Packing Problems in Industrial Environments:

Application to the Expedition Problem at INDASA

José F. F. Beirão
IDMEC/IST, Universidade Técnica de Lisboa (TU Lisbon)
Av. Rovisco Pais, 1049-001 Lisboa, Portugal
Email: joseffbeirao@gmail.com

Abstract – This work proposes a generic algorithm to ameliorate the organization of expedition containers at INDASA. The implementation of the proposed algorithm required to simulate the packing of orders to be delivered. The problem was divided into a Bin Packing Problem, the pallet loading with the products, and a Knapsack Problem, the container loading with the already loaded pallets. In order to solve these two problems, three different algorithms are proposed in this paper: 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. Two real world INDASA instances were tested, and it was possible to reduce the number of necessary pallets from 26 to 23 three in the first instance, and from 24 to 20 in the second one. Moreover, benchmark instance for Bin Packing Problems and for Knapsack Problems were tested. The proposed algorithm was able to achieve better or similar results than state of the art algorithms.

I. INTRODUCTION

In any factory/company logistics is an important factor, thus 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.

Packing problems main objective is to determine a feasible arrangement of a subset of boxes, which maximizes a cost function; most times it is the volume occupation. 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 previous defined tolerance);
- Each box is not overlapping another box;
- Each box is placed parallel to the side walls of the container, which comes from the orthogonal packing.

Packing Problems can be divided into many different problems [1], due to the nature of the developed work only the three dimensional case of the Bin Packing Problem (BPP) and the Container Loading Problem (CLP) are approached. These problems ask for an orthogonal packing of a given set of rectangular shaped boxes into three dimensional rectangular bins. Each box \( j \ (j=1, \ldots, n) \) is characterized by a width \( w_j \), depth \( d_j \) and height \( h_j \). The three dimensional bins have width \( W \), depth \( D \) and height \( H \). The used coordinates origin is 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.

II. CLASSICAL METHODOLOGIES

In the literature, CLP can be solved using two approaches [1]. First, problems in which the complete load has to be stowed are differentiated from those which tolerate some goods to be left behind. Problems of the first type are known in the literature as Bin Packing Problems BPP, and problems of the second type as Knapsack Problems (KP). While BPP aims, e.g., to minimize the required container costs, the target of KP is usually to maximize the stowed volume. Here the CLP is approached as a KP, which is the most used approach in the literature. The BPP will be treated as a specific case due to its relevance in the literature.

Additionally, packing problems can be divided according to the type of boxes. Two boxes are considered being of the same type if, for a suitable space orientation, all their dimensions are equal. In this way there are problems with homogeneous set of boxes (only one type of boxes), weakly heterogeneous set of boxes (few types off boxes with many of the same type) and strongly heterogeneous set of boxes, (many boxes with few of the same type).

Most of the algorithms used to solve Packing Problems, whether we are talking about BPP or CLP, tackle the problem by dividing it in the three different sub-problems:

1) The place representation of the boxes inside the bin (PRB) – it consists of defining the location of the boxes inside the bin. In the one-dimension problem this is not an issue, but with the increase of dimensions the difficulty increases. In the three-dimensional the way this representation is defined has great influence on the final result;

2) The constructive heuristics (CH) – after having the box positions, it is necessary for a given a set of boxes, define how to place them in the bin or check the feasibility of a given solution. Often constructive heuristics are based on the first sub-problem;
3) The searching heuristic (SH) – the goal of this heuristic is to find the best box distribution among 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. The searching heuristic usually uses a constructive heuristic to verify the feasibility of the packing or to construct an initial solution.

Using these three sub-problems it is possible to define two paths to solve the problem as shown in Fig. 1. As a first way (path 1), CH are used to find an initial solution, and SH to refine the solution, until it reaches a satisfactory result. A second approach (Path 2) is to use directly SH to construct a solution and use CH to verify if it is a feasible or not. It is important to notice that CH is usually associated with PRB.

### III. LITERATURE OVERVIEW

There are three different problems that must be defined at this point, BPP, CLP and PRB. The last one is important as it may be used in different situations, as CH or SH may be associated to a specific case of BPP or CLP.

####  A. Place representation of the boxes inside the bin (PBR)

Concerning the importance of the PRB to have a good solution in the remaining points two different cases are highlighted. Firstly, the Extreme Points (EP) defined by Crainic et al. [2], which is a refined case of the Corner Points defined by Martello et al.[3] in which all the points available for packing are considered. This is done using the projection of the last packed box in the surrounding environment, Fig. 2.

![Extreme Points representation](image)

The second case regards a graph theoretical approach defined by Fekete and Schepers [4]. The authors considered the relative position of the items in a feasible packing and defined a graph describing the items overlapping according to the projection of the items in each orthogonal axis, Fig. 3.

#### B. Bin packing problem (BPP)

The BPP, the most referred problem in the literature, the possible alternatives are many due to the combination of CH and SH. The first exact solution concerning the three dimensional problem was reported by Martello et al. [3][5], expanding the work done for the two dimensional problem [6]. They developed a two level Branch & Bound (MVP) to find this solution; the first search tree assigns the items to the bins. For each node of the first-level search tree, they use a second Branch & Bound to verify whether the items assigned to each bin can actually be packed into it. Given a partial packing, there is a limited number of points within the residual space of the container where it is possible to accommodate a new item without the new packing being dominated by another, the Corner Points.

A Guided Local Search (GLS) is proposed by Faroe et al. [7]. This has proven 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 is reached. To speed the local search a Fast Local Search algorithm was used. Since the neighborhood is quite large the FLS speeds up the search for a local minimum.

Tabu search has also been a tool used by more than one author, Lodi et al. [8] did one approach by using a one level tabu search, nevertheless Crainic et al. [9] used a two level tabu search (TS²-Pack). 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 and Schepers [4] reducing the search space.

By simply using a heuristic based on First Fit Decreasing and Best Fit Decreasing in order to use Extreme Points Crainic et al. [2] developed a good and simple heuristic. First they test a set of different rules to order the boxes. Secondly, in order to use Best Fit Decreasing, they test different variables to quantify the best fit.
A different approach following multi-objective optimization is done. This was done using swarm optimization in a two dimensional problem by Liu et al. [10]. In this case not only the space utilization was optimized but also the desired position for the center of gravity. An evolutionary approach was used with a mutation operator as a source of diversity. The accommodation of the boxes inside the bins was done by a bottom left fill heuristic.

C. Container loading problem

Concerning the Container Loading Problem (CLP) a common approach is based on the Wall Building Algorithm used by George and Robinson [11]. Another alternative is to use a tree-search algorithm to find the set of layer depths and strip widths which results in the best overall filling, proposed by Eggblad and Pisinger [12]. 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 the possibility of backtracking in order to reach depth and width/height combinations that fit well together. Each strip filling problem is optimally solved using an efficient knapsack algorithm, and pairings boxes such that they form layers of uniform depth.

Another alternative is presented by Moura and Oliveira [13]. These authors present a greedy randomized adaptive search procedure, where the rejected spaces are amalgamated in order to improve the space use. In order to improve the amalgamation the layer width is assumed to be flexible in order to optimize the free space. After a first solution is found, it is improved by means of a local-search algorithm. This local search is based both on a greedy and a random algorithm. A greedy algorithm would lead to the choice of the best box type, and a completely random strategy would draw a solution from the entire list.

Bordfeldt et al. present a hybrid genetic algorithm [14] that is used to generate stowage plans with a wall building structure. A constructive heuristic is used to create the initial solution, and the first walls are the starting individuals for the evolution. For the generation of subsequent generations, problem-specific operators, are used. These include transferring some unchanged walls from the parents to the problem-specific operators, which cooperate through the exchange of best solutions.

In the application of Simulated Annealing and Tabu Search by Mack et al. [16], simulated annealing clearly wins over tabu search, especially for problems with larger solution spaces. However, the fine-tuning of the steering parameters of SA is a 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.

Wei et al. [17] propose a dynamic space partition and the usage of a tertiary-tree-based heuristic. 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 maximize the size of most likely usable spaces. An optimal fitting rule is also proposed 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.

IV. PROBLEM DESCRIPTION

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.

To solve it a tool was developed to simulate the packing of the products into the pallets, pallet loading, and then the pallets into the container, container loading. This aids both the picking process, by only getting the necessary products from the warehouse, and in the packing problem, by knowing which products go into what pallets. The tool includes also packing algorithms to optimize the system. These algorithms were developed on top of the packing problems models previously described with very the specific constraints that usually are not tackled in the literature:

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.

2) Rotation constraint – Most products only allow rotation around the vertical axis. This avoids placing the products in positions that can damage them. In this way, the products can only take two positions, the original position or rotated 90° around the vertical axis due to the orthogonality of the packing.

3) Weight constraint – Products with more weight must be on the bottom. This prevents the bottom products from being damaged by excessive weight. A weight margin which prevents a heavier upper product from damaging the other products must be defined.

4) Different products constraint – The order list contains two different types of products, rolls and non rolls that must be stowed in different pallets.
5) Product agglomeration constraint – Products with the same SKU\(^1\), 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.

Considering these specific constraints, it is now possible to define the packing problems. There are two different problems involved, based on the relation between the physical entities and the packing problem entities in TABLE I. The pallet loading is the real issue here – there are up to 3400 boxes divided in 165 types to be placed in more than 20 bins. This problem is tackled as a BPP which is the real issue concerning large instances. The simpler of these problems is the container loading – in this case there are less than 30 boxes, usually divided in 2 types, rolls and non rolls, to usually pack in only one bin. This problem is approached as a pallet loading problem, simplifying the CLP to a two dimensional problem in which only pallets with equal or smaller dimensions are piled on top of each other, treating this problem basically a general KP. By treating the container loading as a KP and filling one container at a time, the first containers to be filled should be the best containers and so if one container has to be erased, it should be the worst, the last one to be filled.

<table>
<thead>
<tr>
<th>Packing Problem VERSUS REAL Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packing</td>
</tr>
<tr>
<td>Bin</td>
</tr>
<tr>
<td>Box</td>
</tr>
</tbody>
</table>

If the number of necessary containers exceeds a desired number, or if one container is not as full as desired the user has two options; let the tool choose the products that are to be eliminated, in order to have the desired number of containers, or manually erase less desirable products, and run again the simulation. When the desired number of containers is reached, the tool then creates the expedition list according to the products packed on those containers. The overall process is described in Fig. 4.

V. PROPOSED ALGORITHMS

In order to solve the two packing problems, concerning the container loading and the pallet loading, two different approaches were followed. The first one based on the Layer Building Algorithm (LBA), and the second based on the Extreme Points (EP) [2]. The second approach resulted in two different algorithms, the first that applies a simple First Fit Decreasing (EPFFD) heuristic, and the second that uses an Ant Colony Optimization (ACO-EP). To notice that all algorithms aim at solving the more complex pallet loading and thus admitting that the algorithms could be adapted to the simpler case of the container loading.

Since there is a weight constraint asking for heavier products to be at the bottom, and both the LBA and the EPFFD ask for a predefined ordering of the boxes, for the pallet loading case the boxes were ordered by weight. If the heavier boxes are stored first that prevents them from crushing the lighter ones. Also for all the algorithms 90° rotation around the vertical axis is allowed so the orientation is chosen considering the available space and the box volume. So for a given square area the orientation to be chosen is the one that allows more boxes to fit it [18].

A. Layer building algorithm

The LB is based on the wall building algorithm [11], but instead of using vertical layers, parallel to one of the bin wall, it uses horizontal layers. The LBA does not use a PRB which resulted in a handicap. The non existence of a PRB was solved by dividing each bin into a set of layers where the height depends on the first box to be stowed in that layer. Also each layer is divided into rows that are adapted to the dimension of the box that generates the given row. This algorithm also creates a matrix relating the products height. So if a layer is not completely filled, boxes whose height is sub multiple to the original box height are chosen before the others to fill the layer.

Another important issue regarding the LBA is the implementation of the agglomeration constraint, for the pallet loading. As the original packing does not take into account this constraint, this is applied recursively. Hence if one product does not obey the constraint, it is removed from the packing and repacked in a different pallet. Since this may lead to some pallets being underused, the pallets whose percentage of occupied volume is under a value, determined by the percentage of occupied volume of all the bins, are

---

\(^1\) Stock Keeping Unit, a code that defines each product
repacked in order to improve it. However this method of removing boxes from a full pallet may prove to undermine the stability constraint.

B. Adapted extreme points

The EP case is an adaptation of existing work [2]. In this case some modifications were required, due to the fact that large instances are involved and due to the specific constraints. In order to reduce the number of extreme points available, instead of packing single boxes, whenever possible the boxes are packed in rows.

In order to have square areas, necessary to define the box orientation and the rows of products to be stowed, two groups of boxes are defined: the floor boxes and the obstacle boxes. By combining both is possible to obtain the available area for packing, as shown in Fig. 5. The floor boxes are the ones whose the top is at the same level as the EP where the present box is to be packed, and the top of those boxes defines the floor area. The obstacle boxes are the ones that are at the same level as the box to be stored. Considering the stability constraint, the path that was followed was to allow, a value equal to a percentage of the minimal horizontal dimension of the box, not being supported in each horizontal direction. So for zero percent the box is fully supported, until fifty percent the constraint is met and in order to disregard the constraint the percentage should be infinite.

The agglomeration constraint is applied by dividing each type of boxes in smaller blocks that satisfy the constraint, whenever it is not possible to stow all the boxes in the same bin.

C. Extreme points algorithms

At this point it is possible to define the two associated algorithms. The EPFFD is based on a well known First Fit Decreasing heuristic. This done by simply introducing the block of boxes into the first bin in which it fits, if there is none available a new one is created, Fig. 6.

The ACO-EP is a more complex problem even if many of the constraints were already implemented in the EP. Differently from the other used algorithms, the ACO-EP, has to find the order in which the boxes must enter the bins. The nodes in this problem represent the blocks of boxes and the paths the order in which they are stored into the boxes.

Since in the packing it is important to know the first box to be stowed, a zero node was created, which works as the ants nest. This node is always the starting point to the packing of a new bin. So if a block of boxes does not fit in a bin, a new one is created and the ant returns to zero node. The weight constraint is also inserted directly into the definition of the nodes. In this case the heavier box does not have to necessarily be at the bottom, a weight slack which prevents heavier boxes from damaging the lower ones is considered. So the path connecting node A to node B is only possible if some modifications were required, due to the fact that the constraint the percentage should be infinite.

The constraint: The percentage of occupied volume in the bin and the number of volume in a given bin, from being chosen.

So having the zero node as a starting point, a node is chosen and the block of products packed. Whenever a node is chosen the block of boxes is packed. If that is not feasible the bin is closed and the algorithm returns to node zero. In the case of the BPP bins are created until all the boxes are stowed: for the CLP bins are created until it reaches the chosen number of bins, hence finishing the packing. For the CLP the algorithm may finish before the number of iterations is reached if all the boxes are packed in the required number of bins, as it is represented in Fig. 9 at the end of the current paper.
VI. RESULTS

The results presented here concern three different types of instances: the BPP instances [3], from only four cases where chosen, the Loh and Nee instances [19] for the CLP, and the INDASA instances. For the first two problems only the ACO-EP was tested, since one of its strengths is the possibility of solving both the BPP and the CLP. For the third case all the developed algorithms were tested.

A. Bin Packing Problem

In order to test the ACO-EP algorithm it only four cases from the instances defined by Martello et al.[3] were used, Class I and Class VI were chosen in order to have different problems, variety of possible dimensions for the second case. For both, instances with fifty and two hundred boxes were generated.

From Fig. 7 it is possible to observe that the ACO-EP only outperforms the state of the art in one Class I with fifty boxes. However the remaining instances the results are not as good. The LB is always the best performing method since it represents a theoretical lower bound proposed by Boschetti and Mingozzi [20]. However these results are restrained by the fact that the instances used were generated solely for this problem and are therefore different from the ones used in the other problems. A bigger problem is the computational time, where the instances with two hundred boxes need 100 iterations with 10 ants each, which introduce a computational time between 950 and 1000 seconds. The instances with fifteen boxes needed only 40 iterations with the same number of ants and took less than 70 seconds.

B. Container Loading Problem

In this case each type of boxes was considered to be a node. By doing so the space of search was reduced and the problem became simpler. This fact allowed testing all the instances proposed by Loh and Nee[19].

**TABLE II** shows that most of the instances revealed to be quite easy to solve, only one instance was needed. Only four instances needed all the iterations, and from those only for LN7 instance was possible to obtain the optimal solution. These results can be explained by the algorithm itself, since it tries to pack all the boxes of one type instead of box by box. That leads to under packing and the solution probably would be better if the different types of boxes were mixed.

C. INDASA Problem

The INDASA instances resume to two different problems, the container loading and the pallet loading, and even for each of these two problems there are two different instances. The container loading proved to be simpler as expected. In this case the objective was to get the same packing as the one provided by INDASA. The LBA proved to be insufficient to solve it if the order in which the pallets were loaded was not the optimal. However both the EPFFD and the ACO-EP proved to do so efficiently, the ACO-EP by finding a suitable loading order and the EPFFD by rotating the container in order to equal that solution. Both cases presented good time results, when compared to the pallet loading, less than one second for the EPFFD and around three seconds for the ACO-EP. Nevertheless the ACO-EP may prove to be an optimal candidate for this problem since it can use different cost functions and by doing so introduces different and more interesting optimization problems, such as multi criteria problems.

The pallet loading proved to be more complex and also more time consuming. All the algorithms proved to be better than the real solution provided by INDASA. The LBA took between 25 and 60 seconds to solve the problem and was the fastest. It also allowed a comparison between the problem with the agglomeration constraint and without it. Thus without the constraint it were necessary six pallets less considering the sum of all the instances.
The EPFFD proved to find the best solution. For this method it the case with totally supported boxes and the case without that support were tested. In order to have fully supported boxes three additional bins were necessary. The computational times even if not so good as the LBA was between 120 and 400 seconds.

Finally the ACO-EP found better results than the real case, but it was outperformed by the other two algorithms and the time consumed was much larger, between 29000 and 81000 seconds. However the time for packing was less than the EPFFD, which shows that the high time consumed is dependant from the number of ants and iterations. A nicer tuning of the parameters evolved would probably decrease the time needed.

VII. CONCLUSIONS

This work proposed three different models to solve packing problems: Layer Building algorithm, an Extreme Points First Fit Decreasing algorithm and an Ant Colony Optimization applied to Extreme Points, which are adapted to the constraints imposed by INDASA, including large instances.

These models outperform the INDASA real results in acceptable time for the pallet loading. As for the container loading, only the EPFFD and the ACO-EP were able to solve it. The EPFFD proved to be the best alternative to the pallet loading. As for the container loading both the EPFFD and the ACO-EP proved to be viable alternatives; however the ACO-EP is a more flexible algorithm and may prove to have better advantages if it is necessary to work with several objectives.

Regarding the Benchmark instances, the ACO-EP is able to solve the BPP and the CLP. The obtained results are not at the same level than the state of the art. However the results are still good, since the algorithm was not developed to solve these instances.

As future work, an exhaustive study of the ACO-EP parameters may be of great help. Also, due to the time spent in processing the INDASA instances, parallel processing can be implemented in order to reduce it. Finally, LBA can be implemented using EP and compared with the current LB algorithm.

A. Bibliography

[17] Lijun Wei, Defu Zhang, and Qingshan Shen, ”A


![Diagram](image-url)