tr>
<td>LN7</td>
<td>84,7</td>
<td>84,7</td>
<td>84,7</td>
<td>84,7</td>
<td>84,7</td>
</tr>
<tr>
<td>LN8</td>
<td>59,4</td>
<td>59,4</td>
<td>59,4</td>
<td>59,4</td>
<td>59,4</td>
</tr>
<tr>
<td>LN9</td>
<td>61,9</td>
<td>61,9</td>
<td>61,9</td>
<td>61,9</td>
<td>61,9</td>
</tr>
<tr>
<td>LN10</td>
<td>67,3</td>
<td>67,3</td>
<td>67,3</td>
<td>67,3</td>
<td>67,3</td>
</tr>
<tr>
<td>LN11</td>
<td>62,2</td>
<td>62,2</td>
<td>62,2</td>
<td>62,2</td>
<td>62,2</td>
</tr>
<tr>
<td>LN12</td>
<td>78,5</td>
<td>78,5</td>
<td>78,5</td>
<td>78,5</td>
<td>78,5</td>
</tr>
<tr>
<td>LN13</td>
<td>85,6</td>
<td>85,6</td>
<td>85,6</td>
<td>85,6</td>
<td>82,0(40)</td>
</tr>
<tr>
<td>LN14</td>
<td>62,8</td>
<td>62,8</td>
<td>62,8</td>
<td>62,8</td>
<td>62,8</td>
</tr>
<tr>
<td>LN15</td>
<td>59,5</td>
<td>59,5</td>
<td>59,5</td>
<td>59,5</td>
<td>59,5</td>
</tr>
<tr>
<td>Average</td>
<td>70,2</td>
<td>70,3</td>
<td>70,1</td>
<td>70,9</td>
<td>70,0</td>
</tr>
</tbody>
</table>

Table 6.23 State of the art versus ACO-EP
6.5 INDASA results

In order to obtain the results for the INDASA instances, the three developed algorithms were used. The first algorithm to be tested is the Layer Building Algorithm, then the Extreme Points First Fit Decreasing heuristic and finally the Ant Colony Optimization applied to the Extreme Points.

At the end all the results obtained are compared and a general overview is done for both the pallet loading and the container loading.

6.5.1 Layer building heuristic results

The results obtained with this algorithm present certain questions concerning the constraints as already stated on section 4.2.1, the fact of some pallets being under packed, and more important if the stability is kept. Table 6.24 presents the results for this problem using the agglomeration constraint.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Products type</th>
<th>Number of pallets</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDASAUSA</td>
<td>Non rolls</td>
<td>19</td>
<td>24.7</td>
</tr>
<tr>
<td>INDASAUSA</td>
<td>Rolls</td>
<td>5</td>
<td>26.5</td>
</tr>
<tr>
<td>INDASAUSA2</td>
<td>Non rolls</td>
<td>22</td>
<td>51.0</td>
</tr>
</tbody>
</table>

Table 6.24 - Layer Building Heuristic results

As in this algorithm the agglomeration constraint was an addition to the main algorithm it is possible to have results without it. So it is interesting to know the number of necessary pallets without obeying to that constraint, those results are presented on Table 6.25.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Products type</th>
<th>Number of pallets</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDASAUSA</td>
<td>Non rolls</td>
<td>16</td>
<td>62.2</td>
</tr>
<tr>
<td>INDASAUSA</td>
<td>Rolls</td>
<td>4</td>
<td>33.3</td>
</tr>
<tr>
<td>INDASAUSA2</td>
<td>Non rolls</td>
<td>20</td>
<td>40.6</td>
</tr>
</tbody>
</table>

Table 6.25 - Layer Building Heuristic results without agglomeration constraint

Comparing both results, Figure 6.7, it is possible to notice that the agglomeration constraint increases the number of necessary pallets, which is natural due to the complexity of the problem.

Nevertheless this algorithm is not sufficient to solve the container loading independently of the order in which the pallets are provided. Specifically, for a given set of pallets to be packed it must have a specific order to successfully obtain the optimal packing.
6.5.2 Extreme points heuristic results

Two tests were run using the Extreme Points First Fit Decreasing, with two different values of the percentage of the minimal dimension of the box to be packed, as defined in the section 4.3.4. In the first case it was admitted a percentage of forty percent to guarantee the stability constraint. In the second case it was admitted a zero percent tolerance, to have the total support for each box. The results are presented on Table 6.26. By comparing them it is possible to notice that if the products do not need to be totally supported the results improve, as shown in Figure 6.8.

Due to the direction in which the rows of products are constructed, rotating the bin 90º may affect the packing obtained using this algorithm.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Type</th>
<th>Number of Pallets</th>
<th>% minimal dimension</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDASAUSA</td>
<td>Non rolls</td>
<td>18</td>
<td>40</td>
<td>121,8</td>
</tr>
<tr>
<td>INDASAUSA</td>
<td>Rolls</td>
<td>5</td>
<td>40</td>
<td>386,5</td>
</tr>
<tr>
<td>INDASAUSA2</td>
<td>Non rolls</td>
<td>21</td>
<td>40</td>
<td>197,1</td>
</tr>
<tr>
<td>INDASAUSA</td>
<td>Non rolls</td>
<td>19</td>
<td>0</td>
<td>121,0</td>
</tr>
<tr>
<td>INDASAUSA</td>
<td>Rolls</td>
<td>5</td>
<td>0</td>
<td>845,7</td>
</tr>
<tr>
<td>INDASAUSA2</td>
<td>Non rolls</td>
<td>23</td>
<td>0</td>
<td>179,5</td>
</tr>
</tbody>
</table>

Table 6.26 - Extreme Points Heuristic results
In this case the container loading proved to be feasible independently of the order of the pallets. However the algorithm must be run using container rotated in order to succeed. Hence considering the first instance, in which there are two different types of pallets to be loaded into the container, if the rolls pallets are to be packed first, the optimal packing is not reached. However, by switching the width and depth of the container it is possible to obtain the optimal packing. It is important to highlight that, even doing the packing with both orders, due to the simplicity of the problem, the time spent is equal to 0.42 seconds, which is negligible.

![Extreme Points First Fit Decreasing](image)

**Figure 6.8 - Extreme Points First Fit Decreasing heuristic**

### 6.5.3 Ant colony optimization – extreme points results

The parameters used for testing these instances were the same in the tuning section. Due the size of the instances, instead of using different degrees of stability, as in section 6.5.2, and due to the results obtained on that section, it was decided to always use forty percent of the minimal dimension as the margin for keeping that stability.

For the pallet loading it was possible to obtain better results than the real case. Nevertheless the time cost was quite big as shown in Table 6.27.
<table>
<thead>
<tr>
<th>Instance</th>
<th>Type</th>
<th>Number of Pallets</th>
<th>Iterations</th>
<th>Stabilization Iteration</th>
<th>Time [s]</th>
<th>Time per ant [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDASAUSA</td>
<td>Non rolls</td>
<td>19</td>
<td>80</td>
<td>30</td>
<td>29200</td>
<td>36.5</td>
</tr>
<tr>
<td>INDASAUSA</td>
<td>Rolls</td>
<td>5</td>
<td>45</td>
<td>16</td>
<td>81282</td>
<td>180.6</td>
</tr>
<tr>
<td>INDASAUSA2</td>
<td>Non rolls</td>
<td>23</td>
<td>80</td>
<td>50</td>
<td>45920</td>
<td>57.4</td>
</tr>
</tbody>
</table>

Table 6.27 - Ant Colony Optimization - Extreme Points result

Considering the simplicity of the container loading, the Ant Colony Optimization algorithm seems to solve it only due to its stochastic nature, independently of the pheromone trail. Doing so it is possible to obtain the best solution, in the worst case, in three seconds. In order to better test this algorithm for this situation the problem should be more complex.

6.5.4 Comparing the three algorithms

Comparing the results, Figure 6.9, for the pallet loading it is possible to notice that the better alternative is the presented Extreme Points heuristic, emerging the Ant Colony Optimization applied to the Extreme Points as the worst choice, but still providing better results than the real methodology used at INDASA.
The time spent by each method is important, Table 6.28. Even if the Extreme Points Heuristic presents a time that is one order of magnitude above the Layer Building algorithm, it is a reasonable time. The Ant Colony Optimization presents a computational time that is two orders of magnitude above the Extreme Points Heuristic. Though, the time per ant, time to do a packing using Extreme Points, is least than the time used by the Extreme Points Heuristic. So the time consumed by the Ant Colony Optimization is only a matter of iterations and ants used since the method in which the Extreme Points are used reduces the packing time when compared to the Extreme Points Heuristic.

For the container loading, both the Extreme Points heuristic and the Ant Colony Optimization algorithm present good results. The first one presents the solution by rotating the containers, bin, and with the better time. The second one offers the solution by randomizing the pallets order, due to its stochastic nature, with a worse time, but even so small when compared to the pallet loading times. The Ant Colony Optimization algorithm offers an advantage; if there are exceeding pallets, the cost function can be adapted, so that only the best ones are chosen.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Type</th>
<th>Layer Building Algorithm</th>
<th>First Fit Decreasing</th>
<th>Ant Colony Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDASAUSA</td>
<td>Non rolls</td>
<td>24,7</td>
<td>121,8</td>
<td>29200</td>
</tr>
<tr>
<td>INDASAUSA</td>
<td>Rolls</td>
<td>26,5</td>
<td>386,5</td>
<td>81282</td>
</tr>
<tr>
<td>INDASAUSA2</td>
<td>Non rolls</td>
<td>51,0</td>
<td>197,1</td>
<td>45920</td>
</tr>
</tbody>
</table>

Table 6.28 - Time comparison
7 Conclusions

In this chapter the conclusions obtained, by examining the results, are exposed. In order to solve the packing process at INDASA, it was necessary to implement algorithms that could manage large instances and the specific constraints. In this work three different algorithms were proposed and implemented, mainly regarding the pallet loading: a Layer Building Algorithm, an Extreme Points First Fit Decreasing algorithm and the Ant Colony Optimization Extreme Points algorithm.

7.1 Benchmark instances

Regarding the Ant Colony Optimization algorithm applied to Extreme Points it was developed in a way that could solve both the Bin Packing Problem and the Container Loading Problem. Therefore the algorithm was tested for benchmark instances for both the problems.

The Bin Packing Problem instances were important not only for obtaining comparative results but also to tune the Ant Colony Optimization model. In this way, using different instances, it was possible to progressively tune the referred algorithm and use the tuning in all the other experiments.

In the comparative results, for the first class with fifty boxes, concerning Bin Packing Problems, it was possible to see that the algorithm could achieve better results than the state of the art; nevertheless for the second class it proved to be not as good. For the Container Loading Problem the algorithm proved to be at the same level as the state of the art. The limitations of this algorithm in the Container Loading Problem are due to the method which is used to pack boxes of the same type, trying to pack all those boxes at the same time. Mixing boxes of different types in the packing order would most certainly improve the presented results.

However this algorithm seems to be more time consuming than other algorithms that were presented for both the Bin Packing Problem and the Container Loading Problem. The most significant case is the presented Extreme Points Heuristic (C-EPBFD) used in the comparison of results. So a heuristic offers similar results and so can be in the same level as this meta-heuristic, thus a well chosen order of packing boxes can be the key for the best result.
7.2 INDASA results for the pallet loading

Considering the INDASA instances for the three developed algorithms, all the three proposed algorithms proved to be able to do better packing, in the pallet loading case, than the methodologies currently in use at INDASA.

7.2.1 Layer building algorithm

The Layer Building algorithm was the most basic and that took less time. Nevertheless, and even if the results matched the best obtained, this algorithm presents some issues with some of the constraints. This has to do with the agglomeration constraint, since it was applied by removing boxes that did not obey this constraint. This would make some of the packing not to satisfy the stability constraint. However, by removing the agglomeration constraint, it was possible to have considerably better results in terms of number of used pallets.

7.2.2 Adapted extreme points

In the case of the Extreme Points it was possible to apply all the constraints and additionally obtain a spatial representation of the boxes inside of the bin. Using this as a method to successfully introduce boxes inside a given bin, it was possible to apply it to two different algorithms. The first case was a simple First Fit Decreasing heuristic, the second was using a metaheuristic, Ant Colony Optimization.

7.2.3 Extreme points first fit decreasing

For the Extreme Points First Fit Decreasing heuristic, good results were obtained in a reasonable amount of time. Considering two cases, of absolute stability and of partial stability, both presented better results than the real results. It was also possible to notice that allowing a part of the product not being totally supported led to a decrease in the number of necessary bins and therefore to a better result. In this way it was possible to notice that simply by ordering the products by weight and applying a First Fit Decreasing algorithm, it was possible to have good results in all the instances.
7.2.4 Ant colony optimization applied to the extreme points

As for the case of the Ant Colony Optimization, it was possible to obtain similar results to the previous algorithms, and having all the constraints applied as derives from the Extreme Points algorithm. Nevertheless the computational time is much higher when compared to the methods used before. This leads to the conclusion, already referred, that having a good sorting of the boxes may prove enough to have a good packing. This conclusion is further backed up by the Bin Packing results.

The Ant Colony Optimization seems a good way of implementing further constraints and different cost functions. As the optimization presented here was dimensional, maximizing the occupied space, the heuristics can easily do it. However if a more complex optimization is presented, such as maximizing a certain type of products or the profit obtained, the heuristics may not be enough and the Ant Colony Optimization may prove to be a good and flexible alternative, by only having to adapt the cost function.

7.3 INDASA results for the container loading

For the INDASA instances, in terms of the container loading, it was possible to simulate the current organization. The Layer Building algorithm proved insufficient at this point; however the two other algorithms successfully achieved the required packing.

The Extreme Points heuristic applied to this case proved once again to be faster than the Ant Colony Optimization. However, even the Ant Colony Optimization, proved to have negligible times when compared to the pallet loading cases. The First Fit Decreasing had to have the container rotated in order to successfully pack all the pallets, so two packing were made. The Ant Colony Optimization proved to successfully do the packing without any modification.

7.4 Model for the expedition INDASA problem

The main motivation to all this work was to construct a model that could solve the INDASA problem. The steps that were taken until this moment are only a way of finding the correct tools. So considering the Bin Packing Problem for the pallet loading, the methodology followed should clearly the Extreme Points First Fit Decreasing, supported both by the obtained results and the time spent in obtaining them. As for the Knapsack Problem, regarding the container loading, there are two good alternatives that consume few resources. However the flexibility of the Ant Colony Optimization applied to the Extreme Points may be a key decision point. If there is a different goal, other than simply obtaining the given packing, like maximizing the profit,
requiring weight stability on the container, or even combining multiple criteria, this algorithm may prove to be the better solution.

### 7.5 Future work

The future work should be to improve the tool in order to allow its deployment at INDASA. Here is the summary of the steps to take into that direction.

First, the Ant Colony Optimization algorithm may be further improved, especially in order to solve the Container Loading Problem where, at first glance can prove to achieve better results. Due to the great number of involved parameters the tuning was not as complete as it could be. An exhaustive study of the parameters may be of great help. Also a good development may be the study of a normalization of the involved data. Also, due to the time spent in processing the INDASA instances parallel processing should be implemented in order to reduce it.

Secondly, regarding the adapted Extreme Points a further study of the involved parameters may also be done. It is believed that the ordering of the Extreme Points when applying both the First Fit Decreasing heuristic and the Ant Colony Optimization may not reach a consensus, ordering by x,y,z may prove better than z,x,y in one case, but worse in another, however it is worth studying. In this work it is proposed adding the products by rows, however adding it by layers may prove to reduce the time involved and even improve the results.

Thirdly, the order of the boxes was done considering only the products weight; it would be interesting to study other methods of ordering such as using clusters of weight and then ordering the products inside each cluster by volume or area. In the case of the Ant Colony Optimization algorithm, applied to the benchmark problems, the weight associated to each box is replaced by the volume, but it would be important to test other attributes such as the bottom area or height. Also the agglomeration constraint was made resorting to a static division of the product types into blocks, a good alternative would be doing this division dynamically in order to shape those blocks into the available space in a given bin.

Finally the developed Extreme Points approach opens many doors when it comes to implementing new methods. So an interesting path would be to implement the Layer Building algorithm defined in this work, using the Extreme Points in order to compare it to the already developed algorithm. Considering that the only heuristic applied was a First Fit Decreasing it would be interesting to apply other heuristics such as Best Fit Decreasing and evaluate if the computational time does not becomes too high.
8 Bibliography


APPENDIX A Tool developed to solve the INDASA problem - User Manual

The tool presented was developed in Matlab. The tool is composed by a set of windows that are described in the following appendix. Also it is presented the way to handle this tool in order to obtain successful results.

In order to have a better perception of the functionalities involved the appendix is divided into sections, each one explaining the functionalities of each window. The last section describes the procedures necessary to correctly run the program.

A.1 Menu_Inicial

The main menu is the starting point of the tool. As it is possible see in Figure A.1, when the tool is initiated most options are unavailable. Only by following the steps further described do that options become available.

‘Menu_Inicial’ window has two menus associated:

1. ‘Ficheiro’ (Files);
   1) ‘Carregar dados’ (Load data) – Opens progressively three windows asking for the files corresponding to the dimensional data, weight data and containers and pallets dimensions;
   2) ‘Sincronizar dados’ (Synchronize data) – Is part of the future work in order to synchronize the weight and dimensional data;
   3) ‘Carregar encomendas’ (Load orders) – Asks for the file containing the orders;
   4) ‘Exportar lista de expedição’ (Export expedition list) – Export the expedition list obtained after the simulation is done;
   5) ‘Sair’ (Exit) – Exits the tool;

2. ‘Ver’ (View);
   1) ‘Lista encomendas’ (Orders list) – Opens the window referred in section A.2;
2) ‘Lista de expedição’ (Expedition list) – Opens the window referred in section A.3;

Additionally the window is divided in three different sections concerning the container and pallets dimension, and finally a button, ‘Simular’ (simulate), that generates an expedition list. The three different sections are:

1. ‘Tipo de transporte’ (Sort of transportation) – Allows choosing three different transports and shows the dimension associated;
   a. ‘ISO 40’ – Maritime container ISO 40;
   b. ‘ISO 20’ – Maritime container ISO 20;
   c. ‘Camião’ – Truck, here there are predefined dimensions, however having this option chosen the button ‘Nova dimensão’ (New dimension) becomes available allowing changing the truck dimension;

2. ‘Pallet de rolos específica’ (Specific rolls pallet) – If the option is checked the rolls are packed in pallets with specific dimension. By having this option checked it allows to change the pallet dimension, figure;
3. ‘Lista de Paletes’ (Pallets list) – Presents a list of pallets associated to the selected transport and the client associated to the loaded order list. There are two associated buttons to add new pallets, ‘Adicionar palete’, and remove a pallet, ‘Eliminar palete’.

A.2 Lista_Encomendas

The objective of this window, Figure A.2, is to have a perception and control of the orders present for a given client. It is also in this window that it is possible to manually change the quantities of products involved. To do so the window is divided in two sections: ‘Lista de encomendas’ (order list), that present the enumeration of all the order lines involved and ‘Linha de encomenda’ (order line), which present the information concerning a specific order line.

Figure A.2 - Lista_Encomendas

‘Lista de encomendas’ can be divided in two popup menus, a list and a button, their function is:

1. First popup menu – allows seeing the list of products to be packed, and a list of products with incomplete data and so impossible to stow;

2. Second popup menu – ‘Ordenar por’ (sort by), it allows to sort the list bellow by different values such as: product volume, line order total volume or product weight;

3. List – It presents the product list depending on the option chosen in point 1;
4. Button – ‘Repor Valores de Origem’ (reset to original values), if the quantities of products have been changed, it reset those values to the original ones;

5. Also there is the display ‘Volume por Arrumar’ (unpacked volume), this only differs from zero if a packing has already been simulated and there were products unpacked. If a product is removed, the volume associated is subtracted from this value. This allows having an estimative of the products necessary to remove to get the entire list packed.

‘Linha de encomenda’ has a display, ‘Quantidade’ (quantity), which allows changing the number of products of a given order line, this is done by then clicking the button ‘Alterar Quantidade’ (change quantity). The other button, ‘Eliminar Linha’ (erase line), allows erasing a complete order line. The remaining displays show information concerning the selected order line.

**A.3 Lista_Palete**

This window, Figure A.3, is made available after a simulation is complete and shows the containers, the pallets in each container and the products packed in each pallet. This is only a display window and does not allow changing the packing.

![Figure A.3 - Lista_Palete](image-url)
Hence this window is divided in three different sections:

1. ‘Contentor’ (container) – It has a popup menu with the list of containers for the present packing. Additionally the displays show the percentage of occupied volume and the respective weight.

2. ‘Palete’ (pallet) – It has a popup menu with the list of pallets for the selected container, point 1. Additionally the displays show the percentage of occupied volume and the respective weight.

3. ‘Produtos’ (products) – It presents a list of all the products packed in the pallet selected in point 2. It also presents the data concerning the product selected in the list. The data concern the quantity, product description, dimension and order line.

A.4 Running the tool

In order to use efficiently use the tool it is necessary to follow a given set of steps here presented:

1. Go to ‘Menu_Inicial’ then the ‘Ficheiro’ menu and select ‘Carregar dados’ and then select the three required files. Having this done ‘Carregar encomendas’ becomes available;

2. Go to ‘Menu_Inicial’ then the ‘Ficheiro’ menu and select ‘Carregar encomendas’ and then select the file containing the orders. With this the option ‘Lista de encomendas’ in the ‘Ver’ menu becomes available. Also the sections in ‘Menu_Inicial’ become accessible;

3. Select the transport, ‘Tipo de transporte’;

   a. If the transport is ‘Camião’ it is possible to change the dimension, Figure A.4;

Figure A.4 – Dimensoes
4. Select if there is a specific pallet for rolls;
   a. If there is a specific pallet select the dimension by pressing ‘Alterar palete’, which calls the window shown in Figure A.5;

5. Select the dimension of the pallet to store the other products;
   a. If the list is empty or the pretended pallet is not there, press ‘Adicionar palete’, which calls the window shown in Figure A.5. For this case it is possible to associate the chosen pallet to the transport and client;

6. Press ‘Simular’ in order to start the simulation;
   a. If it is not possible to do the packing in only one container a message appears asking if it is desirable to use a new container, Figure A.7

7. Having the simulation complete the ‘Lista de expedição’ in the ‘Ver’ menu becomes available. Also if the packing is not complete the option ‘Volme por arrumar’ in the window ‘Lista_Encomendas’ is updated.

Having this done it is possible to start a new simulation by simply pressing ‘Simular’ again. The window shown in Figure A.5 also offers the possibility of introducing a new dimension, Figure A.6.