Constrained-optimization in a 3D bin packing realistic problem

Yamil Mateo Rodríguez,
Escuela Técnica Superior
de Ingenieros de Telecomunicación,
Universidad Politécnica de Madrid,
Madrid, Spain

Juan Carlos Dueñas López,
Escuela Técnica Superior
de Ingenieros de Telecomunicación,
Universidad Politécnica de Madrid,
Madrid, Spain

Javier Andión Jiménez,
Escuela Técnica Superior
de Ingenieros de Telecomunicación,
Universidad Politécnica de Madrid,
Madrid, Spain

2021 International Young Engineers Forum (YEF-ECE) | 978-1-6654-1264-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/YEF-ECE52297.2021.9505150

***Abstract*—This article describes the implementation of a new approach in 3D bin packing optimization and tests its performance in different scenarios. It takes into consideration most of the real constraints a logistics company may face when loading cargo and produces several feasible solutions for a system or a human operator to choose from. As a result of the wide range of constraints the algorithm can handle, it is highly customizable and adaptable to different kinds of companies with different interests and procedures. Optimizing bin packing within the current fast-paced supply chain reduces complexity left to humans, decreases costs and frequency in problems that may damage the customer experience and helps to minimise traffic and carbon emissions by maximizing volume utilization.**

I. Introduction

Taking as a starting point the informs of Industry 4.0 [1] and Industry 5.0 [2], the process of digitalization has gone from the "Automatisation Revolution" to the "Human-centric and sustainable" era. During this transition, logistics companies have realised the importance of investing in technology to be able to meet the demand of supply, that is why the global market for Digital Transformation Spending in Logistics is estimated at US$45.3 Billion in the year 2020 and projected to double by 2027 [3]. The human-centric approach is a concept in which the technology guides the worker in adapting to his/her skills instead of trying to replace them.

Packing is one of the duties the workforce of a warehouse has to carry out daily, it is a complex and physically demanding job if the packing plan is not good enough. Unfortunately, humans are not capable of efficiently compute many combinations to get the best solution that will meet all the requirements of the cargo and make it as easier as possible for individual labourers. Packing optimization is one of the biggest challenges for the logistics industry, in a recent trend report from DHL [4], 90% of companies say the packaging is on top of their agenda, being their current difficulties: maintaining a reasonable packing spend, reducing shipment damage and optimizing transport capacity. These three are highly correlated, being too general with a few set of pre-sized boxes eases packing but reduces the volume optimisation, on the other side handling a wide range of carton boxes increases costs and adds several problems in packing and volume optimization let to the fulfilment human operators. The mentioned research report [4] provides a couple of details exposing this necessity, "Packaging density optimization is an industry-wide challenge. 24% of the volume of the average shipping container is empty space. In some e-commerce categories, parcels contain up to 40% air".

This problem is a Container Loading Problem (CLP) part of the family of combinatorial optimization problems. More concretely our algorithm is focused on resolving -in a 3D space- a mix of Single Large Object Placement Problem (SLOPP) and Single Knapsack Problem (SKP) defined in [5] consisting of given two sets of items, one of larger items $L = \{l_1, ..., l_n\}\ n \in \mathbb{Z}$, the container or containers and the other set for smaller items $I = \{i_1, ..., i_n\}\ n \in \mathbb{Z}$, the packets. The objective is to maximize different outputs in our case those are the volume, the weight and the dimensional weight utilization of I in any $l_n \in L$. The CLP is a non-deterministic Polynomial-time Hard (NP-hard) problem, these kind of problems are approached using heuristics and approximation algorithms. In this case, its hardness depends on the constrains exposed in the next section.

II. Literature Review

To evaluating the state-of-the-art procedures solving CLPs, it is important to present most of the real constraints these packing problems face. Wäscher[6] provides an in-depth analysis of the real-world constraints so grouped by his criteria we will briefly explain these constraints applied to our scenario:

1) Container-related: referred to physics constraints of the truck.
   - *Weight Limits (C1)*: maximum tonnage allowed to the truck. Implemented during the packing.
   - *Weight Distribution (C2)*: distribution along the width, height and length axes of the container. Our heuristic guarantees this constraint in the horizontal axis imposing a maximum deviation margin of $\pm 20\%$ between the actual middle and the mass center. Implemented during the packing.
2) Item-related:
   - *Loading priorities (C3)*: items which may have preference over the rest. In our case, it will be a binary condition and the priority items have to be packed or the solution is not valid. Implemented both during and after "core".

- *Orientations (C4)*: allowed orientations of an item (i.e, some must not be turned around). Implemented. In [7] where introduced these:

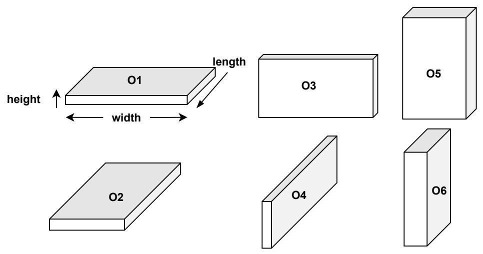

Fig. 1. All possible orientations

- *Stacking (C5)*: there is a dynamic weight limit an item can support depending on its fragility. In our case, there are fragile, considered as "breakability" items which can only bear half of their weight above and the normal items cannot directly hold more than their weight. Implemented during the packing.

3) Cargo-related:
   - *Complete shipment (C6)*: cluster of items are part of the same parcel. Implemented in our algorithm but making use of the priority to create these subgroups.
   - *Allocation (C7)*: some items may have restrictions sharing the same container or zone i.e, toxic fluids with food goods. Not implemented in our case.
4) Positioning-related:
   - *Dangerous items (C8)*: they do not have restrictions of sharing a space but have to be located in a exact position of the truck. Not implemented in our case.
   - *Relative positioning (C9)*: some kind items may need to be packed to each others with no exact position within the truck. Not implemented.
   - *Multi-drop situations (C10)*: set of items for different customers which have to be sequentially unloaded sorted by the position of the customer on a route. Implemented strictly during the packing.
5) Load-related:
   - *Stability (C11)*: how stable is the set items once loaded. We guarantee at least a 75% base area support, but in most cases it is close to 100%. Implemented during the packing.
   - *Complexity (C12)*: difficulty of a solution for human operators in the loading and unloading process. Not implemented because it is really subjective.

One of the most common approaches to solving CLPs is using mixed-integer linear programming (MILP), the problem is that in most cases these solutions do not give good results with more than one constraint at a time, and some of these constaints are extremely difficult to adapt to a MILP approach [6]. Many solutions have been proposed recently, these heuristics combine different kinds of algorithms: GRASP [8], Genetic Algorithms [9], Neighbourhood Structures [10], Deep Reinforcement Learning [11]. In these approaches, the authors usually compute between three to eight [12]–[14] of the constraints.

III. Our solution

Due to the great number of restrictions taken into consideration, and goods results provided in many examples in literature such [12] and [15], our algorithm is a recursive greedy randomized constructive heuristic (RGRCH) which has the following high-level architecture:

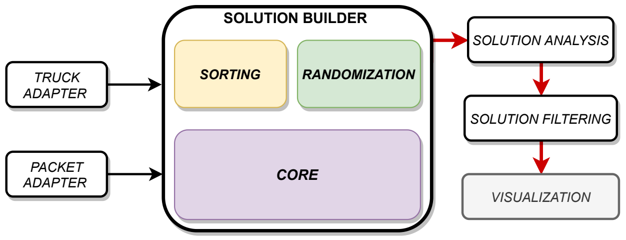

Fig. 2. Algorithm high-level architecture

And a low-level architecture:

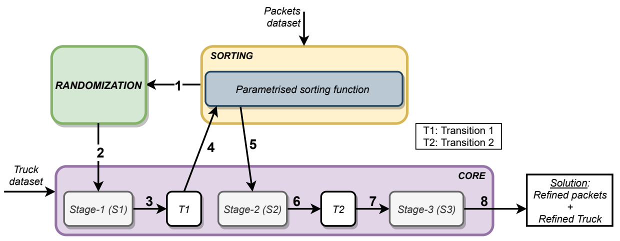

Fig. 3. Algorithm low-level architecture

The objective of the algorithm is to find the combinations of items that maximises primarily volume utilisation of the container but also weight and dimensional weight. But before we begin explaining in depth the algorithm characteristics, in the next figure we present the geometric convention we will be using for both truck containers and items, we reuse the vertices names as in [12]:

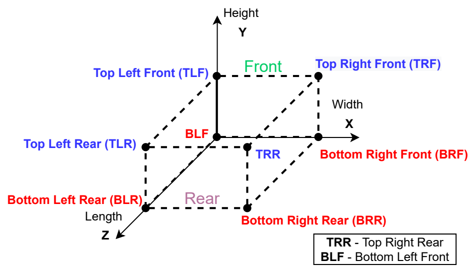

Fig. 4. Corners notation in containers and items

In addition, the insertion of the items in the truck its done following the well-known extreme points technique [16], that manages the creation and the projection of not supported

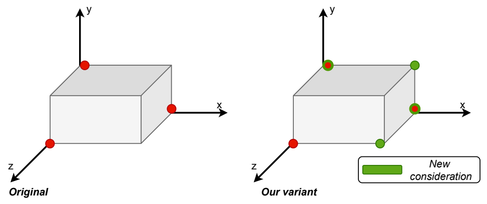

Fig. 5. Extreme points modification

potential points with the following new considerations that have maximized out results:

All of the coloured points of the figure are Potential Points (PP) which means they are possible positions in which to insert any other item. Even though we may refer to insertion in several part of this thesis there two kinds of insertions; a *partial/evaluation insertion* in which item's mass center is adapted to the PP to evaluate each of the constraints and determine if its *definitive insertion* is possible.

The process of how items are inserted follows the next logic in our creation; given an item i_1 in a set of items $S = \{i_1, .., i_n\}$ with width w_{i_1}, height h_{i_1} and length l_{i_1}, a Potential Point P with a 3D coordinates $(x_p,\ y_p,\ z_p)$ and a container C with width w_c, height h_c and length l_c and m_d as the lowest dimension $[w,\ h,\ l]$ if the coordinate x_p is lower than x_l where $x_l = w_c - m_d$ the center of mass of the item will be $(x_p + w_{i_1}/2,\ y_p + h_{i_1}/2,\ z_p + l_{i_1}/)$ otherwise the center of mass if the item will be $(x_p - w_{i_1}/2,\ y_p + h_{i_1}/2,\ z_p + l_{i_1}/2)$. These modifications of Figure 5 are:

- TLF: it is considered as it was in the original but has this new interpretation; given an item i_1 in a set of items $S = \{i_1,\ ...,\ i_n\}$ with width w_{i_1}, height h_{i_1} and length l_{i_1} definitively inserted into a potential point P with a 3D coordinates $(x_p,\ y_p,\ z_p)$ and a container C with width w_c, height h_c and length l_c and m_d as the lowest dimension $[w,\ h,\ l]$ if the item's TLF y-coordinate, $y_p + h_{i_1}$, is greater than y_l where $y_l = h_c - m_d$, thus the TLF is not generated. This thought avoids creating a new PP in which none of the items of the route can be inserted because they will not fit. It also saves the computation time of checking a non-feasible PP.
- BRF: it is considered as it was in the original but has this new interpretation; given an item i_1 in a set of items $S = \{i_1,\ ...,\ i_n\}$ with width w_{i_1}, height h_{i_1} and length l_{i_1} definitively inserted into a potential point P with a 3D coordinates $(x_p,\ y_p,\ z_p)$ and a container C with width w_c, height h_c and length l_c and m_d as the lowest dimension $[w,\ h,\ l]$ if the item's BRF x-coordinate, $x_p + w_{i_1}$, is greater than x_l where $x_l = w_c - m_d$, then the BRF is not generated. This logic is identical to the one with the height limit in TLF. It just does not make sense to create a new PP in which none of the items to be packed could fit nor be placed because it will overlap i_1 concretely when the coordinate x_p is greater than x_l.
- TRF: it is not considered in the original technique. In our heuristic, this new PP is created if an item is definitely inserted in Stage-2 (S2) or Stage-3 (S3) of the core or if it so close to the right edge of the container that no other item would fit. It also follows the same logic about height limits explained in TLF, so the point will not be created if there is no space for any other item to be inserted.
- BRR: it is not considered in the original technique. In our heuristic, this new PP is created every time a BRF is not created for the fitting reasons explained before. The solution to create a potential point when the right side of the packet is so close to the right edge of the container that no item would fit or would overlap the item that created the PP is to generate BRR as the PP instead of BRF, solving all overlapping issues with the item that created the PP.

A. Packet adapter

In this module, we add the volumetric/dimensional weight to a set of items. The volumetric weight is a concept in freight transport used to better estimate the cost of transportation of an item. The IATA factor in road transport we will be using is $1/330[m^3/kg]$. In addition, we make a "*change orientation to best*" is based on given a set of items $S = \{i_1, .., i_n\}$ with average weight W and standard deviation σ_w, then i_n is reoriented to the orientation o_n from a set of feasible orientations $F = \{o_1, o_2, ..., o_n\}$ that maximizes its chances of generating a best insertion and with s_F as the size of the set F. First, F is sorted by a criteria of maximum area so $F' = \{o'_1, o'_2, ..., o'_n\}$ where o'_1 is the orientation with the lowest base area and o'_n the one with the greatest base area and $F = \{o_1, o_2, o_3, o_4, o_5, o_6\}$ if all orientations are feasible and in our instances $F = \{o_1, o_2, o_j, o_k\}$ if some are limited. Despite Figure 6 the algorithm will work with any other kind of distribution. The weight is proportional to the volume by a density constant depending on the material, and typically, heavy items are more voluminous. Hence, we want this kind of items to be oriented in the direction that maximizes their base area improving Stacking (C5) and Stability (C11). Then, for an item i_n, the following figure is a graphical example of all the explained in this section about orientations:

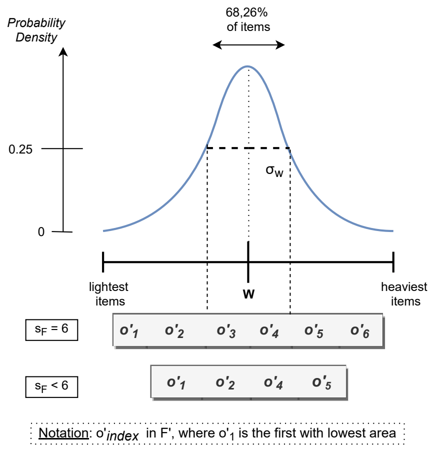

Fig. 6. Packet adapter best orientation algorithm

B. Truck Adapter

This module computes predefined conditions by the carrier. Firstly, they have to decide the number of subzones that are in a container, by default is four, meaning there are four subcontainers inside the actual container.

C. Sorting

This module is in charge of sorting a set of items according to an objective function that will optimize the satisfaction of several constraints and maximize its total volume and weight utilization. Given a set of items $S = \{i_1, .., i_n\}$ with $P_{max} \in \{0, 1\}$ as the maximum priority level of S then W_{max} as the maximum weight for any $i_j \in S$, V_{max} as the maximum volume for any $i_k \in S$ where i_k and i_j could be different or the same and $d \in \{1, ..., n\}$ is the number of customers in the route. Then for an item i_l where $(w_{i_l}, v_{i_l}, p_{i_l})$ are their weight, volume and priority respectively, and $dc_{i_l} \in \{0, ..., d-1\}$ the destination code inside the route, the fitness function is:

$$\begin{aligned} Fitness_{sort} = (d - dc_{i_l}) + \\ + \left(\frac{v_{i_l}}{V_{max}} \cdot \alpha + \frac{w_{i_l}}{W_{max}} \cdot \beta + \frac{p_{i_l}}{P_{max}} \cdot \theta \right) \end{aligned} \quad (1)$$

Hence, the set S will be sorted resulting in $S' = \{i'_1, .., i'_n\}$ where i'_1 is the item which gets the best value and i'_n the packet which gets the worst value in Expression 5. In Figure III two processes require the sorting module. The first one is the direct output of the packet adapter module, and the second is in the transition between points number four and five of the same figure. In these two phases, the value of the parameters $(\alpha,\ \beta,\ \theta)$ will vary. *The values are calculated dynamically, but they always sum 1*. The only restriction for the parameters is that $\alpha > \theta\ and\ \beta > \theta$ because it is more important to build a solid base that to enforce the packing of any light weight/volume priority item.

D. Randomization

The randomization phase got inspiration from [12], in which the authors expressed all the benefits of this phase in their approach. However, it has been hardly modified, resulting in the same formulation. Given a set of sorted items S' for each item $i'_k \in S$ it has three randomization phases:

1) Swap by volume: this phase exchanges one item i_j with its consecutive i_{j+1} with a 50% probability if the ratio between their volumes $v_{i_j}/v_{i_{j+1}} \in [0.85, 1.15]$ and if their destination codes are the same, $dc_{i_j} = dc_{i_{j+1}}$.
2) Swap by weight: this phase exchanges one item i_j with its consecutive i_{j+1} with a 50% probability if the ratio between their weights $w_{i_j}/w_{i_{j+1}} \in [0.85, 1.15]$ and if their destination codes are the same, $dc_{i_j} = dc_{i_{j+1}}$.
3) Swap by priority: this phase exchanges one item i_j with any other i_q where $i_q \neq i_j$ with a 50% probability if they have the same priority, $p_j = p_{j+1}$, the ratio between their weights $w_{i_j}/w_{i_{j+1}} \in [0.85, 1.15]$, the the ratio between their volumes $v_{i_j}/v_{i_{j+1}} \in [0.85, 1.15]$ and if their destination codes are the same, $dc_{i_j} = dc_{i_{j+1}}$.

Our modification implied more strict similarity ratios, and making the randomization locally within a customer cluster which guarantees solutions with less unloading obstacles. Since our heuristic is recursive, this module is fundamental to generating new solutions across the iterations of different algorithm instances over a dataset.

E. Core

Let us go over each of the submodules in figure III.

1) Stage-1 (S1): the main purpose of this stage is to create a solid base for the next stages. After several approaches during the development, we concluded that this initial phase is the key to achieving good results and fulfilling the constraints' requirements. It has the result of the randomization module as the input, and its objective is to create a horizontal layer of items *on the floor*. It does not consider potential points that are not on the floor. Our heuristic takes very strictly Multi-drop situations (C10) so what we do to improve the clustering is build bases for each customer that will occupy a part of the base of the container proportional to the number of items a destination has in the route. The distribution follows this reasoning; given a set of items $G2 = \{i_1,\ ...,\ i_n\}$, the size of G2 as s_{G2}, the average weight of G2 as W_{G2}, the truck's base area t_A, the number of customer in the route $d \in \{1,\ ...,\ m\}$ $m > 3$, a set of sizes that correspond to the number of items per destination in $G2$ as $D = \{a_{d_1},\ ...,\ a_{d_n}\}$, a set of filtered items $G2' = \{i'_1,\ ...,\ i'_n\}$ which $G2' \subseteq G2$ of the items whose weight is greater than $0,5 \cdot W_{G2}$, the size of G2' as $s_{G2'}$ and a set of sizes that correspond to the number of items per destination in $G2'$ as $D' = \{a'_{d_1},\ ...,\ a'_{d_n}\}$ then for each customer:

$$\begin{aligned} MaxArea_{d_j} &= \frac{\frac{a_{d_1}}{s_{G2}} + \frac{a'_{d_1}}{s_{G2'}}}{2} \cdot f_j \cdot t_A \\ f_j &= \begin{cases} if\ j = m\ then\ f_j = 1 \\ otherwise\ f_j = 1 - \frac{m-j+1}{100} \end{cases} \end{aligned} \quad (2)$$

Expression 2 manages the area distribution for each customer, it basically assigns a proportional part of the truck's area corresponding to the mean of both the relative distribution of presence per destination regarding the total items for non-filtered candidate items and filtered items multiplied by a factor. This factor makes the distribution fairer since even in 2D bin packing, for strongly heterogeneous datasets with several customers, it is usual to have an accumulation of waste space out of the current proportion assigned to each destination that affects the area utilization of the destinations closer to the rear of the truck, leaving less area to the last customer which has a big negative impact especially in instances with priority items. Therefore, the container filling algorithm in this phase is basically, given a set of Potential Points PP_{list} and a set of items I_{list} while the current area assigned to a destination does not exceed its maximum container area allowed or there are candidate items that would fit, thus evaluate each $pinPP_{list}$ on the floor and over a set of orientations $O = \{o_{best}, o_{random}\}$, o_{best} according to Figure 6, iterate $i \in I$ over each o checking feasibility of its partial insertion and rating it according to a fitness function, then choose the best partial insertion which will be the one with the best fitness and make the definitive insertion. The fitness function, *which returns a value for each feasible partial insertion* and is used to determine the "best PP", in this stage tries to maximize; the packing of items with heavy relative weight regarding the maximum weight of the set of items which results in better stacking performance, proximity to the rear of the truck translating into better volume utilization and maximize clustering to minimize unloading obstacles. Given a set of items $G2 = \{i_1, \ ..., \ i_n\}$, the maximum weight of $G2$ as W_{G2}, an item i_j with weight w_{i_j}, a container C with width w_c, height h_c and length l_c, and a potential point P with coordinates $(x_p, \ y_p, \ z_p)$:

$$Fitness_{S1} = \left(1 - \frac{z_p}{l_c}\right) \cdot \alpha + \frac{w_{i_j}}{W_{G2}} \cdot \beta + S_{cond} \cdot \theta \quad (3)$$

The value of the parameters $(\alpha, \ \beta, \ \theta)$ have a restriction that $\theta > \alpha \ and \ \theta > \beta$ but sum 1, because a good clustering is so important in this phase. S_{cond} refers to what we have called *"surrounding condition"* which will be introduced for this fitness but is used in the fitness of stage-2 and stage-3. This condition is a penalty parameter that gets weighted by θ. Given an item $i_j \in G2$, the number of destinations d from $G2$, the destination of the item i_j as c_i, a subset of placed items $G2' \subseteq G2$, the size of $G2'$ as s_2', the number of items in $G2'$, as n_d, with the same destination than i_j, a subset of the five nearest items to i_j as $G2'' \subseteq G2$, its size as s_2'', a subset of the nearest placed items with the same destination $G2''' \subseteq G2''$ and its size as s_2''', then:

$$S_r = \begin{cases} if \ n_d < 5 \ then \ S_r = 1 - \frac{c_i}{d} \\ elseif \ s_2''' < 1 \ then \ S_r = -0,3 \\ otherwise \ S_r = \frac{s_2'''}{max(s_2'',1)} \end{cases} \quad (4)$$

It rewards or penalizes depending how good is the clustering.

2) Transition 1 (T1): this part of the core has the function to call the sorting module, what was called before as the second phase inside sorting. The parameters $(\alpha, \ \beta, \ \theta)$ are tuned dynamically as well, but in this case giving more weight to the priority with the purpose of getting more feasible solutions, so the restriction is that $\theta > \alpha \ and \ \theta > \beta$ always normalizing to sum 1.
3) Stage-2 and Stage-3: although these stages have different input and outputs as in figure III, they share the same reasoning in their fitness function to determine the best potential point for an item. Hence, given a set of items $G2 = \{i_1, \ ..., \ i_n\}$, potential point P with a 3D coordinates $(x_p, \ y_p, \ z_p)$, an item i_j with weight w_{i_j} and a mass center with coordinates $(x_{i_{mc}}, \ y_{i_{mc}}, \ z_{i_{mc}})$ with a partial insertion in P, a subset of not yet placed items $G2' \subseteq G2$, a subset of items $G2'' \subseteq G2$ underneath i_j and in direct contact with it whereas $i_k \in G2''$ is the item that created the potential point P, the maximum weight of $G2$ as W_{G2}, a container C with width w_c, height h_c and length l_c, then:

$$Fit_{S2-3} = \left(1 - \frac{z_p}{l_c}\right) \cdot \alpha + HWR \cdot \beta + S_{cond} \cdot \theta + AC \cdot \gamma \quad (5)$$

The first expression, that also appeared in Expression 5 but was not explained, it tries to maximize the volume utilization, being greater the nearest the z-coordinate of the potential point is to the front of the container. ***HWR*** stands for Height Weight Relation, this parameter has the following expression:

$$HWR = \frac{y_{i_{mc}}}{h_c} \cdot \frac{w_{i_j}}{W_{G2}} \quad (6)$$

The weight of ***HWR*** is not big but can be decisive in some situations, its primary purpose is to find if the height of the item, after a partial insertion is a good choice for its weight compared to the maximum weight of the not yet packed items. Thus, AC stands for Area Condition, and has this following aspect:

$$AC = 1 - \left|\frac{BottomArea(i_j)}{TopArea(i_k)} - 1\right| \quad (7)$$

This condition has the intention to reward the stacking of items with similar areas, the function penalizes more it the item above is much smaller than the item behind because it will be, in most, cases wasting potential supporting area. There is one last fitness related aspect to highlight, when two potential insertions for an item have the same value, our algorithm chooses one randomly, which also contributes to get different solutions over the iterations of an instance.
4) Transition 2 (T2): it is in charge of restoring overlapped potential points during the insertion process of S2, putting the on top of the item which is overlapping.

IV. Validation

To compute the algorithm we used a 100 core Google Cloud machine parallelized with Joblib. Each experiment is composed of 360 iterations over the whole high-level infrastructure, the volume of the whole set of items to be packed is approximately the same as the truck used as a model, which is a truck of 13.6 metres of length, and 2.45 of width and height, with a total volume of 81000 cubic metres.

Figure 7 is an overview of the whole set of experiments, with the number of unique items in each experiment, their characteristics such as the number of customers or if it has breakability or priority items. In this figure, there are only presented the feasible solutions, as you can see there are experiments which have more samples than others.

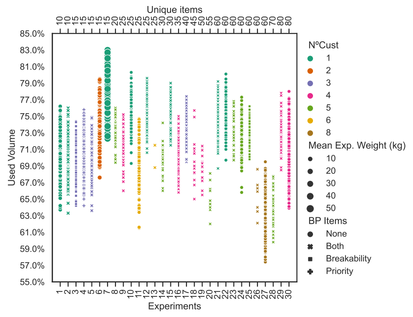
Fig. 7. Experiments overview

Apart from the constraints explained in the other sections, we consider as feasible a solution that satisfy all the conditions enforced during the algorithm plus a filtering of Weight Distribution (C2) and Loading priorities (C3). Then, in the next two Figures, 8 and 10, we expose the performance of out algorithm in two constraints that are very important.

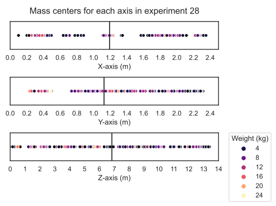
Fig. 8. Mass center proof

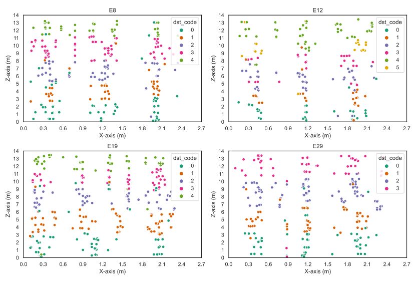
Fig. 9. Clusterization overview

Then for each experiment this is the time taken in each iteration:

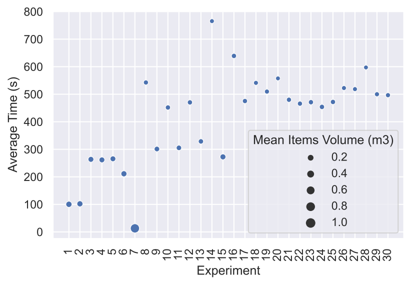
Fig. 10. Time for iteration

V. Conclusion

The results are pretty good and there are some insights that we can conclude from this algorithm. First, the instances with priority and breakability items have a better performance in volume utilization as a result of the imposing of priority items to be packed and the negative impact in the stacking constraints if there is a breakability item packed. Then, the satisfaction of the multi-drop constraints is outstanding, having very few unloading obstacles and the stability is achieved in the three axes. Finally, we can see that the bigger on average the items of a experiment are, less demanding is in terms of computation.

Acknowledgment

This work has been partially funded by the project Sistema de Ordenación de una Red de Transporte (mercancías y viajeros) y Logística con Trazabilidad (SORTLOT) from the Programme CIEN-CDTI, under a contract from InnoCV Solutions.

References

[1] C. M. Jan Smit and M. Carlberg, "Industry 4.0," *Publications Office of the UE*, 2016.

[2] L. D. N. Maija Breque and A. Petridis, "Industry 5.0," *The name of the journal*, pp. 1–17, 2021.

[3] A. J. Mayank Halmare and S. Mutreja, "Global opportunity analysis and industry forecast 2017–2027," *Allied Market Research*, pp. 40–45, 2021.

[4] M. Heck and J. Ward, "Rethinking packing," *DHL Customer Solutions and Innovation Represented by Matthias Heutger*, 2021.

[5] H. H. Gerhard Wascher and H. Schumann, "An improved typology of cutting and packing problems," *European Journal of Operational Research*, 2006.

[6] A. Bortfeldt and G. Wäscher, "Constraints in container loading – a state-of-the-art review," *European Journal of Operational Research*, 2012.

[7] T. Fanslau and A. Bortfeldt, "A tree search algorithm for solving the container loading problem," *INFORMS Journal on Computing*, 2010.

[8] T. A. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, 1995.

[9] J. F. Gonçalves and M. G.C.Resende, "A biased random key genetic algorithm for 2d and 3d bin packing problems," *International Journal of Production Economics*, 2013.

[10] J. O. J. T. F.Parreño, R.Alvarez-Valdes, "Neighborhood structures for the container loading problem: a vns implementation," *Journal of Heuristics*, vol. 16, pp. 1–22, 2010.

[11] V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *arXiv e-prints*, 2018.

[12] M. Gajda, A. Trivella, R. Mansini, and D. Pisinger, "An optimization approach for a complex real-life container loading problem," *SSRN*, 2020.

[13] "A matheuristic framework for the three-dimensional single large object placement problem with practical constraints," *Computers Operations Research*, vol. 124, p. 105058, 2020.

[14] "Automating the planning of container loading for atlas copco: Coping with real-life stacking and stability constraint," *European Journal of Operational Research*, vol. 280, pp. 1018–1034, 2019.

[15] R. A.-V. D. Alvarez Martínez and F.Parreño, "A grasp algorithm for the container loading problem with multi-drop constraints," *Pesquisa Operacional*, 2015.

[16] G. P. Teodor Gabriel Crainic and R. Tadei, "Extreme point-based heuristics for three-dimensional bin packing," *Informs Journal on Computing*, vol. 3, no. 20, pp. 368–384, 2008.