A heuristic algorithm for the Air Transport Unit Consolidation Problem

Lorenzo Peirano(1)  Enrico Angelelli(1)  Claudia Archetti(2)

(1) University of Brescia, Department of Economics and Management, Brescia, Italy
lorenzo.peirano@unibs.it, enrico.angelelli@unibs.it

(2) ESSEC Business School in Paris, Department of Information Systems, Decision Sciences and Statistics, Cergy, France
archetti@essec.edu

Abstract

Consolidation of loose packages into transport units is a fundamental activity offered by logistics service providers. Moving the transport units instead of loose packages is faster (with one movement only, multiple packages are loaded instead of having one load operation for each package), safer (chances of damage and loss is reduced) and cheaper. One of the typical objective of consolidation problems is the minimization of the number of transport units used, e.g., containers. In air transportation, however, transport units have multiple aspects which concur in the calculation of the cost and thus optimization in the number and characteristics of the transport units is required. In this paper, we present the air transport unit consolidation problem where the aim is to determine how to consolidate loose packages in transport units with the goal of minimizing the corresponding cost. The problem is a variant of the three-dimensional bin packing problem where the objective function is formulated according to the way costs are calculated in the air transport business. In addition, side constraints are included to take into account specific requirements. We propose a heuristic algorithm which construct an initial feasible solution and then improves it through a local search algorithm. Computational tests on randomly generated instances show that the algorithm provides high quality solutions in a reasonable computing time. In addition, tests on real data show that it improves solutions found in practice by the company providing the data.

Keywords: air transport; three-dimensional bin packing; consolidation; heuristics.

1 Introduction

Freight forwarders handle international shipments for their customers and focus their activity on offering the most complete logistic service from origin to destination. This involves not only the management of documents and the physical transfer of goods, but includes other logistic operations (packaging and labelling are just some examples). One of the activities carried out is the consolidation of loose packages into transport units (TU). For air shipments TUs are usually pallets of various dimensions, and crates. Since most of the times shipper companies are not able to consolidate themselves the goods of which the shipment is composed, freight forwarders spend a high amount of time in performing this operation. The most evident reason behind consolidation in TU is that the movement of a TU is not only faster, more efficient and cheaper than handling a higher number of packages, it also increases the safety of the shipment since it is more difficult to loose or damage packages. Moreover, since the information included in the documentation must be aligned to the characteristics of the shipment, controls by any monitoring organization (customs, FDA, police, etc.) are easier, and, thus, faster. However, choosing the number and quality of TU to be used is not a simple task and it requires the consideration of different aspects. Indeed, the way in which loose packages are consolidated impacts in the shipment cost. In fact, TUs can have different dimensions and weight capacities, which generates different costs. In the following, we assume the measures are expressed in centimeters and in the \( width \times length \times height \) format, with the latter sometimes not reported. The most common TU used is pallets. EUR and EUR1 pallets are the standard European 120x80x15 pallets, EUR2 are 120x100 while EUR6 are 80x60. Height is the same for every pallet and from now on it will not be reported anymore. American standards are slightly different (40x48 inches) while in Japan the standard measure is 110x110.
Air transportation services apply a unitary freight on the taxable weight, which is the higher between the real weight of the shipment and the weight equivalent conversion of its volume. Every cubic meter is equal to approximately 167 kilograms of weight. When the TU is stackable, the volumetric weight is converted on the actual volume of the TU. On the other side, when the TU is not stackable, it effectively occupy the volume generated by the projection of the base dimensions to the maximum height of the vehicle.

Each airplane model has different sizes of the cargo hull and thus its limitations and characteristics must be considered. Air transportation is carried out by two main families of aircraft: Passengers and Freighter. Passenger aircrafts (PAX) transport passengers on the upper deck and cargo on the lower section of the aircraft. Higher priority is given to passenger’s luggages and the remaining slots are allocated to cargo shipments. Depending on the model of the aircraft, different kind of Load Unit (LU) can fit inside the lower deck, defining the constraints in loading the cargo into the plane. The most common internal heights for LU for passenger airplane are 130 and 160 cm for medium and long range flights, respectively, while short range flight usually requires smaller airplanes with maximum height on the lower deck of 110 cm. Very small aircrafts do not use LUs or any kind of consolidated shipment and only accept loose packages with very limited weight and dimensions.

LUs shape and size defines the maximum dimensions allowed for each TU to be accepted on a given airplane. The TU is limited in every dimension by the physical boundaries of the LU. However, when using standardized TU like pallets, their base is always loadable within any LU and the only dimension which is limited is the height.

Freighter (Cargo Aircraft Only, or CAO) aircraft do not transport passengers and thus can make use of the upper section deck to transport cargo. The upper section deck exploits the circular shape of the aircraft to allow the transportation of bigger packages. Most of the times the upper deck do not use LUs and directly load the consolidated TU. One strict condition imposed by air companies is that the material to be loaded on the upper deck must be forkliftable and movable with handling vehicles.

![Figure 1: Cross-section of a CAO](image)

In Figure 1, a cross-section of a CAO aircraft is shown. The lower part is the lower deck and it is loaded with LU only. Upper deck is loaded with both LU and loose packages, especially if their dimensions are out of the gauge for the LU. The upper deck height limit depend on the base dimension of the package: with a small base it can be placed near the center where the shape of the aircraft allows for a higher height, otherwise, the wider the base, the lower is
the maximum height allowed. For aircrafts with a side door the height of the door is usually considered as maximum height allowed for transport. CAO aircrafts have a maximum loading height for cargo of 290 cm for medium sized aircraft and 310-320 for large sized aircraft. Even bigger CAO aircraft exists but do not operate with a line service and rely purely on chartering and thus are not considered in this work.

Figure 2: Boxes, transport units and load units

In this paper we study the problem of consolidating loose packages (boxes) in TU to be transported through air transportation services. As shown in Figure 2, we want to consolidate boxes into pallets, pallets are themself then loaded into LUs by the air company (or by a third party logistics company appointed by the air company). We focus on the first part of the problem, i.e., consolidating loose packages in TUs and we refer to Paquay et al. (2018) for a reference on the problem of loading already palletized items or loose packages into LU suitable for air transportation. The problem tackled in Paquay et al. (2018) focuses on loading loose packages and palletized packages into LUs. This operation is carried out by air companies (or, on their behalf, by airport handling operators) and happens after the problem we want to tackle in this work. Here, we focus on the first step where a freight forwarder (or a shipper) need to consolidate multiple packages into TU of multiple and varied dimensions in order to minimize the costs that will be paid for the transportation by air means. This is a separate and distinct operation carried out by shippers / freight forwarders on a previous time before delivering the shipment to the airport. While the procedure can be considered as a whole singular operation when shippers/freight forwarder have the availability of LUs, this is not a very common scenario. Most of the time they deliver loose packages or packages consolidated in TUs, and therefore the first operation illustrated in Figure 2 need to be tackled separately.

The Air Transport Unit Consolidation Problem (ATUCP) is thus the problem where, given a set of loose packages, the best packing into TUs has to be determined in order to minimize the total cost. Based on the definition given in Bortfeldt and Wäscher (2013), the ATUCP can be considered as a Multiple Bin-Size Bin Packing Problem (MBS-BPP) since the boxes to be packed are extremely heterogeneous, while pallets are weakly heterogeneous. We want to consolidate a set of boxes into the optimal heterogeneous mix of TUs of any given type that seeks the minimization of an objective function considering the value of the TUs (i.e. their taxable weight) and the position of the center of gravity. As explained later, the latter term measures the stability of the TU: we want a TU as stable as possible to avoid damages during transportation. The taxable weight is the unit of measure to which the unitary freight is applied by air companies. Clearly, it cannot be lower than the sum of the weights of the boxes. When weight is low, though, the volumetric weight conversion is usually higher and thus a proper consolidating technique is needed to effectively reduce the cost of the shipment.

Boxes can be stackable or non stackable. In the following we treat the case where TU are always non-stackable, i.e., no TU can be loaded on top of another TU. Whenever a TU is non stackable, the volume is calculated applying the maximum height of the airplane deck. Usually this maximum is considered using a standard height of 160 cm for lower deck cargo and 290 cm for upper deck cargo. Moreover, boxes can or cannot be turnable, i.e. two dimensions can be swapped. It is assumed that base dimensions are always swappable, the length can become the width and vice versa. This is not always the case if other means of transportation are considered (for instance, if we consider the consolidation of goods to be shipped through containerization, a package can have extraordinary dimensions that
allow it to be forkliftable on one side only, and thus it can be loaded only with the dimensions derived by the given rotation).

One aspect to be taken into account when planning the load layout of a TU is the position of the center of gravity of the load. The center of gravity is the unique point where the weighted relative position of the distributed mass sums up to zero. This is the point to which a force may be applied to cause a linear acceleration without an angular acceleration. Assuming that the weight is always equally distributed, every box have its center of gravity equal in geometrical center, and the center of gravity of the TU is the weighted average point of the boxes’ centers of gravity. A well balanced TU with a center of gravity placed near the central point of the base is easier to transport when lifted with forklifts or cranes, is more stable due to a better balance and, especially when pallets are considered, is more resistant and less subject to damages to the base. Usually it is better to keep heavy items at the bottom of the TU in order to avoid damages to other boxes and improve the balance of the cargo when subject to horizontal movement. Therefore, one of the objectives we seek is to place the center of gravity as near as possible to the center of the base and in the lowest position possible. Thus, the objective of the ATUCP combines the minimization of total costs and the optimization of the center of gravity. A numerical example of cost calculation is shown in Section 1.1 while the calculation of the center of gravity is explained in Section 2.2.

The contribution of this paper can be summarized as follows:

1. We introduced the Air Transport Unit Consolidation Problem (ATUCP) which is a problem finding relevant practical applications in airline freight transportation.

2. We consider an objective function including solution cost, measured on the basis of the TUs used to consolidate packages, and stability, measured by the center of gravity which depends on the disposition of packages in each TU.

3. We present a heuristic algorithm which first construct a feasible solution and then improves it through a local search based on different neighbourhoods.

4. We propose a procedure for generating ATUCP instances which enables a comparison between the solution provided by the heuristic algorithm and an optimal solution determined analytically.

5. We perform exhaustive tests on randomly generated instances and on real data. The results show that the heuristic algorithm provides high-quality solutions and largely outperforms solutions generated by the operators of the company providing the real data.

The paper in organized as follows. Section 1.1 presents a numerical example of how solution cost is calculated. The relevant literature is reviewed in Section 1.2 while the problem is formally defined in Section 2. Section 3 is devoted to the description of the algorithm while computational results are presented in Section 5. Finally, some conclusions are drawn in Section 6.

1.1 Example of cost calculation

Consider the following simple example: we need to transport 12 boxes with dimensions 60x40x60 and we have availability of pallets with base 120x80. Assume that the weight of boxes is very limited and therefore only the volume is considered. We can choose to consolidate the boxes in one single pallet 120x80x195 (solution 1) or two pallets where one has dimensions 120x80x135 and the other has dimensions 120x80x75 (solution 2). The following table report the volumetric weight for the two possible consolidations.

| Solution   | Volume Conversion |
|------------|-------------------|
| Solution 1 | 465 kg            |
| Solution 2 | 513 kg            |
Consolidating in one pallet generates less volume \((120 \cdot 80 \cdot 290 = 2.784 \text{ CBM})\), but forces to fly with a freighter aircraft since its height is greater than the maximum height of the LU \((160 \text{ cm})\). Pallets with lower height on the other side generates more taxable weight \((120 \cdot 80 \cdot 160 \cdot 2 = 3.072 \text{ CBM} \cdot 167 = 513 \text{ kg})\). Cargo flights are typically less frequent and more expensive. For instance, if a passenger airline (PAX) company offers a freight of \(\text{€ 2.00/kg}\) and a cargo airline (CAO) company offers a freight of \(\text{€ 2.30/kg}\), the total freight paid is as per table 2 below.

| Solution 1 | € 1,069.50 |
| Solution 2 | € 1,026.00 |

In this case, creating two pallets is the cheapest option, since it reduces the total cost of the air shipment. Furthermore, if we consider the availability of pallets with base 120x120 we can consolidate all the boxes on a single pallet 120x120x135 with a volume weight conversion equal to 384 kg. Considering the freight of PAX flight \(\text{€ 2.00/kg}\) the cost is \(\text{€ 768.00}\). Consolidating the boxes in a single pallet with base 120x120 is then the best option. In our work it is assumed that there always is a wide choice of flights and aircraft, thus the need for optimization. While during the COVID-19 crisis the number of passenger’s flight reduced significantly (even though many of them were used to transport cargo nonetheless), the situation returned to pre-pandemic levels, where multiple companies offer their services for the transportation of cargo, using both passenger’s and Cargo Aircraft Only options. The effective availability of space on aircrafts impacts the time needed to transport goods and the cost of it, but it is not depending on the consolidation layout and is not directly handled by shippers/forwarders and thus, it is not considered in this study.

1.2 Related literature

The ATUCP is a three dimensional bin packing problem (3D-BPP) where the objective is finding the optimal way to allocate loose packages to TUs in order to seek the minimization of TU’s cost function and optimize the center of gravity. \cite{Martello2000} give a description of the 3D-BPP and propose an exact branch and bound algorithm for the solution of large instances. A MILP formulation with different valid inequalities is presented by \cite{Hifi2010b}. 3D-BPPs are often very difficult to solve with exact methods. Henceforth, a wide range of heuristics approaches has been introduced in the literature. Tabu search techniques are presented in \cite{Lodi2002} and \cite{Crainic2009} with good results. \cite{Gendreau2006} propose a tabu search algorithm capable of solving a combined capacitated vehicle routing and loading problem. \cite{Crainic2009} propose a two level tabu-search heuristic. The first level heuristic is meant to deal with the optimality of the bin packing problem, while the second level of the heuristic finds feasible layouts for the packing of the items in the bins. Neighborhoods are explored using a k-chain-moves procedure involving chains of \(k\) changes in the solution, which proves to be effective in exploring large neighborhoods in contained computational times. This is clear in the computational results where the method proposed result particularly effective in finding good solution when the time available is limited. Similarly, the three-dimensional single bin-size bin packing problem is studied in \cite{Hifi2010a} where the authors present a MILP formulation of the problem. Moreover, they propose valid inequalities usefull to improve the relaxed lower bound of the formulation proposed. Final solutions are found through an heuristic approach. \cite{Mack2012} present a straightforward heuristic capable of solving two-dimensional and three-dimensional bin packing problems. In the same work the authors introduce a set of 1,800 benchmark instances with on average 1,500 items each, which are useful in order to test bin pacing algorithms on very large instances. \cite{deAlmeida2010} present an heuristic for a particular case of the container loading problem where special constraints on the position of the boxes are implemented. The heuristic approach proposed by the author is designed to be easy to implement for warehouse workers during the loading operation and the results of the computational campaign shows that the methodology is capable of generating good solutions, especially when real-world data is used as input. \cite{Alvarez2013} present a two phase algorithm useful for
solving two and three-dimensional multiple size bin packing problems. The procedure proposed first seek a starting solution using a GRASP heuristic including a constructive phase, a postprocessing phase and improvement moves. The best solution found by the GRASP heuristic are then combined together using a Path Relinking procedure, proposed in three different versions. They compare the computational results with literature benchmark instances. In their work [Trivella and Pisinger (2016)] present an application of the container loading problem with the additional practical extension of considering the load balance. This latter objective consist in finding the average center of mass in the most suitable position. First, the authors introduce a MILP formulation for the problem. Given the difficulties encountered to solve even small instances, a multi-level local search heuristic approach is presented. Alvarez-Valdés et al. (2015) present a study with the aim of improving the lower bounds for integer programming formulations for some relaxations of the original problem. Logical considerations, such as avoiding symmetrical solutions, fixing boxes into bins and setting bound for each type of bins, are implemented in order to improve the bounds for the problem. An exact MILP formulation designed to accomodate packages by layers is designed by Taylor et al. (2017). The authors show that when the data is pre-processed in order to arrange the order of the boxes to be packed by non-increasing height or volume, calculating very tight lower bounds is an operation of little complexity, which allow the formulation proposed to be solved in reasonable times even for large instances. In general, the objective of the pallet loading problem is the minimization of the number of pallets used to pack the loose packages available, while ensuring their stability. In the ATUCP the object is slightly different and minimizing the number of pallets it’s not always the best option. The minimization of the volume (and thus the cost generate for shipping by air means) is part of the objective as well, considering th multiple dimensions of TUs available, it is possible that having a greater number of smaller TUs proves to be a better solution. Elhedhli et al. (2017) study the mixed-case palletization problem, which is an extension of the three-dimensional bin packing problem which requires the boxes laying upon other boxed to be properly supported. In order to solve the problem the authors propose a MILP-based slicing heuristic which aims at maximizing the fill rate of the lower levels in order to achieve a higher density, allowing upper layers to be placed upon them. The iterative process pack subsets of similar boxes separately, ideally starting from bigger boxes and then filling the gaps remaining by positioning smaller boxes. The methodology was tested against well-known instances in the literature. Further studies on the mixed-case palletization problem are considered in Elhedhli et al. (2019) where the authors propose a novel formulation and a column generation approach suitable to solve it. The sub-problem to optimize is a two-dimensional layer generation problem making use of the concept of superitems, where a superitem is a collection of individual items that are compactly stacked together with no gaps between them. The superitems are used to ensure the compactness of the layers. Computational experiments are carried on both benchmark instances derived from the literature and from instances built using industrial data. Genetic algorithms proved to be very successful in solving the 3D-BPP: examples can be found in Wu et al. (2010), Kang et al. 2012, Gonçalves and Resende 2013, Li et al. 2014, Dornas et al. 2017 and Ha et al. 2017. Approximation algorithms are proposed in Miyazawa and Wakabayashi 2009. One interesting application is proposed in Wu et al. (2017). The authors divide their problem in three different layer. The first one requires to pack irregularly shaped item into boxes in order to maximize the utilization rate of the boxes. Afterwards, the boxes are packed in to crates, where the dimensions of the crates could vary significantly and needs to be optimized, leading to a three-dimensional variable-bin packing problem. Finally, the crates need to be loaded into containers. The latter problem is a more classic single container loading problem. The solution is found through the use of a three-stage heuristics. For the second problem, the heuristic at first pack the boxes into a minimum number of crates of larger dimensions, than try to improve the solution by checking if crates of smaller dimensions are a better option. The effectiveness of the heuristic is tested on real-world instances inspired by an industrial case. Alonso et al. 2019 propose mathematical formulations for problems arising in practical applications in logistics. An industry application evolved from a layer-based column generation approach is presented by Gzara et al. 2020, where real-life constraints widely applied by industries are considered and an extensive experimental campaign is provided. None of the papers mentioned above tackle the ATUCP. Pallet loading problems and container loading problems are similar in their objective of finding stable, compact, balanced loading that maximize the volume occupation of the container (container loading) or minimize the number of containers used (pallet loading problem). Both the problems, in their
classical representation, uses one kind of container/pallet of fixed dimensions. In the ATUPC, we consider multiple dimension for the TU that can be used, a highly heterogeneous set of boxes, and the objective we want to optimize takes into account multiple aspects that are not considered together in previous works. The main contribution of this work is the introduction of such a unique problem. Thus, we now provide a formal description of the problem.

2 Problem Description

In this section we introduce the notation and the formal definition of the problem.

2.1 Problem setting and notation

2.1.1 Transportation units

Several types of TU may be available for consolidation, we denote as $TUS$ the set of different types of TU at hand. Each TU type $tut \in TUS$ is associated with the following parameters:

- $X_{tut}$: the TU width;
- $Y_{tut}$: the TU length;
- $Z_{tut}$: the TU maximum height allowed;
- $Q_{tut}$: the maximum weight that can be carried by the TU;
- $V_{tut} = X_{tut} \cdot Y_{tut} \cdot Z_{tut}$: the maximum box volume that can be carried by the TU;

As procurement of TU items in general is not an issue, we assume to have an unlimited availability for each type of TU $tut \in TUS$. In the following we will use $tu \in TU$ to refer to a generic TU item of any type and we will denote as $X_{tu}$, $Y_{tu}$, $Z_{tu}$, $Q_{tu}$ and $V_{tu}$ the respective measures inherited from its type $tut \in TUS$.

Position of consolidated boxes on each TU will be described according to a three-axes Cartesian coordinate system with origin in the south-west corner of the TU base. In other words, we consider the base as a geographical map, so we move along the $X$ axis from west to east, along the $Y$ axis from south to north, while axis $Z$ starts from the down position and ascends towards the up position. Thus, the base of a TU $tu$ cover the rectangle $[0, X_{tu}] \times [0, Y_{tu}]$ and its center is point $C_{tu} = \frac{1}{2}(X_{tu}, Y_{tu}, 0)$. The available space for box placement is the rectangular cuboid $[0, X_{tu}] \times [0, Y_{tu}] \times [0, Z_{tu}]$ with volume $V_{tu} = X_{tu} \cdot Y_{tu} \cdot Z_{tu}$. Figure 3 shows a representation of the coordinate system used.

![Coordinate system](image)

2.1.2 Boxes

We indicate as $B$ the set of boxes to be consolidated. Each box $b \in B$ has a weight $w_b$ and three dimensions (width, length and height) that can be potentially oriented in space in six different manners: we have to choose which dimension out of three is parallel to the $Z$ axis, then which of the other two is parallel to $X$ axis; the remaining dimension is
thus forced to be parallel to Y axis. Flipping dimension orientation (up-down, west-east or north-south) is irrelevant as this does not change how the box occupies space. At least two orientations are guaranteed for each box: height parallel to Z, and width parallel to either X or Y. Logical attributes TXZ<sub>b</sub> and TYZ<sub>b</sub> of box b say whether the box can be oriented with dimension width and length parallel to the Z axis, respectively. The values of these attributes depend on the characteristic of box content and are fixed for each box. For sake of simplicity, whatever orientation a box b is given, we denote with width<sub>b</sub>, length<sub>b</sub>, height<sub>b</sub> its extension along the X, Y and Z axes. An example of the six possible orientations is given in figure 4. Finally, the logical attribute st<sub>b</sub> denotes whether the box is stackable or not. If a box is non-stackable, no boxes can lay upon it.

![Figure 4: Box orientations](image)

2.1.3 Boxes loaded on a TU

When a box b is assigned to a TU tu we say, with a little abuse of notation, b ∈ tu.

The position of a box b ∈ tu is described in terms of the coordinate system of tu and we denote as (x<sub>b</sub>, y<sub>b</sub>, z<sub>b</sub>) the position of its west-south-down point in this coordinate system. As a consequence the box will lay in the 3D interval [x<sub>b</sub>, x<sub>b</sub> + width<sub>b</sub>] × [y<sub>b</sub>, y<sub>b</sub> + length<sub>b</sub>] × [z<sub>b</sub>, z<sub>b</sub> + height<sub>b</sub>], and assuming a homogeneous density for the box, its center of gravity will be in point CG<sub>b</sub> ≡ (x<sub>CG,b</sub>, y<sub>CG,b</sub>, z<sub>CG,b</sub>) = (x<sub>b</sub>, y<sub>b</sub>, z<sub>b</sub>) + \frac{1}{2}(width<sub>b</sub>, length<sub>b</sub>, height<sub>b</sub>).

The center of gravity CG<sub>tu</sub> of a loaded TU tu is the weighted position of the centers of gravity of the boxes b ∈ tu. In formula CG<sub>tu</sub> ≡ (X<sub>CG,tu</sub>, Y<sub>CG,tu</sub>, Z<sub>CG,tu</sub>) = \sum_{b \in tu} CG<sub>b</sub> · w<sub>b</sub> ∑_{b \in tu} w<sub>b</sub>;

We also introduce three normalized centering measure of CG<sub>tu</sub> with respect to the center of the tu base C<sub>tu</sub>, defined as

\[ M_{x,tu}^x = \frac{2X_{CG,tu} - X_{tu}}{X_{tu}} \]
\[ M_{y,tu}^y = \frac{2Y_{CG,tu} - Y_{tu}}{Y_{tu}} \]
\[ M_{z,tu}^z = \frac{Z_{CG,tu} - Z_{tu}}{Z_{tu}} \]

In particular, measure \( M_{xy,tu}^{xy} = M_{x,tu}^x + M_{y,tu}^y \) determines how much centered is the projection of CG<sub>tu</sub> on the base of the tu. Ideally, the perfect layout has \( M_{xy,tu}^{xy} = 0 \). Figure 5 shows level curves of \( M_{xy,tu}^{xy} \).
2.2 Problem formulation

The objective of the ATUCP is to find the feasible consolidation layout of all boxes in $B$ which minimize the value of objective function (1), which we call $FITNESS$:

$$FITNESS = \sum_{t_u \in TUL} (V_{t_u} + \alpha \cdot (M_{xy}^{t_u} \cdot M_{z}^{t_u} + \Theta)) + \beta \cdot |TUL|$$

(1)

where $TUL$ is the list of TUs actually used in the consolidation:

- $V_{t_u}$ is the maximum volume allowed to the TU $t_u$. As explained in Section 1.1, we use the volume as an approximation of the cost;

- $M_{xy}^{t_u}$ is the already mentioned measure of the distance between the projection of the center of gravity $CG_{t_u}$ of the load on the base of $t_u$ and its center $C_{t_u}$. Ideally, the perfect layout has $M_{xy}^{t_u} = 0$ with a balanced distribution of load around the center of the TU.

- $M_{z}^{t_u}$ is used to evaluate the vertical position of the center of gravity of the TU. Ideally, heavier boxes should be positioned on the bottom, which leads the load center of gravity to lower positions.

- $\Theta$ and $\alpha$ are tuning parameters aimed at balancing the impact of $CG_{t_u}$ and $V_{t_u}$ on the objective function. In particular, $\Theta$ is used as a scaling factor to balance the term related to the center of gravity evaluation with the other two terms in the objective function.

- The term $|TUL|$ is the number of TU used in the consolidation while parameter $\beta$ represents the fixed cost for each TU expressed in volume. Indeed, each used TU generates handling time cost as x-rays check, transshipment, and customs and security inspections.
Thus, the objective of the problem is to minimize the number of TUs used and the total volume generated while keeping the center of gravity of the TUs in a position that is as centered and low as possible. While the major objective is the minimization of the total volume, the other objective ensure the stability of the TUs created and the minimization of the costs derived by a higher number of TUs created. The parameters are tuned in order to give priority to the minimization of the costs (generated by volume and number of TUs), while optimizing the center of gravity of the loading layout. The feasible solution space is defined by the following set of constraints:

- **Box orientation:** according to attributes $TXZ_b$ and $TYZ_b$, orientations with width and/or length dimensions parallel to Z axis may not be allowed to some boxes. However when width or length dimensions are allowed to be parallel to Z axis, the height dimension can be parallel to X or Y axis;

- **Weight limit:** the sum of box weights assigned to each $tu \in TUL$ must be lower than the TU weight capacity $Q_{tu}$. That is, $\sum_{b \in tu} w_b \leq Q_{tu}$

- **TU boundaries:** all boxes assigned to each $tu \in TUL$ must respect the TU boundaries; that is, for all $b \in tu$ the following conditions must hold:

$$\begin{align*}
0 \leq x_b & \land x_b + \text{width}_b \leq X_{tu} \\
0 \leq y_b & \land y_b + \text{length}_b \leq Y_{tu} \\
0 \leq z_b & \land z_b + \text{height}_b \leq Z_{tu}
\end{align*}$$

In words, along each axis the edge of the box must be fully contained within the TU boundary. Namely, regarding X axis, the west coordinate must be positive ($0 \leq x_b$) and the east coordinate must be not larger than $X_{tu}$ ($x_b + \text{width}_b \leq X_{tu}$). Analogously, With respect to Y a Z axes.

- **Overlapping:** boxes must not overlap each other, i.e. one box cannot penetrate within another box; overlapping check is given by the statement: two boxes $a, b \in tu$ overlap each other if and only if the following system is satisfied

$$\begin{align*}
\max(x_a, x_b) & < \min(x_a + \text{width}_a, x_b + \text{width}_b) \\
\max(y_a, y_b) & < \min(y_a + \text{length}_a, y_b + \text{length}_b) \\
\max(z_a, z_b) & < \min(z_a + \text{height}_a, z_b + \text{height}_b)
\end{align*}$$

- **Box stackability:** The space above non-stackable boxes must be left empty, this means that no boxes can occupy the space of the projection along the Z axis of the base of the non-stackable box.

Note that none of these constraints considers vertical stability. The usual approach to ensure vertical stability is to fill the eventual empty spaces with pluriball/alluminium sheets. The proposed packing procedure focuses on trying to limit as much as possible the creation of gaps and empty spaces and henceforth no hard constraints are implemented to ensure vertical stability.

To the best of our knowledge, there is no version of MBSBPP considering this set of constraints in the literature.

### 3 An iterated local search algorithm for the ATUCP

We approach the enormous complexity of the problem with a heuristic algorithm based on the Iterated Local search (ILS) scheme which we call *ILS-ATUCP algorithm* and whose general logic is depicted in the next lines. More details follow in the next sections.

#### The construction algorithm

Each of the following phases strongly relay on a constructive algorithm called 3DBP that receives in input a type of TU $tut \in TUS$ and a set of unassigned boxes. The algorithm picks one new TU of the given type and feasibly fills it with unassigned boxes; this step is iterated until the set of unassigned boxes is empty. Details on algorithm 3DBP are provided in Section 3.1

#### Initialization phase
In the first step set $TUS$ is sorted according to increasing values of volume $V_{tut}$ and the constructive algorithm 3DBP is called on the first TU type in the list and the whole set of unassigned boxes.

**LS1 search**

Once the initial solution is constructed, a first order local search (LS1), whose moves are described in Section 3.2, is applied to possibly improve the packing by moving boxes between TUs or by destroying some TUs and assigning again the boxes by means of algorithm 3DBP in an attempt to reduce the number of utilized TUs and/or better assign part of boxes to TUs. The type of TU used by algorithm 3DBP in this stage is the last used to build a solution. Namely, when LS1 is executed immediately after the initialization phase, the smallest volume TU type will be used, when LS1 is executed after the LS2 search, the last TU type considered in LS2 will be used in LS1. The search proceeds until no improvement is found.

The aspect of considering different air services with different air freights is not considered, since the selection of the best air service to be used is a complex problem which consider a wide spectrum of technical aspects. However, given a set of TUs that can be loaded in a given air service, the algorithm, through the use of LS2, will search for the best layout possible allowing to pack the loose boxes in a mixed set of different types of TU. If multiple air services are available, the procedure can be repeated for each service by adjusting the available types of TU between those that can be accepted by the air service considered. In this way the best layout for each service will be given by the algorithm.

**LS2 search**

When LS1 fails to improve the incumbent solution, a second order local search (LS2) is started. A set of TUs, whose loads satisfy certain requirements, are removed from the incumbent solution and constructive algorithm 3DBP is repeatedly called to rebuild a solution, every time with a different type of TU. Every time an improving solution is found through LS2, the algorithm restarts with LS1. LS2 is applied until all possible TU types are checked in the same round, i.e. when all the possible TU types are tried with no improvements, the algorithm stops.

The reason why we emphasize the distinction between LS1 and LS2 is that while LS1 is devoted to little adjustments of the solution (exchange of boxes between TUs or replacing a number of TUs with other TUs of the same type), LS2 search is aimed at modifying the solution by diversifying TU types. As we show in the computational analysis, the LS2 search is of fundamental importance in solution improvements proving the role of diversification of TUs type in the load consolidation.

### 3.1 Algorithm 3DBP

The construction algorithm receives in input a type of TU $tut \in TUS$ and a list of unassigned boxes $\bar{B}$. The output produced by the algorithm is a feasible assignment of boxes in $\bar{B}$ to an appropriate number of TU of the given type.

We implemented an adaptation of the Extreme Point (EP) methodology introduced by [Crainic et al.](2008) for the 3DBP. EPs are the positions where we want to place the south-west-down corner of a box. For each TU $tu \in TUL$, the list $EPL_{tu}$ of available EPs is initialized with the south-west corner of its base where the origin of the coordinate system associated to the TU is located. Figure 6 show a representation of the spatial system used by the algorithm. The list is updated each time a new box is added to the TU.
Every EP $e$ is defined by its coordinates $(x_e, y_e, z_e)$ and three characteristics that also depend on the current load of the TU:

- $\bar{X}_e$: maximum dimension allowed along the $X$ axis for a box $b$ to be positioned in $e$ (i.e. $\text{width}_b \leq \bar{X}_e$). Any box with a larger width would either pass the TU boundary or compenetrate some other box.
- $\bar{Y}_e$: analogously to $\bar{X}_e$ defines the maximum dimension allowed along the $Y$ axis for a box to be positioned in $e$ (i.e. $\text{length}_b \leq \bar{Y}_e$).
- $\bar{Z}_e$: analogously to $\bar{X}_e$ defines the maximum dimension allowed along the $Z$ axis for a box to be positioned in $e$ (i.e. $\text{height}_b \leq \bar{Z}_e$).

Given the set $\bar{B}$ of boxes to be packed and one TU type $tut$, the algorithm first sorts the boxes using algorithm $\text{SORT}$, described in Section 3.1.1, to identify the order in which boxes are assigned to TUs. Once the list of boxes in $\bar{B}$ is properly sorted, the algorithm starts to pack them one at a time.

Each box is loaded in its best position on one of the TUs already in $\text{TUL}$ if possible. Otherwise a new TU $tu$ of type $tut$ is added to $\text{TUL}$, its EP list $\text{EPL}_{tu}$ is initialized with $e = (0, 0, 0)$, and the current box is placed at that EP.

Once the current box is placed, $\text{EPL}_{tu}$ is updated by using the algorithm $\text{EP-Generation}$ described in Section 3.1.3, the box is removed from $\bar{B}$ and the next box is considered until $\bar{B}$ is empty.

In order to decide the position of a box, for each TU $tu \in \text{TUL}$, for every EP $e \in \text{EPL}_{tu}$ and for every allowed box orientation, algorithm $\text{CanFit}$ (described in Section 3.1.2) checks whether the box can be feasibly placed in $e$ on $tu$.

In this case, the assignment is priced by the formula:

$$\text{COST}(b, e) = N \cdot z_e + x_e + y_e$$
$$+ M \cdot (z_e + \text{height}_b)$$
$$- N \cdot \theta \cdot [(\bar{X}_e - \text{width}_b) + (\bar{Y}_e - \text{length}_b)]$$
$$+ \lambda \cdot [(\bar{X}_e \mod \text{width}_b) + (\bar{Y}_e \mod \text{length}_b)] - N\text{BOX}$$

where $N$ and $M$ are fixed big constants and:

- $N \cdot z_e + x_e + y_e$ is the evaluation given to the position of the EP $e$. Lower first and western-southern EPs are preferred, as this leads to more compact layouts where boxes are more likely to be tight, without empty spaces between them. Even though western-southern points do not explicitly seek for load centering, this guarantee a better exploitation of space in horizontal dimensions.
- $M \cdot (z_e + \text{height}_b)$ is the evaluation given to the position of the west-south-up corner of the box. It represents the height of the box within the TU and favors, accordingly with the objective function, orientations that minimizes the height, to keep the center of gravity as low as possible.
Lastly, each of the box weight. First order cluster \( C \) way to obtain the minimum possible height. Then, the algorithm divides the set \( \bar{T} \) to summarize, the desired layout minimizes the height of the TUs in with the highest number of boxes already placed. It tries to find a perfect partitioning of the empty space left to obtain tighter layouts, and prefers to place boxes in TUs available after a box has been inserted, to better occupy the empty space of the TU. Moreover, the pricing formula helps improve the fitness of the assignment.

The algorithm places the current box in a TU and position and orientation which is feasible and with minimum value of function \( COST \).

The idea of the pricing formula is to prefer the west-south-down available EP, while taking into account both the height of the box (in order to favor orientations where the height of the package is lower) and the remaining space available after a box has been inserted, to better occupy the empty space of the TU. Moreover, the pricing formula tries to find a perfect partitioning of the empty space left to obtain tighter layouts, and prefers to place boxes in TUs with the highest number of boxes already placed.

Function \( \cdot \) components act together in order to find layouts that suit the objectives described in \( \cdot \). To summarize, the desired layout minimizes the height of the TUs in TUL, thus minimizing the value of \( CG_{tu} \). Having compact layouts, where the space utilization is optimized, helps to reduce the total volume \( V_{tu} \) and the number of TU used in the final solution. Moreover, compact layouts with no empty spaces within boxes tend naturally to produce centers of gravity toward the center of the TU. Overall, every aspect of the cost function \( \cdot \) is designed to synergize with the general minimization of objective function \( \cdot \).

When the ideal position of the box is identified, the algorithm inserts the box in the selected TU at the selected EP and:

- updates the list of available EPs using algorithm EP-Generation (EP occupied by the box is removed and new potential EP points are added to \( EPL_{tu} \)).
- updates \( \bar{X}_e, \bar{Y}_e \) and \( \bar{Z}_e \) of every EP of the TU.

### 3.1.1 Algorithm SORT

Algorithm SORT is focused on creating an ordered list of packages on the basis of weight, dimension and height, as described in the following. The algorithm is called by algorithm 3DBP an receives in input the type of TU \( tut \) algorithm 3DPB is working with, the set of unassigned boxes \( \bar{B} \) and two parameters \( n \) and \( m \) whose role is explained below.

Ideally, heavy boxes needs to be placed at the base of the TU, to optimize the center of gravity objective. At the same time, big boxes, especially those with a large base, should be allocated first when large space slots are still available while small boxes will be easily placed in a some of many small space slots formed during TU loading operations. Thus, algorithm SORT starts by rotating each box \( b \in \bar{B} \), compatibly with its rotation attributes \( TXZ_b \) and \( TYZ_b \) in such a way to obtain the minimum possible height. Then, the algorithm divides the set \( \bar{B} \) in \( n \) first order clusters based on box weight. First order cluster \( C(i) \) contains boxes with weight in the range \( \frac{\mu_w}{n} \cdot ((i - 1), i] \forall i = 1, \ldots, n. \) Afterwards, the algorithm divides each first order cluster in \( m \) second order clusters based on box base dimension. Second order cluster \( C(i, j) \) contains boxes from cluster \( C(i) \) with base area \( (x_b, y_b) \) in the range \( \frac{X_{min}Y_{max}}{m} \cdot ((j - 1), j] \forall j = 1, \ldots, m. \) Lastly, each of the \( n \cdot m \) clusters is internally sorted with respect to decreasing height of the boxes.
Finally, clusters $C(i,j)$ are taken in decreasing order ($i = n \ldots , 1$ and $j = m \ldots , 1$ for each $i$) to compose the final sorted list which is sorted by decreasing weight first, decreasing base second, and decreasing height last.

### 3.1.2 Algorithm CanFit

Algorithm CanFit receives as input a TU $tu$ with its current load, a candidate EP $e$ and an oriented box $b$. The algorithm must respond with a yes or no to the inquiry whether box $b$ can be placed on $tu$ in the given EP $e$ with the given orientation. First, algorithm checks whether $b$ dimensions are lower than the maximum dimensions allowed for a package in the EP chosen, that is $\text{width}_b \leq \bar{X}_e$, $\text{length}_b \leq \bar{Y}_e$, and $\text{height}_b \leq \bar{Z}_e$. If just one of the inequalities fails, the algorithm stops with a no answer. Otherwise, we do not have an answer yet and we need to check inequalities (2) where the role of box $a$ is played by each box already loaded on $tu$. If at least one box $a$ exists such that equation (2) is satisfied the algorithm stops with answer no, otherwise return answer yes. To better illustrate the situation we provide the following example. In a pallet $120 \times 80 \times 100$ we have, among others, one box $b_1$ with dimension $30 \times 30 \times 30$ in position $(0,0,0)$ and another box $b_2$ with dimension $40 \times 40 \times 40$ in position $(15,30,0)$. If we consider EP $(0,0,30)$, the one generated by the south west up corner of the first box, the maximum dimensions for boxes considered for that EP are 120 along the $X$ axis, 80 along the $Y$ axis and 70 along the $Z$ axis, as showed in Figure 7.

![Figure 7: Boxes placed in the pallet and maximum dimensions for EP (0,0,30)](image)

Next, we want to insert a box $b_3$ with dimension $60 \times 60 \times 20$. By applying the first fit check mentioned above, we should be able to insert it in $(0,0,30)$ since the dimensions of the box are lower than the maximum dimensions of the EP. Figure 8 shows the result of this operation.
Clearly, boxes $b_2$ and $b_3$ overlaps. Henceforth, we need a further check based on the position and dimensions of the boxes already placed in the TU in order to avoid overlaps.

3.1.3 Algorithm EP-Generation

Each time a box $b$ is allocated to a TU $tu$ the EP $e$ where the box is placed is removed from $EPL_{tu}$. Then up to five new EPs are added to $EPL_{tu}$. The new EPs are generated according to the following rules:

1. The east-south-down corner of box $b$ with coordinates $(x_e + \text{width}_b, y_e, z_e)$ is projected down, until it touches the base of the TU in the $XY$ plane or the up face of a box. The resulting point is then projected south, until it touches the plane $XZ$ or the north face of a box. The point resulting from the projection is added to $EPL_{tu}$. More precisely, the projection down either touches the plane $XY$ at point $(x_e + \text{width}_b, y_e, 0)$ or it stops on the face of the first package $b_1$ it encounter in point $(x_e + \text{width}_b, y_e, z_e_1 + z_{b_1})$ where $e_1$ is the EP where box $b_1$ is placed. The algorithm proceeds to project the newly found point to the $XZ$ plane until it touches the plane in either point $(x_e + \text{width}_b, 0, z_e_1 + z_{b_1})$, or until it touches the face of the first package $b_2$ along the projection in either point $(x_e + \text{width}_b, y_e + \text{length}_b, 0)$ or point $(x_e + \text{width}_b, y_e + \text{length}_b, z_e_1 + z_{b_1})$, depending on the previous projection. The point found, either on the face of the TU or on a face of another box, is added to $EPL_{tu}$.

2. The west-north-down corner of box $b$ with coordinates $(x_e, y_e + \text{length}_b, z_e)$ is projected first on the $XY$ plane and then on the $YZ$ plane like in case 1.

3. If box is stackable (attribute $st_b$ is true), west-south-up corner point with coordinates $(x_e, y_e, z_e + \text{height}_b)$ is added to $EPL_{tu}$.

4. If box is stackable (attribute $st_b$ is true), the west-south-up corner point with coordinates $(x_e, y_e, z_e + \text{height}_b)$ is projected to the $XZ$ plane until it either touch the plane in point $(x_e, 0, z_e + \text{height}_b)$ or it stops the projection on the face of the first package encountered $b_1$ in point $(x_e, y_e_1 + \text{length}_b, z_e + \text{height}_b)$. The point resulting from the projection is added to $EPL_{tu}$.

5. Similarly to 4), west-south-up corner point with coordinates $(x_e, y_e, z_e + \text{height}_b)$ is projected to the $YZ$ plane only if the box is stackable. In this case, the projected point is added to $EPL_{tu}$.

As a special case we observe that the first box to be assigned to a TU is always placed in $e = (0, 0, 0)$ and projections (3), (4) and (5) produce the same point $(0, 0, \text{height}_b)$, resulting in the generation of 3 points instead of 5.

In Figure 9 we show a graphical example of the depicted procedure, where a newly inserted box (circled in red) generates five EPs. For the sake of clarity, only the newly generated EPs are shown in the picture.
3.2 First order local search (LS1)

Local search LS1 implements three moves $N_1$, $N_2$ and $N_3$ applied sequentially. If a move succeeds in finding an improving solution, the search is started over from $N_1$ with the new incumbent solution. If no move is successful, the search proceeds to LS2. Implementation of each move consists in applying a determined sequence of steps (micro-moves) for a fixed number of times. Specifically, each step is repeated three times in the implementation whose results are presented in Section 5.

$N_1$: At each step, one box on top of one TU (origin) is taken and moved on top of another TU (destination). The box to be moved is randomly selected among the three with highest position in the origin TU and placed in the destination TU so that $COST$ function is minimized. Different strategies are used to select which TU is used as origin and which TU is used as destination, i.e. different strategies are employed to decide from which TU move boxes and which boxes move them to. The strategies are employed in the following order:

- the origin TU is the one with the greatest weight; the destination TU is the one with smallest weight;
- the origin TU is the one with the greatest weight; the destination TU is randomly chosen;
- the origin TU is the one with the greatest height; the destination TU is the one with smallest height;
- the origin TU is the one with the greatest height; the destination TU is randomly chosen;
- the origin and destination TUs are randomly chosen;

$N_2$: The iterated sequence of steps is composed as follows:

- two TUs are randomly selected;
- one box for each TU is randomly chosen among those that are laying on the top;
- if the two boxes can be feasibly swapped their new position is determined so that $COST$ function is minimized for each of the receiving TUs.

$N_3$: The iterated sequence of steps is composed as follows:

- draw a random integer number $n$ between 2 and $|TUL|$;
- destroyed $n$ TUs randomly chosen
– call algorithm 3DBP to recover a feasible solution; input for algorithm 3DBP is the set of boxes released by TUs destruction and the last type of TU either used in the initialization step or in the last call from LS2 as explained in Section 3.3.

### 3.3 Second order local search (LS2)

While Local search LS1 is focused on adjusting load of TUs already packed using multiple moves, LS2 is based on a single type of move. The idea is to variate the type of used TU by destroying a set of TUs in the incumbent solution and rebuild a feasible solution by repeated calls to algorithm 3DBP with different types of TU. Destroyed TUs are those which satisfy at least one of the following criteria:

- TUs with a filling rate lower than $\Omega$. The idea is that TUs with a low filling rate contain few boxes which might be easily inserted in existing TUs or in TUs of a different type.

- TUs where the lateral space left is greater than $\gamma$. If the minimum space left between the boxes and the boundaries of the TU along the X and Y axis is high, the solution can be potentially improved by selecting a different TU. TUs with a broader base can contain more boxes, potentially reducing the total number of TUs used. On the other side, TUs with a narrow base are used better and their cost is potentially lower.

After TUs destruction, algorithm 3DBP is repeatedly called with a different TU type taken from $TUS$. Namely, $TUS$ is sorted by increasing volume $V_{tut} = X_{tut} \times Y_{tut} \times Z_{tut}$, then is circularly scanned with a pointer starting over at the first place as it passes the end of the list. The pointer is initialized at the beginning of the list when Algorithm ILS-ATUCP is started and it determines the type of TU to use each time algorithm 3DBP is called (to build the first solution), during move $N3$ of LS1, and during LS2.

Local search LS2 stops when a call to 3DBP succeeds in finding an improving solution, in which case algorithm jumps back to LS1; in this case the pointer will determines the type of TU to pass to algorithm 3DBP during move $N3$, and the point to start from the next time LS2 will be executed. Otherwise, pointer completes a whole scan of $TUS$ and the algorithm definitely ends it run.

### 3.4 Numerical example

We present a numerical example to better explain how the 3DBP algorithm works. Figure 10(a) shows a partial solution produced by the 3DBP algorithm for a standard Europallet 120x80x130 where we highlight the currently available EPs. In the next iteration we need to insert a box of dimensions 30x40x20, weight is not considered in the example.

Figure 10: Boxes and EPs
Table 3 shows the available EPs and the maximum dimensions allowed for a box to be placed at the EP.

| EP | X \(e\) | Y \(e\) | Z \(e\) | MX \(e\) | MY \(e\) | MZ \(e\) |
|----|--------|--------|--------|--------|--------|--------|
| 1  | 30     | 0      | 20     | 90     | 80     | 110    |
| 2  | 60     | 0      | 20     | 60     | 80     | 110    |
| 3  | 0      | 60     | 0      | 80     | 20     | 130    |
| 4  | 0      | 0      | 20     | 120    | 80     | 110    |
| 5  | 90     | 0      | 20     | 30     | 80     | 110    |
| 6  | 0      | 40     | 20     | 120    | 40     | 110    |
| 7  | 80     | 0      | 20     | 40     | 80     | 110    |
| 8  | 80     | 40     | 20     | 40     | 40     | 110    |
| 9  | 110    | 40     | 0      | 10     | 40     | 130    |

It can be easily checked that the box can fit in every EP with the only exception of EP 9, since the box has no dimension lower than 10, the maximum width allowed by the EP. For every EP where the box can fit, we calculate the cost of the EP using (3) for every available rotation and select the best rotation. We then select the EP associated with the minimum cost. Table 4 shows the results of the calculation, represented visually by Figure 10(b). For simplicity, we only consider and report the cost of the best orientation for every EP.

| EP | X \(e\) | Y \(e\) | Z \(e\) | Cost   |
|----|--------|--------|--------|--------|
| 1  | 30     | 0      | 20     | 1630   |
| 2  | 60     | 0      | 20     | 1660   |
| 3  | 0      | 60     | 0      | 660    |
| 4  | 0      | 0      | 20     | 1600   |
| 5  | 90     | 0      | 20     | 1690   |
| 6  | 0      | 40     | 20     | 1640   |
| 7  | 80     | 0      | 20     | 1680   |
| 8  | 80     | 40     | 20     | 1720   |
| 9  | 110    | 40     | 0      | NO FIT |

Our goal is to place the box in the lowest, western southern EP available. Even by visual inspection it is easy to verify that the best EP is (0,60,0) and the cost function reflects that. We note that the high difference between EP3 and all remaining EPs is due to the fact that EP3 is the only one with \(z_e = 0\). Figure 11 shows the visual representation of the TU once we insert the box in the lowest-cost EP. In order to insert the box in EP (0,60,0) we must rotate it to have dimensions 40x20x30.
4 Computational results on 3D-BPP benchmark instances

In order to study the behaviour and the effectiveness of the ATUCP algorithm, the instances introduced by Bischoff and Ratcliff (1995) for the 3D-BPP are solved. Seven subsets of instances do not consider weight of goods, while 5 subset of instances do. Each subset contain 100 instances and it is defined by the quantity of different types of boxes considered. For the sake of easing the comparison between the results, in this section the same output reported in Bischoff and Ratcliff (1995) for the first 700 instances (the ones without considering weight) are compared with the results obtained by the 3DBP-Algorithm proposed in this paper. We now provide some details on the instances

There are seven subset of instances, each one containing 100 instances. In every instance boxes are regular hexahedrons. The seven subsets of instances are differentiated on the basis of the number of types of different boxes: 3, 5, 8, 10, 12, 15 and 20, respectively. For all instances a single type of bin is used with dimensions 587x233x220 (i.e. a 20 feet standard container).

The algorithm proposed in Bischoff and Ratcliff (1995) is focused on maximizing the volume utilization of the container and ensuring the stability of boxes. It is also optimized to handle single bins and weakly heterogeneous boxes, while the algorithm presented in this work tries to find optimal solution using weakly heterogeneous bins and strongly heterogeneous boxes.

Table 5 show the results obtained for each subset of 100 instances solved in terms of: solution time (ST), average utilization rate of fullest container (A%), highest utilization rate of the fullest container (Max%), minimum utilization rate of the fullest container (Min%).

| Ist | ST(s) | A%     | Max%   | Min%    |
|-----|-------|--------|--------|---------|
| 1   | 10.64 | 77.23% (81.76%) | 90.01% (94.36%) | 59.34% (64.96%) |
| 2   | 9.24  | 72.47% (81.70%) | 83.12% (93.76%) | 57.84% (66.90%) |
| 3   | 11.46 | 67.72% (82.98%) | 79.99% (92.76%) | 56.96% (66.91%) |
| 4   | 10.44 | 69.98% (82.60%) | 80.01% (88.89%) | 59.32% (66.46%) |
| 5   | 13.12 | 69.01% (82.76%) | 79.85% (90.39%) | 61.01% (70.38%) |
| 6   | 11.75 | 67.85% (81.50%) | 79.92% (89.15%) | 57.44% (64.86%) |
| 7   | 12.89 | 67.77% (80.51%) | 78.37% (88.18%) | 62.12% (70.50%) |
When compared to the results obtained in Bischoff and Ratcliff (1995) reported in round parenthesis, the ATCUP algorithm performs reasonably, even if it is less competitive. The average utilization rate is lower and both maximum and minimum utilization rates are lower. The main reason for this performance issues are to be searched in the logic behind the 3D-BPP proposed and the problem tackled. In Bischoff and Ratcliff (1995) the container loading problem is studied where the objective is to maximize the utilization rate of containers loaded, while in ATUCP the objective is different and considers multiple aspects of air transportation. Furthermore, the benchmark instances introduced by Bischoff and Ratcliff (1995) are generated in order to study the maximum number of boxes that the algorithm can load in a single container. That means that the final solution always contains at most two TUs, and therefore the only local search operators used in the ATUCP algorithm are the ones in \( N_1 \) and \( N_2 \). As we will see, the most effective local search operators are \( N_3 \) and \( LS_2 \). Thus, the improvements made by the use of local searches on the instances proposed in Bischoff and Ratcliff (1995) are negligible. Overall, though, the ATUCP algorithm finds compact and good solutions. Utilization rate is satisfying and boxes are stable and few empty spaces are left between packed boxes.

5 Experimental campaign

No benchmark instances exist for the ATUCP. Therefore, we generated them according to the method explained in the following. We carried out two sets of experiments.

The first set aims at evaluating the solutions provided by ILS-ATUCP on instances for which a lower bound on the solution value can be determined. First we identify the lower bound solution for 100 instances (more details in the Appendix in Section 7.1). Then, starting from the lower bound solutions found, three partitioning schemes are used to obtain the list of boxes to be inserted in the TUs. The three algorithms partition the TUs of the lower bound solutions in order to generate the list of boxes boxes. The first scheme follows a layer based procedure to generate boxes. The second scheme generates boxes by filling up the volume of the TU. The third one generates a number of boxes of fixed dimensions that perfectly fit the TUs. Finally, the solution found by ILS-ATUCP is compared to the lower bound solution previously identified, in terms of number of TUs used and volume occupation.

The second set of tests aims at comparing the solution of ILS-ATUCP algorithm with the ones provided by the internally developed solver used by the Italian freight forwarding company which inspired this work. The comparison is made on two sets of instances. The first set consists of instances generated as described above. The second set consists of real instances provided by the company.

Before presenting the results, we now report some observations related to the parameters included in the objective function (1). According to the instances characteristics (described in the following) we set the value of \( \Theta \) equal to 100: this provides solutions in which the value of the center of gravity term of the objective function is balanced with respect to the other two terms. The value of \( \alpha \) has been set to 1 and we performed a sensitivity analysis at at verifying the robustness of solutions with respect to this parameters. As mentioned in Section 5.2 this parameter has a negligible impact on the solution structure. Finally, the value of \( \beta \) has been set to 100. Also in this case, when changing the value of \( \beta \) the solutions structure did not change.

This section is organized as follows. Section 7.1 describes how we generated ATUCP instances. Section 5.2 is devoted to the evaluation of ILS-ATUCP performance by comparison with the optimal solutions and to the sensitivity analysis on algorithm’s parameters. Section 5.3 presents a comparison between algorithm’s solutions and company’s practice.

5.1 Partitioning schemes to generate boxes

Three partitioning schemes are proposed to partition the TUs identified by the optimal solution of the lower bound into boxes. They do not consider weight as the weight density is constant. Thus, they are just based on volumes.

- The first scheme partitions the volume of each TU according to a layer-based procedure. The TU is first divided in layers based on the height, then each layer is divided first along one direction and then on the other. The sketch of the procedure is given in Partitioning Scheme 1. \( Z_{left}, X_{left}, Y_{left} \) are the residual height, width and length of the TU while \( Z_{UB}, Y_{UB}, X_{UB} \) are the upper bound values of box height, length and width dimension.
respectively, $Z_{LB}, Y_{LB}, X_{LB}$ are the lower bound values of box height, length and width dimension respectively. Upper and Lower bound values are inserted in order to limit the randomness of the generation procedure.

**Algorithm 1:** Partitioning Scheme 1: Layer-based procedure

1. **Initialization** ($Z_{le:f} \leftarrow Z_{TU}$)
2. **while** $Z_{le:f} > 0$ do
   1. $X_{le:f} \leftarrow X_{TU}$
   2. if $Z_{le:f} < Z_{LB}$ then
      1. $Z_b \leftarrow \text{Random}(Z_{LB}; \text{MIN}(Z_{UB}; Z_{le:f}))$
   else
      1. $Z_b \leftarrow Z_{le:f}$
   end
3. **while** $X_{le:f} > 0$ do
   1. $Y_{le:f} \leftarrow Y_{TU}$
   2. if $X_{le:f} < X_{LB}$ then
      1. $X_b \leftarrow \text{Random}(X_{LB}; \text{MIN}(X_{UB}, X_{le:f}))$
   else
      1. $X_b \leftarrow Z_{le:f}$
   end
4. **while** $Y_{le:f} > 0$ do
   1. if $Y_{le:f} < Y_{LB}$ then
      1. $Y_b \leftarrow \text{Random}(Y_{LB}; \text{MIN}(Y_{UB}, Y_{le:f}))$
      2. Generate box with dimensions ($X_b, Y_b, Z_b$)
   else
      1. $Y_b \leftarrow Y_{le:f}$
      2. Generate box with dimensions ($X_b, Y_b, Z_b$)
   end
   1. update $Y_{le:f}$
5. update $X_{le:f}$
6. update $Z_{le:f}$

Figure 12 shows a graphical illustration of the partitioning scheme 1, where layer 1 is the layer identified along the Z axis, layer 2 the layer along the X axis and Layer 3 is the last one, along the Y axis.

- The second scheme starts by generating a box with random dimensions in the southern-eastern-down corner of the TU, the same location where EP (0,0,0) is positioned when the TU is empty. Once the box is generated, one
of the faces directed towards the inside of the TU is selected at random (direction 1, 2 or 3 in Figure 13). Then, boxes are generated with the dimension of the face selected fixed, and random dimension along the direction faced by the selected face of the box. The partitioning scheme partitions the "tunnel" generated by projecting the selected face in the direction of the projection. For instance, direction 2 projects the face in the YZ plane along the X axis, so it partitions the projected tunnel along the X axis. The boxes generate will have identical $Y_b$ and $Z_b$ dimensions and different $X_b$ dimension. Once one direction is fully partitioned, new EPs are generated and the procedure continues until the volume of the TU is fully partitioned. The procedure is described in Partitioning Scheme 2 where $f$ is the selected face.

**Algorithm 2: Partitioning Scheme 2**

Initialization: Add EP (0,0,0) to EP-List

while $|\text{EP - List}| > 0$ do

select one EP $e$ at random from EP-List;

Generate one box $b$ with random dimensions $(X_b, Y_b, Z_b)$;

$f \leftarrow \text{Random}(1,3)$;

if $f = 1$ then

while $Z_{\text{left}} > 0$ do

$Z_u \leftarrow \text{Random}(Z_{LB}, Z_{\text{left}})$;

Generate one box $u$ with dimensions $(X_b, Y_b, Z_u)$

Update $Z_{\text{left}}$

end

Remove $e$ from EP-List, add EPs in position 1 and 2 (if available) to EP-list according to Figure 13

end

if $f = 2$ then

while $X_{\text{left}} > 0$ do

$X_u \leftarrow \text{Random}(X_{LB}, X_{\text{left}})$;

Generate one box $u$ with dimensions $(X_u, Y_b, Z_b)$

Update $X_{\text{left}}$

end

Remove $e$ from EP-List, add EPs in position 1 and 3 (if available) to EP-list according to Figure 13

end

if $f = 3$ then

while $Y_{\text{left}} > 0$ do

$Y_u \leftarrow \text{Random}(Y_{LB}, Y_{\text{left}})$;

Generate one box $u$ with dimensions $(X_b, Y_u, Z_b)$

Update $Y_{\text{left}}$

end

Remove EP $e$ from EP-List, add EPs in position 2 and 3 (if available) to EP-list according to Figure 13

end

Figure 13 shows a visual representation of the idea behind the second partitioning scheme. Once a box is created, one direction is chosen and the "tunnel" created by projecting the face directed towards the chosen direction is partitioned into boxes until the edge of the TU is reached. From there, new EPs are added and used as starting location for the generation of new boxes and the procedure iterates until the volume of the TU is fully partitioned.
The third partitioning scheme identifies a perfect partition of each type of TU using boxes of standard dimensions and simply generates boxes based on the lower bound solution. For instance, a pallet 120x80x160 can be partitioned in 32 boxes 40x30x40, so for each TU with dimension 120x80x160 in the lower bound solution, 32 boxes 40x30x40 are generated. The dimensions of the boxes are chosen arbitrarily to perfectly partition the volume of the TU. The number and dimension of the boxes generated is the same for each kind of TU used in the solution, i.e. a TU 120x80x130 will generate a different number of boxes with different dimensions than those generated by a TU 120x100x160, but every TU 120x80x130 will generate the same boxes.

Table 13 in the Appendix reports some statistics about the instances generated by the three partitioning schemes. The first scheme generates the lowest number of boxes, but the variety of boxes dimensions is the largest, both in terms of number of unique dimensions and medium square deviation from the average for each dimension (the average square deviations for the three dimensions are (30.21, 27.65, 32.82) for partitioning scheme 1, (11.42, 9.54, 7.10) for partitioning scheme two and (0, 3.15, 1.07) for partitioning scheme three). The second scheme generates a high number of small boxes with a great variety of dimensions. The third scheme generates an high number of boxes, often higher than the second partitioning scheme, but the number of unique dimensions is limited and the dimensions are more suitable for simpler and clearer allocations of the boxes in the TUs.

## 5.2 Computational results on randomly generated instances

The 300 instances generated (100 instances for each partitioning scheme) are solved with ILS-ATUCP and the solutions are compared with the lower bounds. Moreover, different values of $\Omega$ in LS2 are used to solve the 300 instances and the solutions are studied in order to carry out a sensitivity analysis. We also performed experiments by varying the value of $\gamma$ in LS2 but noticed that this parameter has no impact on the solution produced by LS2. Thus, we fixed it to 100 in the following experiments. In Table 14 some aggregated statistics statistics for the 100 instances generated using each partition algorithm are reported.
Columns A1, A2 and A3 refers to the set of instances generated by the first, second and third partitioning scheme, respectively. The first row reports the average number of TUs used in the solutions found, the second row is the average difference of volume generated with respect to the lower bound solutions. For difficult instances the ILS-ATUPC algorithm is incapable of perfectly recreating the original TUs layout, the solution found generates a higher volume. The third and fourth row reports the average value of the center of gravity coordinates $CI^{xy}$ and $CI^{z}$ as determined in (1), respectively. The fifth row reports the average solution time in seconds. The sixth row reports the number of instances where the solution found by the ILS-ATUPC algorithm is equal to the lower bound solutions. The seventh and eighth row report the average maximum and minimum filling percentage of the solutions found, respectively.

It is interesting to note the evolution of the solution when varying the value of $\Omega$. For the first and second partitioning schemes, columns A1 and A2, the pattern is fairly clear. By allowing a wider search in LS2, the ILS-ATUPC algorithm is capable of reducing the number of TUs used. On the other side, the volume developed is slightly higher on average. This is because while noticeable improvements are made by the use of less, bigger TUs, in some instances the search deviates highly and find worse solutions, since some TUs with high fill rate (85+%) are destroyed and rebuilt with a different kind of TU which is not always optimal. Moreover, this result is influenced by the value.
of the parameter $\beta$. The higher is this value, the greatest are the efforts in reducing the number of TUs used, even if this leads to an increase in the volume used. The solutions of the instances generated using the third partitioning scheme present a particular behaviour. For every value of $\Omega$ with the exception of $\Omega = 90$ and $\Omega = 95$, the solution found is always the same. This is due to the fact that the solution found before LS2 is good enough to never trigger LS2, and so the solution space is not thoroughly explored. With $\Omega = 90$ and $\Omega = 95$, or by forcing the search through LS2, different dimensions of TUs are used and the solutions found show great improvements. The reduction in the number of TUs used and the difference in the gap with the solutions found by the exact method are measures of the improvements found through the use of LS2. Moreover, the center of gravity position found by the ILS-ATUPC algorithm for the instances generated by the third partitioning scheme is optimal in almost every TU built.

We now move to the analysis of the ILS-ATUCP algorithm performance on the basis of the partitioning scheme used to generate the instances. The solutions found in the instances generated by the first partitioning scheme are such that the TUs used are compact and the volume is well covered. This is due to the layer structure of the instance. While it is nearly impossible for the algorithm to perfectly recreate the original layers, it proves its effectiveness in creating TUs that are compact, with boxes packed together in order to create balanced, leveled layers with few empty spaces between boxes and towards the edge of the TU. The center of gravity is in satisfying position in almost every solution, both in terms of base and height position.

The second partitioning scheme, on the other side, produces a high number of boxes of almost unique dimensions each, and building good TUs proves to be a challenging task, especially on instances with high weight and volume. Considering the difficulty of the task, the solutions generated by the ILS-ATUPC algorithm are compact and with few empty spaces. Moreover, the position of the center of gravity is more centered and at a lower height if compared with the instances generated with the first partitioning scheme. The time required to find a solution, on average, is considerably higher and the impact of widening the research by increasing the value of $\Omega$ is evident. On the other side, the improvements in the solutions found, made possible by a further use of LS2, shows the effectiveness of the local search.

Solutions of the instances generated by the third partitioning scheme are a good example of the potential effectiveness in real life scenarios. Solutions are found in fast times and the TUs built are compact, balanced and with a high utilization rate of the TU volume. Figure 14 shows an example of two TUs built on an instance generated using the third partitioning scheme. One TU is perfectly reproduced by using boxes of equal dimensions that fits the TU shape, while the other contain boxes of various dimensions and are assembled together in a compact, stable and centered layout.

![Figure 14: Example of solutions found for an instance generated using Partitioning Scheme 3](image)

Figure 15 and 16 show a TU built from an instance generated with the first and second partitioning scheme, respectively. Even though the number of boxes is high and each of them has unique dimensions, the ILS-ATUPC algorithm finds a good solution that nicely occupy the volume of the TU.
Overall, the TUs built are compact, stable, found in relatively fast times and have a satisfying utilization rate.

The effectiveness of the local search procedures is reported in Table 7. The first row reports the number of instances where LS1 finds improvements on the original solution. The second and third rows report the average and max improvements found by LS1. The fourth row report the number of instances where LS2 finds improvements of the incumbent solution. The fifth and sixth rows report the average and maximum improvement found by LS2. The last two rows are the average and max reduction in the number of TUs obtained through both local search procedures. As expected, LS1 never reduces the number of TUs, while LS2 is fairly effective in that. LS1 finds little upgrades, mostly through N1 and N3, while N2 is difficult to apply due to the fact that is difficult to find boxes that can be swapped without breaking the overlapping constraint. LS2 on the other side respects the expectations and is fairly efficient in improving the solution.
Table 7: Local search procedures analysis

| Stat          | A1  | A2  | A3  |
|---------------|-----|-----|-----|
| NA° Improvements LS1 | 57  | 55  | 0   |
| Avg. Improvement LS1 % | 1.23 | 1.45 | 0   |
| MAX improvement LS1 % | 1.78 | 2.01 | 0   |
| NA° Improvements LS2 | 94  | 98  | 0   |
| Avg. Improvement LS2 % | 9.67 | 9.97 | 0   |
| MAX improvement LS2 % | 41.22 | 33.44 | 0   |
| Avg TU’s Reduction % | 17.23 | 16.73 | 0   |
| MAX TU’s reduction % | 50  | 50  | 0   |

Regarding the position of the center of gravity, the solutions found are satisfying. The average center of gravity coordinates are always close to the center of the TU for each axis, for each type of TU used. Indeed, the maximum deviation of the center of gravity coordinates with respect to the center of the TU is 2% on the X-axis and 3.6% on the Y-axis. For the Z axis the center of gravity depends on the TU occupation, but still the maximum deviation with respect to the half-height of the TU is 10.7%. We performed a sensitivity analysis to evaluate whether the value of $\alpha$ has an impact on the structure of the solution, in terms of number and type of TUs used and in terms of positioning of boxes in the TUs, and thus center of gravity. In particular, we tested values of $\alpha = 50, 100, 1000$. We noticed that solutions did not change, with respect to the ones found with $\alpha = 1$, in terms of number and types of TUs used. Deviations of center of gravity positions were also negligible, lower than 0.1% on all coordinates and all types of TUs, for all values of $\alpha$. Thus, we concluded that the ILS-ATUCP algorithm always provides stable solutions in terms of boxes disposition.

Table 8 reports the average, minimum and maximum position for each coordinates of the center of gravity, throughout all the 300 instances solved, for different values of $\alpha$, i.e., the weight of the center of gravity components in function $f$. The $CI^xy$ and $CI^z$ indexes present satisfying results, with values close to those of the theoretical optimum, which corresponds to the mid $X$, $Y$ and $Z$ values of the TU reported in the first column of the table.
The most noticeable aspect is that throughout all instances solved, the center of gravity is fairly centered along the X and Y axis. The height of the CG is on average slightly below half the maximum height allowed, which is fairly satisfying. One interesting fact is that by enlarging the base of the TU, the height of the center of gravity decreases. This means that by allowing a wider space on the base, the algorithm is capable of finding improved layouts with centers of gravity placed at a smaller height.

We carried out a sensitivity analysis by varying the value of parameter $\alpha$ in equation (1). The aim is to check whether, and how, the solution may vary by changing the weight given to the position of the center of gravity in the computation of the objective function. The values of $\alpha$ are set 1, 50, 100 and 1000. The results show the variation of the value of $\alpha$ brings minimal changes to the solutions found, meaning that the ATUCP algorithm always provides stable solutions in terms of boxes disposition.

This is mainly due to the fact that the search of the optimal placement of boxes in the TU, ensuring stable, compact layouts with centered centers of gravity, is given by the cost function (3) of the 3DBP-Algorithm. By changing the parameters in cost function (3), different objectives are explored (for instance by changing the value of parameter $M$, the TUs generated have a center of gravity in lower position, but less centered on the base), but the solutions found are maintained substantially unvaried with the variation of parameter $\alpha$ of equation (1).

### 5.3 Comparison with company’s practice

The second set of tests is about comparing the solution found by ILS-ATUCP with the solution found by operators of an Italian freight forwarding company. Both real world and randomly generated instances were used for comparison.

Randomly generated instances are a selection of 10 instances generated with the partitioning scheme 1, 10 instances generated with the partitioning scheme 2, and five instances generated with the partitioning scheme 3. The last ones have a low variety of types of boxes and dimensions are fairly standard and fit the TUs. They are used as a benchmark to verify whether the solutions provided by ILS-ATUCP fits with the ones produced by company’s operators.

Results are shown in Table 9. The first two columns report the number of TUs used by the company’s operators and by ILS-ATUCP algorithm, respectively, while the last column reports the difference in the total volume generated by the ILS-ATUCP solution with respect to the solution of the company’s operators. The last column reports the

| $\alpha$ value | TU       | X coordinates | Y coordinates | Z coordinates |
|----------------|----------|---------------|---------------|---------------|
|                | Avg.     | Min | Max | Avg. | Min | Max | Avg. | Min | Max |
| $\alpha = 1$   | 120x80x130 | 58.83 | 55.02 | 64.75 | 38.56 | 37.42 | 43.92 | 60.44 | 48.12 | 75.25 |
|                | 120x80x160 | 58.77 | 55.01 | 64.73 | 38.57 | 37.55 | 43.89 | 76.12 | 70.98 | 98.44 |
|                | 120x100x130 | 58.82 | 55.09 | 65.01 | 48.64 | 44.18 | 55.32 | 59.21 | 48.06 | 73.43 |
|                | 120x100x160 | 58.75 | 54.98 | 64.73 | 48.58 | 45.76 | 54.99 | 73.32 | 68.12 | 89.68 |
| $\alpha = 50$  | 120x80x130 | 58.83 | 55.03 | 65.08 | 58.67 | 55.33 | 65.12 | 58.02 | 47.29 | 71.73 |
|                | 120x80x160 | 58.75 | 54.98 | 64.73 | 48.58 | 45.76 | 54.99 | 73.32 | 68.12 | 89.68 |
| $\alpha = 100$ | 120x80x130 | 58.83 | 55.02 | 64.75 | 58.67 | 55.33 | 65.12 | 58.02 | 47.29 | 71.73 |
|                | 120x80x160 | 58.75 | 54.98 | 64.73 | 48.58 | 45.76 | 54.99 | 73.32 | 68.12 | 89.68 |
| $\alpha = 1000$| 120x80x130 | 58.83 | 55.02 | 64.75 | 58.67 | 55.33 | 65.12 | 58.02 | 47.29 | 71.73 |
|                | 120x80x160 | 58.75 | 54.98 | 64.73 | 48.58 | 45.76 | 54.99 | 73.32 | 68.12 | 89.68 |

The most noticeable aspect is that throughout all instances solved, the center of gravity is fairly centered along the X and Y axis. The height of the CG is on average slightly below half the maximum height allowed, which is fairly satisfying. One interesting fact is that by enlarging the base of the TU, the height of the center of gravity decreases. This means that by allowing a wider space on the base, the algorithm is capable of finding improved layouts with centers of gravity placed at a smaller height.

We carried out a sensitivity analysis by varying the value of parameter $\alpha$ in equation (1). The aim is to check whether, and how, the solution may vary by changing the weight given to the position of the center of gravity in the computation of the objective function. The values of $\alpha$ are set 1, 50, 100 and 1000. The results show the variation of the value of $\alpha$ brings minimal changes to the solutions found, meaning that the ATUCP algorithm always provides stable solutions in terms of boxes disposition.

This is mainly due to the fact that the search of the optimal placement of boxes in the TU, ensuring stable, compact layouts with centered centers of gravity, is given by the cost function (3) of the 3DBP-Algorithm. By changing the parameters in cost function (3), different objectives are explored (for instance by changing the value of parameter $M$, the TUs generated have a center of gravity in lower position, but less centered on the base), but the solutions found are maintained substantially unvaried with the variation of parameter $\alpha$ of equation (1).
CPU time of ILS-ATUCP.

In the simpler instances (rows 1-5 in Table 9), i.e., those generated with partitioning scheme 3, the performance of the ILS-ATUCP algorithm and operators are similar, thus proving that ILS-ATUCP is capable of producing solutions fitting with practical rules in a short computing time.

Results for the 10 instances generated by the first partitioning scheme are reported in rows 6-15 of Table 9 and show that ILS-ATUCP largely outperforms company’s practice, both in terms of number of TUs used and in terms of volume generated. Moreover, the algorithm is capable of finding the solution in very limited time. Results for the instances generated with the second partitioning scheme are reported in rows 16-25. The algorithm again finds much better solutions in every instance.

Table 9: Comparison on randomly generated instances

| N boxes | CI | TU op | ILS-ATUCP | Vol ∆ (%) | CI | Op. | ILS-ATUCP | Vol ∆ (%) |
|---------|----|-------|-----------|-----------|----|-----|-----------|-----------|
| 1       | 64 | 4     | 4         | 0         | 2  |     |           |           |
| 2       | 48 | 3     | 3         | 0         | 3  |     |           |           |
| 3       | 32 | 2     | 2         | 0         | 2  |     |           |           |
| 4       | 128| 6     | 6         | 0         | 1  |     |           |           |
| 5       | 144| 7     | 7         | 0         | 2  |     |           |           |
| Avg.    | 81.30 | 4.40 |           |           | 0  |     |           |           |
| 6       | 233| 3     |           |           | 2  |     |           |           |
| 7       | 135| 9     |           |           | 7  |     |           |           |
| 8       | 96 | 5     |           |           | -12.37 | 5 |     |           |           |
| 9       | 75 | 4     |           |           | -4.05 | 4 |     |           |           |
| 10      | 124| 7     |           |           | -7.61 | 6 |     |           |           |
| 11      | 174| 11    |           |           | -11.34 | 5 |     |           |           |
| 12      | 167| 10    |           |           | -13.27 | 4 |     |           |           |
| 13      | 138| 7     |           |           | -2.77 | 3 |     |           |           |
| 14      | 55 | 4     |           |           | -3.54 | 2 |     |           |           |
| 15      | 312| 21    |           |           | -10.38 | 3 |     |           |           |
| Avg.    | 113.30 | 8.80 |           |           | -8.23 | 3.70 |     |           |           |

For real world instances we compared the difference in the volume generated by the solutions found by the ILS-ATUCP algorithm and by the company operators. The instances are real-world requests received by the company, the size of the boxes is highly heterogeneous between instances but fairly homogeneous in the instance (i.e., each instance has boxes that are similar with only a low number of different box dimensions, but every instance is different from the other). Results are reported in Table 10. Columns TU op, TU ILS-ATUCP and Vol ∆ have the same meaning as before, while columns CI, CI = ILS-ATUCP, column CI Op., CI = ILS-ATUCP and CI Op. report the CI and CI value of the solutions found by ILS-ATUCP and the company’s operators, respectively.

Table 10: Comparison on real world instances

| N boxes | CI | TU op | CI = ILS-ATUCP | CI Op. | CI = ILS-ATUCP | CI Op. |
|---------|----|-------|----------------|--------|----------------|--------|

| N boxes | CI | TU op | CI = ILS-ATUCP | CI Op. | CI = ILS-ATUCP | CI Op. |
|---------|----|-------|----------------|--------|----------------|--------|
| 1       | 64 | 4     | 4.00           | 0.000  | 2.69           | 0.000  |
| 2       | 48 | 3     | 3.00           | 0.000  | 3.00           | 0.000  |
| 3       | 32 | 2     | 2.00           | 0.000  | 2.00           | 0.000  |
| 4       | 128| 6     | 6.00           | 0.000  | 6.00           | 0.000  |
| 5       | 144| 7     | 7.00           | 0.000  | 7.00           | 0.000  |
| Avg.    | 81.30 | 4.40 |           |           | 0              |           |
| 6       | 233| 3     |           |           | 2  |     |           |           |
| 7       | 135| 9     |           |           | 7.32 | 3 |     |           |           |
| 8       | 96 | 5     |           |           | -12.37 | 5 |     |           |           |
| 9       | 75 | 4     |           |           | -4.05 | 4 |     |           |           |
| 10      | 124| 7     |           |           | -7.61 | 6 |     |           |           |
| 11      | 174| 11    |           |           | -11.34 | 5 |     |           |           |
| 12      | 167| 10    |           |           | -13.27 | 4 |     |           |           |
| 13      | 138| 7     |           |           | -2.77 | 3 |     |           |           |
| 14      | 55 | 4     |           |           | -3.54 | 2 |     |           |           |
| 15      | 312| 21    |           |           | -10.38 | 3 |     |           |           |
| Avg.    | 113.30 | 8.80 |           |           | -8.23 | 3.70 |     |           |           |
ILS-ATUCP is capable of better consolidating loose packages generating lower volume in 19 instances out of 25. In five instances the operators and ILS-ATUCP use the same number and type of TUs. In one instance the operators find a solution using a lower volume. Moreover, the number of TUs used by ILS-ATUCP is lower or equal the one used in operators’ solutions for all instances except one. On the other side, the solutions found by the operators present much worse values of both $CI^{xy}$ and $CI^z$ values. The layouts of the solutions provided by the company’s operators lead to centers of gravity which are positioned fairly high when compared to the maximum height of the TU, and far from the center of the base of the TU. The solutions found by ILS-ATUCP, on the other side, have a center of gravity that is much more centered on the base and in a much lower height. This leads to more balanced TUs, increasing the safety of the cargo.

6 Conclusions

This paper introduces the Air Transport Unit Consolidation Problem (ATUCP) where loose packages have to be consolidated in TUs before being transport by air. The goal is to determine the best consolidation setting in order to minimize the cost generated by the number and the volume of the TUs used and, also, to define the best disposal of loose packages inside each TU so that the TU is stable. The problem is extremely complex and finds relevant practical applications in freight forwarding services. Also, the problem is new to the literature.

We proposed a formal description of the problem and an iterated local search heuristic approach based on a construction algorithm for generating an initial feasible solution, followed by local search using different types of neighbourhoods. In addition, a procedure for generating random instances is proposed that allows a comparison with an optimal solutions even for complicated instances.

Computational results on randomly generated instances and on real data show that, on one side, the ILS-ATUCP algorithm produces high-quality solutions when compared with the optimal ones and, on the other side, it largely outperforms the practice of the company providing the real data.

Future research directions includes relaxing assumptions made in the definition of the problem, like non-stackability of TUs. Also, from the methodological point of view, an interesting field of research is related to the development of exact solution approaches for the problem.

References

Alonso, M., Alvarez-Valdes, R., Iori, M., and Parreñó, F. (2019). Mathematical models for multi container loading problems with practical constraints. *Computers & Industrial Engineering*, 127:722–733.

Alvarez-Valdés, R., Parreño, F., and Tamarit, J. M. (2013). A grasp/path relinking algorithm for two-and three-dimensional multiple bin-size bin packing problems. *Computers & Operations Research*, 40(12):3081–3090.

Alvarez-Valdés, R., Parreño, F., and Tamarit, J. M. (2015). Lower bounds for three-dimensional multiple-bin-size bin packing problems. *OR spectrum*, 37(1):49–74.

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

Bortfeldt, A. and Wäscher, G. (2013). Constraints in container loading—a state-of-the-art review. *European Journal of Operational Research*, 229(1):1–20.

Crainic, T. G., Perboli, G., and Tadei, R. (2008). Extreme point-based heuristics for three-dimensional bin packing. *Informs Journal on Computing*, 20(3):368–384.

Crainic, T. G., Perboli, G., and Tadei, R. (2009). Ts2pack: A two-level tabu search for the three-dimensional bin packing problem. *European Journal of Operational Research*, 195(3):744–760.
de Almeida, A. and Figueiredo, M. B. (2010). A particular approach for the three-dimensional packing problem with additional constraints. *Computers & Operations Research, 37*(11):1968–1976.

Dornas, A. H., Martins, F. V. C., Sarubbi, J. F. M., and Wanner, E. F. (2017). Real-polarized genetic algorithm for the three-dimensional bin packing problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 785–792.

Elhedhli, S., Gzara, F., and Yan, Y. F. (2017). A mip-based slicing heuristic for three-dimensional bin packing. *Optimization Letters, 11*(8):1547–1563.

Elhedhli, S., Gzara, F., and Yildiz, B. (2019). Three-dimensional bin packing and mixed-case palletization. *INFORMS Journal on Optimization, 1*(4):323–352.

Gendreau, M., Iori, M., Laporte, G., and Martello, S. (2006). A tabu search algorithm for a routing and container loading problem. *Transportation Science, 40*(3):342–350.

Gonçalves, J. F. and Resende, M. G. (2013). A biased random key genetic algorithm for 2d and 3d bin packing problems. *International Journal of Production Economics, 145*(2):500–510.

Gzara, F., Elhedhli, S., and Yildiz, B. C. (2020). The pallet loading problem: Three-dimensional bin packing with practical constraints. *European Journal of Operational Research, 287*(3):1062–1074.

Ha, C. T., Nguyen, T. T., Bui, L. T., and Wang, R. (2017). An online packing heuristic for the three-dimensional container loading problem in dynamic environments and the physical internet. In *European Conference on the Applications of Evolutionary Computation*, pages 140–155. Springer.

Hiﬁ, M., Kacem, I., Negre, S., and Wu, L. (2010a). Heuristics algorithms based on a linear programming for the three-dimensional bin-packing problem. *Ifac Proceedings Volumes, 43*(8):72–76.

Hiﬁ, M., Kacem, I., N`egre, S., and Wu, L. (2010b). A linear programming approach for the three-dimensional bin-packing problem. *Electronic Notes in Discrete Mathematics, 36*:993–1000.

Kang, K., Moon, I., and Wang, H. (2012). A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem. *Applied Mathematics and Computation, 219*(3):1287–1299.

Li, X., Zhao, Z., and Zhang, K. (2014). A genetic algorithm for the three-dimensional bin packing problem with heterogeneous bins. In *IIE Annual Conference. Proceedings*, page 2039. Institute of Industrial and Systems Engineers (IIE).

Lodi, A., Martello, S., and Vigo, D. (2002). Heuristic algorithms for the three-dimensional bin packing problem. *European Journal of Operational Research, 141*(2):410–420.

Mack, D. and Bortfeldt, A. (2012). A heuristic for solving large bin packing problems in two and three dimensions. *Central European Journal of Operations Research, 20*(2):337–354.

Martello, S., Pisinger, D., and Vigo, D. (2000). The three-dimensional bin packing problem. *Operations Research, 48*(2):256–267.

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

Paquay, C., Limbourg, S., and Schyns, M. (2018). A tailored two-phase constructive heuristic for the three-dimensional multiple bin size bin packing problem with transportation constraints. *European Journal of Operational Research, 267*(1):52–64.

Taylor, G. S., Chan, Y., and Rasool, G. (2017). A three-dimensional bin-packing model: exact multicriteria solution and computational complexity. *Annals of Operations Research, 251*(1):397–427.
7 APPENDIX

7.1 Instance generation procedure

We now describe the procedure to generate instances for the ATUCP.

In the first phase, each instance is defined by the total volume and weight to be shipped. 100 combinations of the values of total volume $V$ and total weight $W$ are randomly generated and reported in Table 11 in the Appendix. The values of $V$ and $W$ are initially generated at random and then adjusted to avoid trivial solutions (for instance, when $W >> V$ the solution is trivial since the weight capacity of the TU is satisfied with a very low number of boxes and thus the layout problem is negligible). For some instances the values of $V$ and $W$ are specifically chosen in order to study whether ILS-ATUCP is capable of identifying the best set of TUs needed to solve the instance. We then solve a mathematical model to identify the best set of TUs that cover the given volume and weight. The solution found by the mathematical model is then used as starting point for the box generation procedure. In this phase, three partitioning schemes are used, described in Section 5.1 Each scheme takes the TUs associated with the solution found by the mathematical model and partition their weight and volume in order to generate boxes. Therefore, from each of the 100 combinations of values of total weight and volume, three instances are generated, one for each partitioning scheme, for a total of 300 instances.

7.1.1 Mathematical Model

The mathematical model used as to generate the instances is the following. Given $TUS$, the set of available types of TUs, the model makes use of the following parameters and variables:

- $V$ is the total volume of the boxes to be packed.
- $W$ is the total weight of the boxes to be packed.
- $v_{tut}$ is the maximum volume capacity of TU of type $tut$.
- $w_{tut}$ is the maximum weight capacity of TU of type $tut$.
- $x_{tut}$ is an integer, non-negative variable representing the number of TUs of type $tut$ used.

The formulation of the mathematical model is the following:

$$\min \sum_{tut \in TUS} v_{tut}x_{tut} + \beta \sum_{tut \in TUS} x_{tut}$$  \hspace{1cm} (4a)
$$\sum_{tut \in TUS} v_{tut}x_{tut} \geq V \hspace{1cm} (4b)$$
$$\sum_{tut \in TUS} w_{tut}x_{tut} \geq W \hspace{1cm} (4c)$$
$$x_{tut} \in \mathbb{N}^+ \forall tut \in TUS \hspace{1cm} (4d)$$

The objective of (4a) is the minimization of the volume and number of the TUs. The objective function is similar to the one of the ATUCP, where the term associated with the center of gravity is missing. We assume that all boxes
have the same weight density (instances in which boxes have different weight density are presented in Section 5.3).

Given how we generate boxes according to the three partitioning schemes described later, the best disposal of boxes always generates a center of gravity which is perfectly centered on the base and placed exactly at half the height of the TU. Note that, anyway, the ILS-ATUCP may not be able to find the best boxes disposition.

The volume is used as an approximation of costs. Constraints (4b) and (4c) ensures that the total volume and weight of the boxes to be packed are respected. The solution of the model is used as the optimal solution to estimate the effectiveness of the ILS-ATUCP in finding good solutions.

In the generated instances, TUS contained the following types of TU: 120x80x130, 120x80x160, 120x100x130, 120x100x160, 120x120x130, 120x120x160. The solutions found are reported in Table 12.

Table 11: Values of total volume (V) and weight (W) for instance generation

| I | V     | W     | I | V     | W     | I | V     | W     |
|---|-------|-------|---|-------|-------|---|-------|-------|
| 1 | 17    | 783   | 36 | 1     | 3,266 | 71 | 24    | 11,663|
| 2 | 5     | 579   | 37 | 1     | 1,765 | 72 | 22    | 4,816 |
| 3 | 13    | 4,268 | 38 | 26    | 2,738 | 73 | 3     | 4,500 |
| 4 | 30    | 5,076 | 39 | 24    | 1,525 | 74 | 11    | 8,031 |
| 5 | 10    | 3,463 | 40 | 25    | 182  | 75 | 25    | 14,057|
| 6 | 15    | 3,242 | 41 | 18    | 3,000 | 76 | 28    | 15,980|
| 7 | 18    | 1,869 | 42 | 7     | 4,081 | 77 | 4     | 3,476 |
| 8 | 10    | 1,519 | 43 | 18    | 3,546 | 78 | 17    | 2,959 |
| 9 | 30    | 4,329 | 44 | 28    | 2,693 | 79 | 29    | 1,025 |
| 10| 12    | 2,433 | 45 | 12    | 3,546 | 80 | 26    | 1,695 |
| 11| 27    | 113   | 46 | 19    | 3,131 | 81 | 26    | 4,933 |
| 12| 19    | 2,268 | 47 | 29    | 3,708 | 82 | 25    | 4,092 |
| 13| 20    | 2,199 | 48 | 18    | 1,230 | 83 | 3     | 1,578 |
| 14| 11    | 4,103 | 49 | 28    | 279  | 84 | 18    | 4,082 |
| 15| 3     | 4,713 | 50 | 20    | 865  | 85 | 30    | 14,555|
| 16| 18    | 2,346 | 51 | 23    | 1,069 | 86 | 29    | 14,766|
| 17| 23    | 3,829 | 52 | 13    | 299  | 87 | 4     | 1,417 |
| 18| 3     | 2,062 | 53 | 12    | 919  | 88 | 18    | 4,307 |
| 19| 27    | 4,300 | 54 | 30    | 16,452|89 | 17    | 4,401 |
| 20| 16    | 3,941 | 55 | 14    | 2,617 |90 | 8     | 1,428 |
| 21| 1     | 3,087 | 56 | 2     | 2,913 |91 | 20    | 12,843|
| 22| 15    | 4,276 | 57 | 29    | 14,500|92 | 9     | 4,156 |
| 23| 10    | 3,962 | 58 | 6     | 1,088 |93 | 15    | 11,353|
| 24| 12    | 4,268 | 59 | 1     | 949  |94 | 20    | 4,866 |
| 25| 18    | 4,348 | 60 | 5     | 253  |95 | 26    | 12,692|
| 26| 20    | 5,829 | 61 | 28    | 3,047 |96 | 22    | 11,209|
| 27| 3     | 1,480 | 62 | 16    | 11,706|97 | 26    | 932  |
| 28| 15    | 4,872 | 63 | 25    | 15,575|98 | 14    | 4,888 |
| 29| 3     | 2,655 | 64 | 10    | 1,643 |99 | 29    | 11,311|
| 30| 25    | 412   | 65 | 13    | 8,389 |100| 13    | 5,656 |
| 31| 22    | 2,579 | 66 | 9     | 5,016 |   |       |       |
| 32| 8     | 2,806 | 67 | 1     | 1,500 |   |       |       |
| 33| 4     | 366   | 68 | 26    | 11,823|   |       |       |
| 34| 6     | 4,291 | 69 | 2     | 3,193 |   |       |       |
| 35| 16    | 3,398 | 70 | 15    | 6,144|   |       |       |
Table 12: Optimal lower bound solutions

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|----|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 2 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 3 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Total

34
Table 13: Partitioning algorithm statistics

| I | A1 Boxes | A2 Boxes | A3 Boxes | A1 unique dimensions | A2 unique dimensions | A3 unique dimensions |
|---|---------|---------|---------|---------------------|---------------------|---------------------|
| 1 | 223    | 771    | 699     | 223                 | 744                 | 4                   |
| 2 | 65     | 230    | 244     | 62                  | 221                 | 3                   |
| 3 | 155    | 637    | 607     | 155                 | 609                 | 4                   |
| 4 | 348    | 1386   | 1463    | 344                 | 1328                | 5                   |
| 5 | 101    | 485    | 364     | 101                 | 458                 | 4                   |
| 6 | 184    | 734    | 756     | 184                 | 706                 | 3                   |
| 7 | 174    | 808    | 567     | 173                 | 773                 | 3                   |
| 8 | 109    | 470    | 426     | 108                 | 455                 | 6                   |
| 9 | 281    | 1427   | 1463    | 278                 | 1329                | 4                   |
| 10| 158    | 566    | 526     | 156                 | 546                 | 5                   |
| 11| 311    | 1233   | 948     | 309                 | 1174                | 5                   |
| 12| 217    | 863    | 698     | 215                 | 822                 | 3                   |
| 13| 204    | 907    | 791     | 203                 | 848                 | 5                   |
| 14| 146    | 528    | 507     | 145                 | 495                 | 6                   |
| 15| 43     | 148    | 125     | 43                  | 141                 | 1                   |
| 16| 260    | 917    | 816     | 259                 | 858                 | 4                   |
| 17| 236    | 1090   | 1083    | 234                 | 1038                | 4                   |
| 18| 29     | 140    | 144     | 28                  | 137                 | 3                   |
| 19| 287    | 1247   | 1059    | 283                 | 1186                | 6                   |
| 20| 200    | 769    | 780     | 198                 | 718                 | 4                   |
| 21| 14     | 60     | 25      | 13                  | 57                  | 1                   |
| 22| 186    | 695    | 756     | 185                 | 673                 | 2                   |
| 23| 114    | 467    | 364     | 114                 | 444                 | 4                   |
| 24| 136    | 515    | 556     | 135                 | 494                 | 4                   |
| 25| 233    | 916    | 872     | 230                 | 868                 | 4                   |
| 26| 241    | 925    | 791     | 237                 | 887                 | 5                   |
| 27| 23     | 159    | 125     | 23                  | 151                 | 1                   |
| 28| 316    | 1216   | 1119    | 315                 | 1168                | 5                   |
| 29| 38     | 143    | 129     | 38                  | 142                 | 3                   |
| 30| 273    | 1190   | 952     | 270                 | 1118                | 5                   |
| 31| 258    | 1063   | 1100    | 254                 | 1004                | 4                   |
| 32| 88     | 351    | 327     | 88                  | 338                 | 3                   |
| 33| 67     | 229    | 173     | 67                  | 218                 | 2                   |
| 34| 60     | 285    | 193     | 60                  | 273                 | 2                   |
| 35| 205    | 771    | 780     | 204                 | 735                 | 4                   |
| 36| 156    | 557    | 507     | 156                 | 536                 | 6                   |
| 37| 11     | 42     | 21      | 11                  | 39                  | 1                   |
| 38| 326    | 1270   | 1256    | 323                 | 1209                | 3                   |
| 39| 278    | 1161   | 920     | 278                 | 1093                | 3                   |
| 40| 285    | 1203   | 952     | 284                 | 1134                | 5                   |
| 41| 210    | 702    | 872     | 210                 | 686                 | 4                   |
| 42| 75     | 259    | 288     | 75                  | 255                 | 4                   |
| 43| 73     | 148    | 176     | 73                  | 148                 | 2                   |
| 44| 357    | 1201   | 1264    | 352                 | 1180                | 6                   |
| 45| 131    | 555    | 526     | 128                 | 540                 | 5                   |
| 46| 241    | 832    | 967     | 239                 | 809                 | 5                   |
| 47| 333    | 1354   | 1044    | 332                 | 1296                | 6                   |
| 48| 203    | 915    | 691     | 201                 | 878                 | 5                   |
| 49| 345    | 1367   | 1035    | 341                 | 1317                | 6                   |
| 50| 182    | 964    | 791     | 180                 | 939                 | 5                   |