IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 12 January 2024, accepted 12 March 2024, date of publication 19 March 2024, date of current version 25 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3378063

RESEARCH ARTICLE

A Multi-Heuristic Algorithm for Multi-Container 3-D Bin Packing Problem Optimization Using Real World Constraints

ANAN ASHRABI ANANNO AND LUIS RIBEIRO, (Senior Member, IEEE)
Department of Management and Engineering, Linköping University, 581 83 Linköping, Sweden

Corresponding author: Luis Ribeiro (luis.ribeiro@liu.se)

This work was supported by the Strategic Innovation Program Produktion2030, a joint venture between Vinnova, Formas, and the Swedish Energy Agency under the scope of the project titled FLAP—Hållbar och flexibel automatisering av säsongsproduktion genom dynamisk resurshantering under Grant 2021-01283.

ABSTRACT With the growing demand for sustainable and optimal packaging solutions, this study proposes a novel two-stage algorithm for the multi-container three-dimensional bin packing problem. The research addresses this problem within the context of a real-world industrial scenario and implements several practical constraints including: full shipment, customer positioning requirements, and product geometric interlocking, for increased stability and with the purpose of minimizing the use of plastic wrapping and/or additional supporting surfaces. The main optimization target is to minimize the total number of containers used in the palletization process of custom orders with varying degrees of complexity. The proposed algorithm includes two stages/phases of processing. In the first phase, the algorithm uses constructive heuristics to generate homogeneous product layers. The layers are then stacked to produce blocks, which are then placed on individual containers or pallets. The second phase packs the leftover items using a genetic algorithm. The performance of the proposed solution is benchmarked using real-world industrial data, as well as a more classic academic benchmark. It is demonstrated, across a very large set of orders, that the algorithm always achieves solutions for full palletization of the orders. The analysis shows that the approach is generic and the quality of the solutions generated is relatively even for both small and large, homogeneous and heterogeneous problem instances.

INDEX TERMS Multi-container 3D-PP, 3D bin packing problem, multi-heuristic algorithm, industrial application, real world constraints.

I. INTRODUCTION

The growing trend in product customization affects not only final products' features but also their transportation and logistics. Product customization is a reality across many different industrial and commercial domains. A key challenge, with considerable economic impact, is to optimally pack multiple items with different weights and form factors on pallets or in containers. Optimal packing can follow different criteria, depending on the sector. However, maximizing the utilization of space while fulfilling an arbitrary number of auxiliary constraints is the main application line. The problem is generally known as the three-dimensional packing problem (3D-PP) [1]. A very important variation of the problem is the multi-container three-dimensional bin packing problem, which this paper directly addresses. Within such variation, a number of items need to be fully packed into the least number of containers. The problem can be easily solved if the items to be packed have the same physical properties (including but not limited to: dimensions, load capacity, shape, etc.). In reality, however, there are many applications where high numbers of extremely heterogeneous items must be packed together.

Currently, palletization and container filling operations for such heterogeneous cases are predominately done by humans, sometimes with the support of machinery for

The associate editor coordinating the review of this manuscript and approving it for publication was Claudio Zunino.

the displacement of heavy items. There are economical, efficiency and even safety aspects that are at stake when manual operation is considered. First, the process needs to be learned and often requires training, which results in training costs and creates a ramp-up period for operators. The impact of training cannot be discounted because many industries will use temporary workers in such processes, who have not necessarily been trained before in the specifics of a particular system. After the ramp-up period, the efficiency is still biased by the operators' capacity to evaluate an order and devise an adequate palletizing sequence. Finally, an operator cannot easily provide guarantees about the stability of the container being packed, which is a safety concern in non-walled open containers such as pallets. Sub-optimal geometric arrangements may also result in an excess of containers being used, which incurs higher transportation costs [2].

3D-PP algorithms can be used to help human operators create more optimal arrangements while providing guarantees in a multi-objective, multi-constrained 3D-PP. Such algorithms can also be used for fully automated palletization. Here, the human aspect becomes important again because manual palletization is a type of work that many operators are not motivated to carry out. It is considered to be a physically intensive, repetitive and unfulfilling task.

Another advantage of an algorithmic approach to the problem is that less obvious arrangements of items can be considered and evaluated by the algorithm. Sustainability aspects can be incorporated into that process. For example, by creating geometric interlocks between the items it may be possible to transport pallets without wrapping their contents in plastic film, substantially reducing plastic consumption. This is in addition to the volume optimization aspects mentioned before.

Generally an algorithmic approach to the problem improves efficiency by reducing human errors, optimizing volume utilization and providing guarantees about key quality aspects of the palletization process. Collectively, the previous dimensions translate to cost reduction through better utilization of resources. Finally, from a sustainability perspective, doing away with plastic wrapping is of high relevance. Moreover, transporting well-packed, compact and stable arrangement of items will also reduce sub-optimal transportation and displacement of goods.

In the literature, 3D-PPs belong to the class of cutting and packing problems, which have been extensively studied by the scientific community. According to [2], 3D-PPs can be categorized (for items of relatively small size compared to the container size) as input minimization or output maximization problems. Input minimization consists of packing all the items in the fewest possible containers, while output maximization consists of attempting to pack as many items as possible in a limited number of containers.

This contribution focuses on the first problem and assumes that, upon receiving an order, a company wishes to ship the products in the least number of containers required

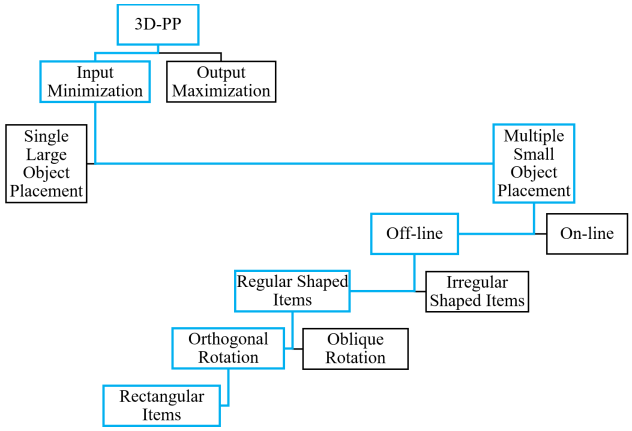


FIGURE 1. Literature categorization of the 3D-PP in respect to its main variables.

(but all the products must be packed). The proposed approach also constrains the problem in the following ways: All the containers are pallets and therefore do not have vertical support walls that a product could be placed against. All the pallets used in the palletization of a single order have the same dimensions. The item's shape is constrained to be cuboid, with variable dimensions and orthogonal rotation allowed along the Z-axis. The previous constraints are widely realistic, as even large e-commerce stakeholders tend to opt for cuboid packaging, sometimes even with further form factor restrictions [3].

A final consideration that is relevant for the definition of the 3D-PP problem is the look-ahead window size with respect to the items being placed in a container or pallet. In some logistic scenarios, it is not possible to know, a priori, all the items that should be packed into a container (they are packed as they come to the packaging area). In other scenarios, the entire composition of the order is known. These two variations of the problem are known as online and offline 3D-PP, respectively. The proposed approach considers the offline 3D-PP problem.

Many alternative formulations of the 3D-PP problem are possible and each formulation will be subject to specific constraints and challenges, and each will have its particular complications and use cases [2], [4]. Figure 1 summarizes the main problem specifications, constraints and applicability limitations/directions of the proposed approach.

After the aforementioned constraints are considered, the overall complexity of the problem, considering real industrial use cases, is still too large for exact approaches and solutions. The general problem of deciding on the optimal palletization sequence grows in a factorial fashion with the number of items to be placed on a pallet. Such complexity can only be tackled by resorting to heuristic methods.

The specialized literature provides a vast amount of potential solutions. However, the authors note that many of the proposed solutions resort to oversimplifications of the problem that make them unviable in a real application scenario. Some of the most common simplifications include

ignoring: the weight of the items, support conditions, pattern complexity, customer item positioning constraints, etc.

There are important practical consequences to ignoring the above. Ignoring the weight of the items often results in solutions where the center of mass of the item aggregate results in unstable pallets, which are hazardous and not acceptable in real-life scenarios. Such solutions could also theoretically allow lighter items to be overly compressed or even deformed by heavier items. Ignoring support conditions leads often to arrangements that are impossible (e.g., an item is placed in simulation, but in reality, there is not enough supporting area on the item to guarantee that the item would not fall or deform). Pattern complexity is another practical constraint of great importance. A fully automated solution can rely on complex positioning patterns because, at least theoretically, the mechanical part of the system is equipment with appropriate sensors and actuators to deliver such accurate positioning. However, in reality, many companies rely on both human-based palletization and a form of automated process, sometimes even in collaboration. Algorithms suitable for the latter case require that the devised positioning provide visual cues to the operator so that the desired item alignment and placement can be achieved. Finally, companies very often require products of the same kind/brand/form factor to be packed together. Many retailers will place the pallet directly on their shop fronts, so there are aspects of item reachability, presentation consistency and branding to be considered in the palletization process.

In light of the previous discussion, the authors propose a multi-heuristic-based solution for the palletization problem that addresses many of the practical constraints mentioned. Particularly, our proposed solutions focus, without loss of generality, on the Food and Beverages (F&B) domain, where the authors have several ongoing R&D efforts. For testing and evaluation purposes, the work considers a real dataset (called Dataset 1000), with roughly 1000 data points that have been previously characterized and made available in [5]. Each data point corresponds to one anonymized customer order.

The results show that the proposed algorithm can palletize a wide assortment of real-world orders in an industry acceptable way considering practical constraints. The authors do, however, recognize some relevant and practical limitations, particularly those related to the execution time of the proposed approach in very large and strongly heterogeneous orders.

Considering the above, the subsequent details are organized as follows: Section II provides a comprehensive overview of the state of the art in 3D-PPs and offers a comparative analysis of the discussed literature. Section III provides a brief introduction to the characteristics of the datasets used in this study, drawing the main differences between real industry data and more academic datasets; Section IV provides a technical description of the proposed solution; Section V discusses the results obtained using the proposed solution on the aforementioned datasets; finally, Section VI offers the final thoughts, lessons learned and immediate development directions that result from this study.

II. LITERATURE REVIEW

As discussed in the introductory section, many variations of the 3D-PP are possible. There are as many variations of the problem as there are possible approaches to it. The latter are generally influenced by two main factors: the size of the problem and the amount of prior knowledge about it (online or offline problem). Both of these aspects require a bit more characterization to unveil their impact on potential solutions.

The dimension of the problem is directly related to the number of items to be palletized/packed. However, the sheer number of items does not tell the whole story. Heterogeneity of form factors combined with a relatively modest number of items will usually yield a problem of higher complexity (as shown in Section V). Highly uniform orders can be quickly palletized by resorting to geometric patterns. Real use cases (see [5] and section III) will denote a very ample variation in the number of items, item types and form factors. For sufficiently heterogeneous orders, size immediately limits exact solutions. These limitations are purely of a computational nature, the number of possible solutions becomes quickly computationally intractable.

A priori knowledge also heavily conditions the approaches. Exact solutions are impossible for online problems because in real-world online scenarios algorithms have access to at most the next 2 or 3 items to be palletized. Not surprisingly, approaches using learning are therefore more frequent for tackling online problems in comparison to offline problems.

For the general case, heuristic or hybrid solutions are the only reasonable strategy. Only limited guarantees of optimal solutions can be offered under the scope of such solutions, and even the sub-optimal solutions will be non-identical depending on the constraints that the algorithm respects.

In the literature, there exists a myriad of methods for solving 3D bin packing problems (see Tables 1, 2, 3, 4, which provide an overview of the cases analyzed in this section).

With the previous in mind, this section analyses both online and offline approaches and attempts to put them into context, given the application scenarios considered in the different contributions. Because of the many possible formulations of the 3D-PP, a direct comparison between approaches is virtually impossible. Different approaches will use different quality evaluation metrics. Computational execution performance metrics are also not comparable as they are sensitive to the quality of the software and hardware used to execute the different algorithms. The computational efficiency of the algorithms could theoretically be investigated but oftentimes algorithms are not sufficiently described to evaluate that either. The analysis in this section is therefore of a more qualitative nature and starts by defining the many different

practical constraints that are frequently considered in the different solutions.

A. CONSTRAINTS

Constraints are of extreme importance for solving 3D-PPs [6] as they cater to quality and safety aspects in the packing process. Constraints also significantly influence the performance of the 3D-PP algorithms by modifying the space of possible solutions. They are used to filter out invalid solutions and satisfy specific customer requirements in real-world industrial scenarios.

Item shape is the most frequent constraint. The great majority of the approaches consider cuboid items, occasionally allowing other regular solids, for example, cylinders. Packing of highly irregular items is uncommon in theoretical and practical applications for several reasons: it is practically impossible to precisely package irregular geometries while guaranteeing or calculating all the forthcoming packaging constraints. For reference, an example of irregular packaging can be found in [7].

The **weight limit** constraint is often used to ensure the total weight of the packages is within the load-bearing capacity of a container [8], [9], [10], [11], [12], [13]. By satisfying this constraint, proper weight distribution of the packages can be achieved [6].

The center of mass (COM) calculation is frequently used to guarantee the stability of the container, thereby preventing the risk of tipping. In the literature, this is usually referred to as **stability constraint** in the literature, which can be divided into static and dynamic stability. **Static stability** protects the container from tipping over when the container is standing still [8], [9], [10], [12], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34]. **Dynamic stability** (which can be further classified into lateral and longitudinal stability) is used to prevent packages from falling over when the container is subjected to forces during displacement [10], [14], [16], [20], [21], [22], [27], [30], [34], [35], [36].

Item orientation is another practical constraint. In real-world scenarios, items can only be placed in a limited number of orientations that guarantee the integrity of the products during packaging, storage and transportation. Limiting the orientation usually has a secondary beneficial effect on the solution side, which is improving its computational performance by greatly restricting the number of possible packaging configurations that need to be evaluated. Due to this, it is a frequently considered constraint across a wide range of solutions (see Tables 1, 2, 3, 4). Transformations in item orientation of cuboid-shaped items are usually restricted to orthogonal rotations and are relatively common, representing the majority of the cases analyzed in this section. This ensures that the edges of items packed in the container must be parallel to the container's and other items' edges. A few solutions allow oblique rotation ([16], [25]) and some forbid any form of rotation at all [12], [37].

Stacking constraints limit the amount of column weight that particular items can tolerate before enduring deformation or damage. Bortfeldt et al. discussed four factors (i.e., condition of the transportation, content inside the item, geometry of the item and material of the item) that determine the maximum allowable pressure an item can handle before failure [6]. In the literature, the general approach for integrating this constraint is to decide an upper limit on the weight an item can support for a given orientation [20], [21], [22]. However, in many cases, the load limits are simply not known and, even in real-world applications, the general rule of not placing heavier items on top of lighter or fragile ones, is usually used as an approximation to the stacking constraint [11], [14], [27], [29], [38], [39].

Support constraints guarantee that items in a container, or pallet, are appropriately supported and balanced [8], [40], [41], [42]. Support is critical for the safety of operators during loading and unloading of the items. Different support criteria may be considered: requiring contact with specific points and edges in the item, requiring a certain percentage of the bottom face of the item to make contact with other items, a combination of both, etc. Within the verification of support constraints, it is sometimes included that items may not protrude outside of a specific volume (usually determined by the area of the container in connection to a maximum admitted packing height). The majority of the examined works seek to fulfill support constraints, with the exception of [43], [44], and [45] where such constraints were not considered.

Non-collision constraints guarantee that items are placed such that they do not collide with each other. All the works examined in this section guarantee non-colliding packing.

Other constraints - aside from the constraints examined before, which are extremely frequent in the literature, some authors introduce more niche or practical constraints. Examples of such constraints include, but are not limited to:

- Complete shipment - all the items in the order must be palletized/packed [16], [46].
- Suitability for robotized and/or operator-based packing - the generated packing solution must be executable by both machines and humans, which often implies creating packing arrangement that include visual cues for operators on where to place the items (overlapping item corners or edges, color coding, etc.) [47], [48].
- No buffering - items cannot be buffered before packing, which usually applies to online problems. Generally, buffering is relevant in both online and offline approaches, but the majority of offline approaches studied assume all the items are available at all times. This is rarely the case in real-world applications where a limited amount of items pertaining to the whole order is always buffered for packaging [48], [49].

B. OFFLINE ALGORITHMS

A considerable amount of the packing algorithms surveyed focus on off-line problems (Tables 1, 2, 3). Solutions to

offline problems can be loosely categorized into exact (Table 1), heuristic (Table 2) and machine learning-based (Table 3).

1) EXACT ALGORITHMS

Exact algorithms (Table 1) perform an exhaustive search within the entire solution space and are guaranteed to find an optimal solution. The application of exact solutions is limited to problems of reduced complexity. As mentioned before, reduced complexity usually means an order with low heterogeneity and a relatively modest number of items and practical constraints. These limitations are currently imposed by the lack of sufficient computational power that would enable an exact algorithm to directly tackle a real order without further heuristics. While the direct and single application of such algorithms, at the time of writing, could be considered inadequate [2], they can be combined with other heuristic approaches to improve their performance [14], [19].

Common techniques used in exact solutions are based on mixed-integer linear programming (MILP) [50] which can largely be seen as the basis for other exact methods ([8], [12], [40], [41]) including combinatorial branch-and-bound algorithms [42].

Junqueira et al. showed that MILP can incorporate practical constraints (i.e., load bearing, dynamic stability) and solve medium-size problem instances (up to 100 items) [12]. The best exact algorithms can, at the time of writing, solve problem instances with a maximum of 180 items, considering load stability and complete shipment constraints, using the dual bound approach [16].

Ocloo et al. developed a MILP formulation for solving a single large object placement problem (SLOPP). The model is capable of satisfying numerous practical constraints like fragility, COM distribution, loading priorities, etc. [19]. Due to the extreme computational load, their model produced optimal solutions for small-size problems up to 16 boxes. Nascimento et al, further proposed an exact algorithm using integer linear programming (ILP) to solve the single container loading problem [14]. Their algorithm approached 3D-PP with 12 practical constraints and solved medium-size problem instances (maximum 110 items).

2) HEURISTIC ALGORITHMS

Due to the complexity of real-world 3D-PP problems, heuristic algorithms, sometimes hybridized with exact approaches, are practically the main solution. The majority of heuristics considered attempts to reduce or simplify the vast solution space by making assumptions about potentially working/non-working solutions, by solving multiple sub-optimization problems and aggregating the results, exploiting specific problem characteristics, etc. A common line pursued in the works examined includes using a so-called constructive approach comprising three steps:

- Loading Pattern Generation - defines item aggregates of some kind (sets of the box in blocks or layers) and/or defines the allowed volumes where such aggregates may be placed.
- Packing Sequence Generation - determines the order and position of the item aggregates satisfying an arbitrary number of constraints.
- Solution Evaluation and Iteration - evaluates the quality of one solution, possibly combining different solutions with each other and re-iterating the algorithm until a certain set of quality parameters' values are met.

In order to generate loading patterns for the items a placement strategy needs to be developed. In the literature, placement strategies are classified into the following major types: horizontal layer building [11], [12], [13], [21], [26], [29], [30], [31], [35], [36], [39], [51], [52], [53], block building [7], [18], [20], [23], [32], [34], [53], [54], stack building [22], guillotine cutting [8], wall building [33], [55], [56] and cuboid building [14]. Different loading patterns will manage the usable packing volume in slightly different ways (prioritizing a certain packing orientation and direction over others, restricting the placement to specific areas, enforcing certain contact points between the packed items, etc.). Additionally, loading strategies aim at simplifying the solution space by creating larger item aggregates with geometries that facilitate packing. The general idea is to create a set of aggregates according to one of the loading patterns and subsequently optimize the placement of these aggregates.

Layer building refers to a loading pattern where items are placed horizontally or vertically (wall building) layer-by-layer. This loading pattern first places items on the floor/container base to create the first layer. The subsequent layers are placed on top of, or next to, the previous layer until no more layers can fit inside the container. There are many different strategies for creating layers regardless of their placement. Layers can be constructed using identical items only [26], [29], [36], [39], [52], [53]. Other authors allow items of different types to be included in the same layer, their process consists of first creating a simple layer of same-type objects and then merging these layers vertically together resulting in a heterogeneous layer [13]. Another possible strategy includes creating layers of items of the same height, not necessarily of the same type, and distributing them to ensure that the COM of a layer is close to the geometric center of the container [31]. Load-bearing limit and item weight have been used as criteria for creating heterogeneous layers in [21]. In a different problem type, the customer provides the construction design of layers and the authors used a heuristic approach to stack the layers [30], [35]. Some variations of the problem consider multi-compartment containers. In this case, the algorithm considers the geometric dimensions of all the compartments in order to generate the layers. A layer can be composed of either identical items or residual items only [22]. When items of different types are combined, the packing density of the layers is usually considered a metric

TABLE 1. Analysis of offline 3D-PP literature based on exact approaches and improvement strategies.

References	Constraints	Optimization objectives	Max item packed	Loading Pattern	Improvement Strategy
[8]	Stability, orientation, load bearing, non-collision	maximize total profit, minimize number of containers	53	Cutting stock	MILP
[17]	Oblique rotation, load stability, complete shipment constraint, non-collision	Maximize the number of packed boxes, minimize the container number	180	Cuboid building	Dual bound
[12]	Stability, load bearing, non-collision, no rotation	Maximize volume utilization, minimize number of container and loading time	100	Layer building	MILP
[41]	Non-collision, orientation	Maximize volume utilization	7	-	MILP
[42]	Non-collision	Minimize number of containers	-	-	MILP
[43]	Non-collision	Maximize volume utilization	90	-	Combinatorial branch and bound
[20]	Fragility, COM distribution, loading priorities, non-collision	Maximize volume utilization	16	-	MILP
[15]	12 practical constraints, i.e., complete shipment, manual loading etc.	Minimize height of the container, maximize volume utilization	110	Cuboid building	ILP

of layer quality [51]. Another strategy for wall building is reported in [55] where the problem is modeled as a single knapsack problem where strips of boxes need to be formed and grouped into vertical layers.

Stack building is another strategy for simplifying the 3D-PP whereby items are arranged in vertical columns (stacks) first, and then the stacks are placed on the container floor by solving a more computationally tractable 2D-PP problem [11].

Another way to solve the 3D-PPs is to use the ''block building'' approach. This approach creates cuboid-shaped blocks of items and then loads them into the container. A block may be homogeneous (items of the same type) or heterogeneous (different types). An example of homogeneous block creation can be found in [32] where a probabilistic method is used to reduce the size of the blocks and optimize their fitting into the available volume in the container. Heterogeneous blocking is reported in [34] and [54] where tree search is used to load the blocks into the container's available volume. A hybrid strategy is reported in [23] where blocks are first created using the work described in [34] and then loaded into the container using an exact method. A similar approach is described in [18], however, instead of using an exact method for the placement of the blocks, a meta-heuristic framework using simulated annealing is considered. An approach with close affinity to block building consists of a three-step process whereby similar items are optimally packed into cartons, and cartons are then packed into crates, which are subsequently loaded into a container using a multi-search heuristic algorithm [7].

The sequence generation and solution evaluation parts of the constructive heuristic approaches tend to have a very high impact on the performance of the different algorithms. Different approaches are possible and fall into two main categories: static or dynamic sequences [57]. Static sequences are defined before the packing process is attempted by the algorithm, and they do not change [56]. Dynamic sequences are flexible and can be adjusted during the process based on the problem's state [20], [36], [53].

3) MACHINE LEARNING ALGORITHMS

Heuristic-based and hybrid approaches improve upon exact approaches by allowing a much larger portion of the solution space to be explored. This, however, comes at a computational performance cost. Recently, machine learning (ML)-based approaches have started to emerge as a potential alternative to heuristic approaches in some selected use cases. The general idea is to induce neural classifiers using existing examples of successfully pre-palletized/packed pallets/containers. The work with these solutions is widely experimental, and there is an important general distinction to be considered between exact and heuristic approaches and ML-based approaches. The first can guarantee that all the considered constraints are fulfilled for all devised solutions, while the latter can not directly provide such guarantees. On the other hand, the advantage of ML-based approaches is that a potentially feasible solution can be obtained in a fraction of the time it takes to compute an exact or heuristic-based solution.

Examples of ML approaches include [60] where Deep Reinforcement Learning (DRL) is used in connection with a heuristic method to pack items in a space while minimizing surface area using real data. The DRL method optimizes the order of items, and the heuristic chooses the best position and orientation for each item. In [44] the authors claim to have improved the algorithm discussed in [60] by using a multi-task selected learning approach, which combines reinforcement learning and supervised learning, to decide the order and orientation of items at the same time. In [17] another neural optimization model based on reinforcement learning is discussed to find an optimal sequence/order and orientation of block selections. The approach assumes a set of pre-blocked items and concentrates on devising the packing sequence only. In [61] a deep neural network is used to

TABLE 2. Analysis of offline 3D-PP literature based on heuristic approaches and improvement strategies.

References	Constraints	Optimization objectives	Max item packed	Loading Pattern	Improvement Strategy
[9]	Load bearing, stability, rotation, non-collision	Waste space minimization	200	-	Weight first best fit
[10]	Load bearing, stability, orientation, non-collision	minimize waste space, number of container	80	-	Relax-and-Fix, Insert-and-Fix, Fractional Relax-and-Fix
[11]	Stacking, load bearing, non-collision	Container volume minimization	200	Layer building	Two-dimensional heuristic decomposition, best-fit, ruin-and-recreate
[27]	Stability, COM, orientation, cargo-related, non-collision	Maximize volume utilization	100	-	Multi-population biased random key genetic algorithm
[29]	COM, non-collision	Maximize volume utilization, minimize packing time	15	-	Adaptive particle swarm optimization (PSO)
[31]	COM, stability, non-collision	Minimize number of trucks and waste space	1993	Layer building	Greedy randomized adaptive search procedure (GRASP)
[36]	COM, dynamic stability, non-collision	Minimize number of containers	142	Layer Building	Integer linear programming
[32]	COM, stability, non-collision	Maximize number of packed boxes	289	Layer building	Constructive heuristic
[26]	Oblique rotation, non-collision, COM	Maximize volume utilization, minimize number of containers	200	-	Column generation-based heuristic
[30]	Stacking, orientation, COM, non-collision	Minimize number of containers	2597	Layer building	GRASP
[59]	Load bearing, orientation, cost, non-collision	Minimize shipping cost	50	-	Combinatorial formulation
[60]	Orientation, non-collision	Minimize shipping cost	125	-	Tabu search, guided local search
[21]	Stacking, positioning, stability, non-collision, orientation	Minimize number of residual items, maximize volume utilization and stability	88	Block building	Constructive greedy-type placement heuristic
[22]	Load bearing, COM, customer sequence, stacking, non-collision	Minimize number of pallets, cost and overlap area	769	Layer building	Layer-based column generation approach
[23]	Load bearing, COM, orientation, stability, non-collision	Minimize number of residuals, minimize COM deviation	2078	Stack building	hybrid heuristic MILP
[28]	Fragility, orientation, load bearing, COM, non-collision	Minimization of waste space	100	-	Constructive heuristic
[47]	Complete shipment constraint, non-collision	Maximize volume utilization, minimize incomplete order shipment number	100	Normal pattern	Variable neighborhood search (VNS)
[34]	Non-collision, orientation, stability	Maximize volume utilization, minimize height of the container	700	Wall building	Tree-search
[40]	Non-collision, orientation, layer building, stacking, item separation	Minimize number of containers, weight of the containers and volume of the containers	-	Layer building	Constructive layer building
[53]	Non-collision, Loading bearing	Maximize volume utilization	100	Layer building	Genetic algorithm
[13]	Maximum weight of a pallet, non-collision	Minimize the number of pallets, minimize weight difference between pallets	100,000	Layer building	Layer building approach
[52]	Non-collision, layer construction constraints	Minimize number of bins	2000	Layer and column building	Branch-and-price scheme
[33]	Non-collision, stability	Minimize number of pallet, maximize volume utilization, minimize packing time, height of the container, number of residual and deviation in COM	50	Block building	Multi-start random constructive heuristic
[55]	Non-collision	Maximize volume utilization	100	Block building	Tree search
[35]	Non-collision, orientation, stability	Maximize volume utilization	100	Block building	Greedy-lookahead algorithm
[24]	Orientation, stability, non-collision	Maximize volume utilization	100	Block building	Genetic algorithm
[19]	Loading priority, customer positioning, COM, pallet maximum weight, stability, non-collision, orientation	Maximize volume utilization	100	Block building	Simulated annealing
[7]	Complete shipping, non-collision, orientation, customer positioning	Maximize volume utilization	-	Block building	3-stage heuristic
[56]	Orientation, support, guillotine cut, non-collision	Maximize volume utilization, minimize container envelope	100	Wall building	Tree search
[57]	Non-collision	Maximize volume utilization	750	Wall building	General heuristic
[54]	Orientation, support, non-collision	Maximize volume utilization	100	Layer building, block building	Multi-layer search
[37]	Orientation, non-collision, cargo stability	Maximize volume utilization and cargo stability	100	Layer building	GRASP

TABLE 3. Analysis of offline 3D-PP literature based on machine learning approaches and improvement strategies.

References	Constraints	Optimization objectives	Max item packed	Loading Pattern	Improvement Strategy
[61]	Orientation, non-collision	Minimize pallet surface area, maximize number of items packed	12	-	DRL
[62]	Non-collision, orientation	Minimize pallet surface area and cost, maximize ranked reward	10	-	Ranked reward algorithm
[45]	Non-collision, orientation	Minimize pallet surface area	12	-	Reinforcement learning (RL)
[18]	Non-collision, orientation, stability	Maximize packing efficiency and stability	10	-	RL

TABLE 4. Analysis of online 3D-PP literature based on their solution approaches and improvement strategies.

Algorithm	References	Constraints	Optimization objectives	Max item packed	Loading Pattern	Improvement Strategy
Heuristic	[49]	Blocking prevention, buffering prevention, non-collision, orientation	Minimize number of containers	1000	-	Deepest-Bottom-Left and Best-Match-First Packing
	[46]	Non-collision	Maximize volume utilization, minimize container height	151	-	Deepest-Bottom-Left
	[25]	Non-collision, stability	Maximize volume utilization	10	-	Height map-Minimization
ML	[50]	Buffering prevention, non-collision	Maximize volume utilization, minimize number of containers	505	-	Deep reinforcement learning (DRL)
	[48]	Robot-implement ability, non-collision, load bearing	Competitiveness ratio	370	-	DRL
	[44]	Non-collision, orientation, no buffering	Minimize total number of pallets and waste space	-	-	Double Q-learning
	[16]	Non-collision, stability, no buffering	Maximize volume utilization	505	-	DRL

estimate a placement policy and a Monte Carlo tree search is used to improve said policy. The previous approaches are, at large, aligned with the tendency to use hyper-heuristics where problem-specific heuristics are automatically created or selected [62], [63].

C. ONLINE ALGORITHMS

Solutions for the online 3D-PP (Table 4) are much more contemporary than the solutions for the offline variations of the problem. Online problems also have a closer affinity to recent automated bin packing problems, made possible by the latest advances in artificial vision systems, which will not be revised in this paper. Exact optimization solutions are not possible for the online problem since there is no optimization scope, every item needs to be immediately packed and therefore global optimal guarantees are not possible.

1) HEURISTIC ALGORITHMS

Online 3D-PPs are different from offline 3D-PPs, the majority of the search-based methods for the latter are not suitable for the former. Therefore, several other heuristic methods have been developed to solve online 3D-PPs. One of the most popular and simple heuristics is the Deep-Bottom-Left (DBL) heuristic, which was introduced in [45] and [48]. This heuristic tries to place the item as deep, bottom and left as possible in the container.

In [24] the Height Map Minimization method is proposed, which aims to minimize the increase in volume occupied by the items counting from the loading direction [24].

Heuristic approaches for the online problem tend to perform poorly from an optimization point of view and can at best guarantee that some limited constraints are fulfilled. In very heterogeneous orders, such heuristics would either create an excess of containers or violate certain constraints (e.g., column load, stability, etc.) when compared to offline heuristics.

2) MACHINE LEARNING ALGORITHMS

The limitations of heuristic approaches in the online variation of the problem render ML-based learning very interesting. ML has the advantage of using previous knowledge to estimate which local decision generally tends to work better for the general case (the algorithm applies a kind of knowledge-based intuition for deciding on the placement of the item).

A DRL-based framework to solve an online 2D-PP that can also work for online 3D-PPs is discussed in [43]. The framework takes an image of the bin as input and outputs

a pixel location where the next item should go. The reward function of the algorithm tries to pack the next item next to the already packed items to maximize the free space for the remaining items. A two-step approach is discussed in [47]. In the first step, the number of possible positions and orientations is reduced by only considering the ones that are likely to be optimal, such as the corners of the container or the edges of the packed items. In the second step, DRL is used to learn good packing strategies and select one position-orientation combination from the candidates.

The online 3D-PP can also be modeled using a constrained Markov decision process and solved utilizing the DRL approach [15]. The authors of that work have used the actor-critic framework to improve the DRL strategy, which combines value function learning and policy search and optimizes the policy based on action feasibility predictions. They used a feasibility predictor to estimate if the placement actions are feasible and adjust the action probabilities of output by the actor during training.

D. SUMMARY AND ANALYSIS

The literature review clearly shows the diversity of problem variations and possible solutions. It also shows that there are concrete scalability limits with respect to the maximum problem size that can be reasonably tackled by the majority of the approaches. Exact techniques peak at 180 items, while heuristic approaches have been tested up to much higher values. All approaches have shown that they are somehow efficient in optimizing towards the most common optimization goal: improving area/volume utilization. The number and quality of the constraints considered, as well as the ability to handle very heterogeneous orders, become the differentiating aspects. Complete shipment constraints are not always considered, which biases the obtained results. Additionally, many of the approaches trying to pack a large number of items are also packing very homogeneous items, which indirectly benefits the solution. The authors' experience and observations from evaluating the literature suggest that the most performance-taxing part of all algorithms is the one that handles the heterogeneity of a specific order. Heavily heterogeneous orders generally defeat all the layer-building and block-building approaches, which rely on the uniformity of item dimensions to reduce the problem/solution space.

ML-based approaches are emerging as candidates for solving many of the performance issues of heuristics and exact solutions in the many variations of the 3D-PP. However, ML solutions require well-curated data, which can only come from two sources: companies' records of previously packed items, which usually do not exist, or artificially generated solutions created by heuristics and exact solutions. One general problem is that realistic datasets for the evaluation of 3D-PP solutions are not available. Previous research carried out by the authors shows that a great majority of the specialized literature refers to 19 different datasets [5] out of which only [4] is publicly available. The orders in the said dataset, compared to the orders' profiles of real companies, are relatively limited in both size and item variety. Some of the works analyzed also claim to have done testing on real-world datasets. However, these datasets are not made public most likely because they contain sensitive data. It is therefore not clear how many of the reviewed approaches would perform with real-world data. This creates additional difficulties in comparing the merits of the different approaches. In addition to that, to the best of the author's knowledge, there are no publicly available realistic datasets for training ML solutions, which complicates future developments.

III. A REALISTIC DATASET FOR EVALUATING 3D-PP SOLUTIONS

The previous section clearly shows the diversity of problem variations and possible solutions. It also shows that the lack of realistic testing data sometimes undermines the correct evaluation of existing and new approaches in real-world scenarios. To mitigate this effect the authors test their proposed approach against the dataset described in [4], to enable a comparison with previous works using that dataset. More importantly, the authors test their approach against the dataset detailed in [5] which is publicly available and based on anonymized real industry data. The distribution of data points in the two datasets is depicted in Figures 2 and 3. Each point in the figure represents an order in the corresponding datasets as a function of the number of items in each order, the number of different products in them and an entropy measurement of the order. The entropy of the orders was calculated in the following way [5]: the dataset was grouped by order and product, and the quantity of each product was summed for each group. The Shannon index was then calculated and normalized for each order. The function *entropy* from the *python scipy. stats._entropy* package was used for calculating the index. Normalization of values between 0 and 1 was carried out by subtracting the minimum entropy value from the calculated index and then dividing by the difference between the maximum and minimum entropy values. This results in a value where the higher the entropy, the higher the uncertainty in the distribution of the number of items per product type.

Comparing both datasets side by side shows that the variation of order composition in Figure 3 is much higher and more unpredictable than in Figure 2. The main differences are in volume of items, product variety but also distribution of product variety. These variations pose additional performance challenges to most algorithms, particularly for high volume high entropy orders, where the authors note the very high number of orders with more than 400 items that simultaneously exceed 20 varieties of products. These numbers are the upper complexity limits of the dataset detailed in [4] and visible in Figure 2. Furthermore, it is important to mention that, the dataset from 2, used with some frequency

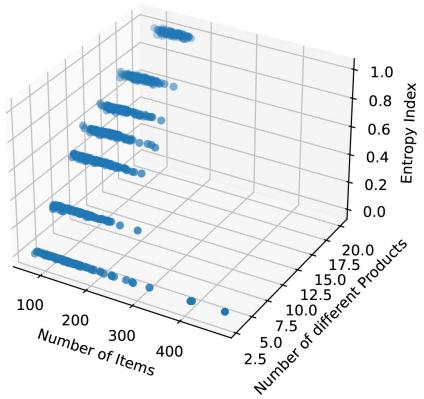


FIGURE 2. Distribution of orders in the dataset from [4].

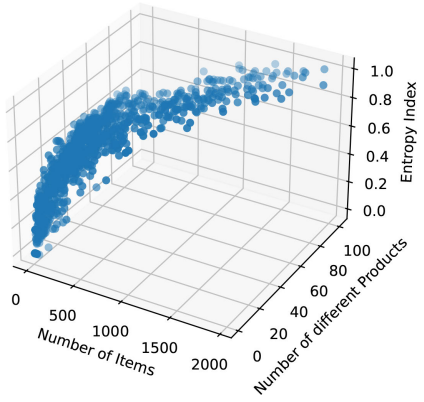


FIGURE 3. Distribution of orders in the dataset from [5].

in the literature, does not include information regarding the weight of the items. The latter is a limitation because item weight largely influences the palletizing patterns in real-world scenarios.

IV. PROPOSED SOLUTION

3D-PP is a well-known combinatorial optimization problem and is also well-known to be NP-Hard. The algorithm described in this section focuses on the off-line multi-container 3D bin packing variation of the problem and considers several practically relevant industrial constraints. These are frequently considered in other algorithms described in the literature but very rarely combined into a single algorithm.

A. PROBLEM DESCRIPTION

In the context of warehouse logistics, and in a very simplified way, customers place purchasing orders with a variety of product compositions that need to be palletized and shipped using containers/pallets. Each order consists of I set of items, of cuboid shape, and each item ($i \in I$) is represented as $[l_i,\ w_i,\ h_i,\ m_i,\ v_i,\ q_i]$, that is length, width, height, weight, volume, and quantity, respectively. These items then need to be palletized using fixed-size containers/pallets. The

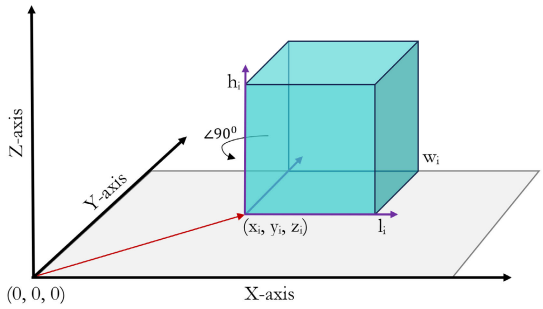


FIGURE 4. Frame of reference for placing items inside a pallet.

pallets are defined as a function of their dimensions as in $[L, W, H]$, representing their length, width and height. In the present case, but without loss of generality, the Euro pallet format [1200 mm, 800 mm, 1400 mm] is considered. The problem is treated as unbounded to satisfy the full-shipment constraint. This means that the algorithm has the capability of adding additional pallets to the problem in order to achieve a solution.

The palletization process, pallet dimensions, item representation and placement will always occur, by convention, in the first quadrant of the XY plane. The origin of the 3D coordinate system considered in the palletization process is therefore, also by convention, placed at the left-bottom corner of the pallet. The x-axis points along the length of the pallet and the y-axis points along the width of the pallet.

The position of an item i is defined by one point in the above-mentioned coordinate system and is represented as a tuple of integer values $[x_i,\ y_i,\ z_i]$. Such a point is also the origin of the coordinate system of the item, which has the same default orientation as the coordinate system previously defined for the pallet (see Figure 4).

The total quantity of items of type i is denoted as q_i, and hence the total number of items in an order is $\sum_{i=1}^{m} q_i$.

The proposed algorithm therefore finds one packing sequence as well as the positions of the items within that sequence for all the items in one order. The proposed solution attempts to minimize the number of pallets used, maximize the compactness (i.e., maximize the contact area between all the faces of items) of the packed items, and minimize the heterogeneity of item types in each pallet. It does so by combining several heuristics in a two-phased process detailed later.

While doing so, the solution simultaneously satisfies eight different constraints: **item orientation, non-collision, stability, support, pattern complexity, complete shipment, customer positioning, layer interlocking**.

According to the literature review in Section II-A and to the best of the authors' knowledge, existing well-documented methods do not simultaneously address all the above-mentioned eight constraints. The authors cannot exclude, however, the hypothesis that there may be commercial solutions that address them. However, one

should stress that the selection of the 8 constraints, detailed above, for inclusion in the proposed solution resulted from a requirements elicitation exercise coupled with an industrial research project, whereby companies mentioned that they had not found an immediate solution that would cover the specifics of their palletization problems.

Of the constraints selected, the first six are hard constraints. A solution will not be considered valid if one of those constraints is violated. The latter two are soft constraints. The algorithm tries to fulfill them; however, it will not reject infringing solutions to achieve fulfillment of such constraints. The majority of the constraints described next have been presented before. Therefore, in this section, only the problem specific details and interpretation of said constraints are discussed.

Constraint 1: The **item orientation** constraint is used to restrict orthogonal rotation to the z-axis. This is a common constraint in many sectors but also something very characteristic of the F&B domain, where this work is grounded. Many food products require a specific side of the package to be facing up and disallow the products from being placed on their sides or upside-down. The current algorithm, therefore, only allows two orientations. The default orientation of the box and its 90 degree rotation along the box's z-axis. The default orientation of the item must guarantee that its edges are parallel to the edges of the pallet so that the permitted rotation will also fulfill the same requirement.

Constraint 2: **Non-collision** guarantees that all the items in the pallet do not occupy overlapping volume envelopes. The algorithm uses axis-aligned bounding boxes to support the detection of collisions. Overlapping bounding boxes are therefore interpreted as colliding.

Constraint 3: The **stability** constraint is evaluated for each pallet in the solution. To that end, the COM of the pallet with the packed items is calculated. The position of the COM is restricted to an area contained on the XY plane and surrounding the geometric center of the pallet.

Tipping conditions are subsequently checked. If the packed pallet passes the tipping test, it is then checked for relevant offsets of its COM on the XY plane relative to the geometric center of the pallet in the same plane. The amount of accepted deviation is configurable. Stability and support constraints (defined next) directly address safety requirements and regulations.

Constraint 4: **Support** constraints are satisfied if the following conditions are true:

- at least 40% of the base of an item and its 4 base vertices are supported by other items, or by the pallet or,
- a minimum of 50% of the base area is supported as well as 3 base vertices or,
- 75% or more of the base area of the item and 2 base vertices are supported.

The algorithm does not actually test the real vertices of the cuboid but rather a set of four points, corresponding to the XY translation of the cuboid base vertices inwards and toward the center of the base. The value of the XY inward translation is configurable. Support constraints are important from a practical perspective. Not all items can be supported in the same way. The proposed solution uses, therefore, a combination of supported corner points with the percentage of an item's bottom area that is supported. These two previous re-configurable constraints cover the following cases:

- items that bend under axial weight - in this case, requiring corner support is not enough, a minimum amount of bottom area must be supported, in addition to a specific number of corners, to adequately support such items;
- items with a very rigid bottom section - in this case, requiring the support of 3 or 4 corners may be enough for as long a small base area is also supported;
- any variations between the two cases described above may be adequately configured.

Constraint 5: The **customer pattern complexity** constraint is a customer-specific constraint. Different algorithms will create different packing patterns. In the context of this work, it is important that the algorithm generate solutions that can be palletized in manual, semi-automated and fully automated modes. The item placement strategy considered ensures that any placed item has at least one vertex and two edges aligned with either the pallet or some other item adjacent to it. The previous provides easy visual cues for manual packing in a real-world industrial scenario. This is opposed to fully automated packing strategies, which may place items in positions that are only possible with precise mechatronic placement. However, it is practically impossible for human operators (e.g., place an item aligning one of its edges with the line on the second fifth of the length of another item). The customer pattern complexity constraint is important to guarantee that packing solutions can be efficiently and safely implemented by human operators.

Constraint 6: The **complete shipment** constraint requires all the items in an order to be packed. Many academic contributions disregard this constraint, particularly when solving single-container problems. In real-world applications, shipment is performed using the least number of containers required but the entire order needs to be shipped.

Constraint 7: **Customer positioning** constraints are very frequent in palletizing operations. In this case, the constraint specifies that, as much as possible, equal items must be packed as close as possible, ideally creating complete layers of products. The rationale behind such a constraint is that, in the context studied, the majority of the pallets are shipped to retailers that wish to de-palletize a shipment directly to their shelves. In that context, products should not be mixed. For example, in a real-world scenario, a retailer may want to de-palletize several layers of milk packages directly onto its shelves without having to sort out other items in a geometrically complicated mix. It is worth mentioning, however, that complex orders do not always palletize into regular arrangements of layers of the same product. In this

case, it is not possible to always fulfil such a constraint, and a few leftovers of certain products will be mixed, in a less organized way, with other products. When this occurs, the algorithm prioritizes the minimization of the number of pallets used over the hard fulfillment of this constraint.

Constraint 8: The **layer interlocking** constraint can be seen as another customer constraint, and it is used to ensure interlocking among the items between two layers stacked on top of each other. This constraint is of high practical relevance since well-interlocked items increase item stability under pallet displacement. Therefore, packing companies can dispense the plastic wrapping of the items during shipment. Reduced plastic usage is desirable due to sustainability aspects.

B. PROPOSED ALGORITHM

The proposed algorithm operates in two distinct phases. In the first phase, constructive heuristics are used to generate layers and blocks of items. The second phase deals with the residual items that cannot be incorporated into layers and blocks; and it uses a genetic algorithm to place them on top of existing blocks or empty pallets. The algorithm therefore assumes a certain repetition of items of the same kind in a certain order (this is corroborated by the data discussed in Section III). A fully heterogeneous order, where each item is of a different kind and a different form factor, invalidates the usage of the first phase, and in those cases, the algorithm proceeds with only using the second phase to generate a solution. In the forthcoming text, an overview of the algorithm is presented, after which the full details of the two phases are provided and discussed.

1) ALGORITHM OVERVIEW

An overview of the algorithm can be found in Figure 5. The algorithm starts by evaluating a complete order and deciding if both phases can be enabled. If so, items are merged into full, half and quarter layers, depending on their quantity and dimensions. The dimensions of the layers are determined by the area of the pallet. A full layer has the same area as the pallet, and a half layer has half of the length of the full pallet and the same width, finally, a quarter layer has half of the length and width of a full layer. Initially, the algorithm will only form layers of the same item type.

Quarter layers of different products but of the same height are subsequently combined into half layers. Half layers of the same height are then combined into full layers. At this point, there may be layers of just one item type but also layers combining up to four item types. The latter happens in the rare cases where four quarter layers were first merged into two half layers, which were subsequently merged into one full layer. Such layers are of course guaranteed to have the same height.

When no more layers can be merged, they are merged into blocks. At this point, there may be residual items (i.e., items that did not make into layers), these are dealt with at a later stage. Blocks are created by combining layers in a way that

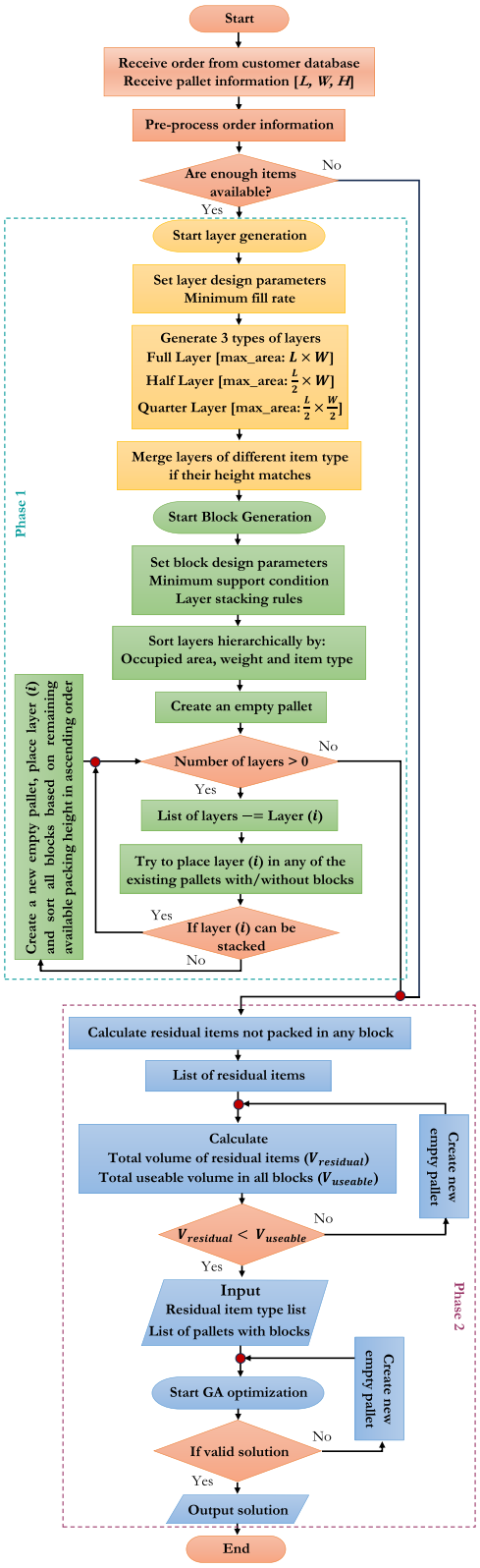


FIGURE 5. Overview of the proposed algorithm.

fulfils the constraints earlier specified. The general approach is as follows:

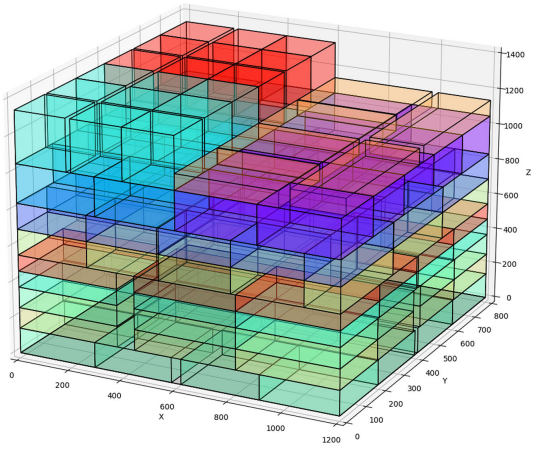


FIGURE 6. Generation of blocks using multiple full layers, half layers and quarter layers.

The algorithm starts from an empty pallet and sorts all the full layers hierarchically by: occupied area, weight and item type. The previous results in layers with the best area coverage, higher weight and with only items of the same type to be on the top of the list of layers. These are placed first and on top of each other. If a placement violates one of the hard constraints earlier specified, or if the maximum packing height is exceeded, a new pallet is created and the layer is placed there. From this point on, the algorithm has more than one pallet available where it can continue to develop the blocks. Before it places any additional layers, the existing blocks are sorted by remaining available packing height in ascending order, and the process repeats. This maximizes the use of the available volume per pallet.

When all the full layers are developed into blocks. The algorithm proceeds with merging half-layers into the existing blocks. The block developing rules are the same and, additionally, the algorithm will balance the height of the blocks by ensuring that a half layer that does not violate any hard constraints is preferably placed in the lowest half of a block.

Finally, the process repeats for quarter layers. The only difference is that, in this final case, the algorithm will balance the height of the block's four quadrants, instead of halves. An example of this block creation procedure can be found in Figure 6 where cuboids of different colours denote items of different types in the same order.

At this stage, all the layers have been developed into blocks and occupy one or more pallets.

The algorithm now proceeds to handle the residual items. These are preferably packed into the existing blocks. The algorithm will therefore pre-check if the remaining usable packing volume on the existing blocks is theoretically sufficient to accommodate the total volume of the residuals. If the check fails, the algorithm spans a new pallet. In either case, the algorithm now starts phase 2 and a Genetic Algorithm (GA) is used to determine the packing sequence and the position of the residual items on the available blocks and pallets. The same constraints specified earlier need to be fulfilled. If the GA is not able to find a solution, a new pallet is spawned and the GA is re-ran. The choice of a GA, among other alternative heuristics approaches, is due to the ease and efficiency of GAs in allowing and processing potential solutions that use permutation encoding. This is the case in the present problem, where the authors wish to find the best sequence for placing items in a pallet.

2) PHASE 1: LAYER AND BLOCK BUILDING

The first phase of the algorithm consists of constructing as many layers as possible and developing them into blocks. The main advantage of pursuing such a constructive approach is that the packing solution for the bulk of a large volume order can be very quickly devised.

At the same time, it becomes very easy to incorporate all the constraints of interest detailed before while directly addressing Constraints 1, 5 and 8.

There are many different ways of creating layers. Constraint 1 limits the orientation of the items but still many packing patterns are possible. Here, the approach leans on simplicity while trying to accommodate Constraint 5. Items with different form factors will fill in a layer differently and use more or less layer area. In the ideal case, the dimensions of an item are such that in a certain quantity the items completely fill the whole area of the layer. In such idealized cases the layer also offers full support for the next layer. However, the previous is hardly ever the case. Therefore, for every item type eligible to form layers, the algorithm:

1) tests the placement of the items in one orientation;
2) evaluates the unused area and;
3) attempts to place therein additional items in the second orientation.

After testing both starting orientations, the algorithm selects the pattern that packs more items into the layer. If both alternatives allow the same number of items in the pallet, the algorithm selects the one that has more items in the default orientation. An example of a layer constructed following this procedure can be seen in Figure 7.

Figure 7 shows two possible layering patterns for the same set of items. All the items are of the same type and the colors are only used to facilitate distinguishing between items. Both layers in the figure satisfy the constraints and pack the same amount of items, so they can be both used. In this case, the algorithm will prefer whatever solution has more items in the default orientation. The reason for using the default orientation is a practical one. Packages are usually designed such that the main aesthetic elements are depicted along the most visible faces within the default orientation of the item. For a layer to be accepted by the algorithm, the items on it must cover a minimum amount of the available base area (fill rate). Such value is a configurable input for the algorithm. The authors have found empirically that 85%, 90% and 90% of minimum fill rate values for quarter, half and full layers respectively, tend to produce the best results.

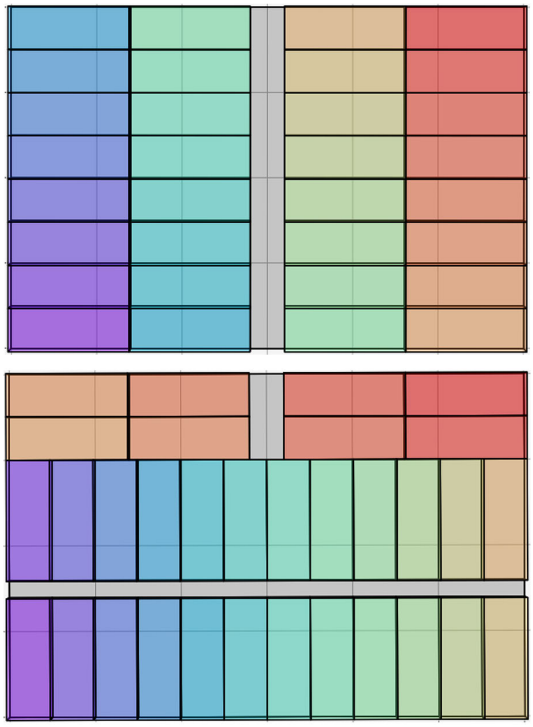


FIGURE 7. Two alternative configurations of the same layer (i.e., the same number of homogeneous items) based on the dynamic shifting approach.

Figure 7 shows another important behavior of the algorithm. After selecting the best layer pattern, the remaining unused space is distributed along the center lines of the pallet. This has the effect of pushing the items to the extremities of the pallet. After studying and evaluating many layering patterns, the authors have empirically found that this simple placing heuristic improves: the stability of the pallet, support for subsequent layers and layer interlocking prospects (Constraint 8).

The creation of layers is computationally very efficient because it does not need to verify for support and eliminates item collisions by design. At the same time, it utilizes the preferred item orientation and fulfills the constraint 5.

Building blocks using layers is therefore a matter of testing for pallet stability and item support between the layers, as well as improving layer interlock and fulfilling the customer positioning constraint.

The heuristic considered for improving the interlocking between layers is based on the calculation of the Hausdorff distance between the top vertices of the items in the bottom layer and the bottom vertices of the items in the top layer. The Hausdorff distance is higher, the higher the distance between the vertices on the bottom layer and the top layer. In the present case, the distance reflects the misalignment of the vertices of the items. The more misaligned, the higher the distance, and the better the interlocking between the layers.

To maximize the Hausforff distance, the algorithm considers 4 different variations of one layer: the original pattern, the pattern with horizontal symmetry, the pattern with vertical symmetry and the pattern with both horizontal and vertical symmetry. The version with a higher distance is selected. Note that this does not change the layer composition. The heuristic is therefore more efficient when the layer is least symmetric. Figures 8(a) and 8(b) show the interlock between two identical layers seen from the top, and the interlock between multiple heterogeneous layers seen from the side of a block, respectively.

This procedure applies to all layer types (full, half and quarter). As mentioned before, block construction prioritizes the placement of the larger full layers first, followed by half and quarter layers.

The first half layer is always placed on top of the first half of the last full layer on the block or on the first half of the pallet (if that half layer is the first layer in the block). From that point on, the subsequent layers are always placed on the lowest half of the block.

A similar procedure is considered for the quarter layers with the exception that the first quarter layer is always placed in the lowest quadrant of the block. The only exception is when the quarter layer starts on an empty pallet and is effectively the first layer in the block. In that case, it is always placed in the first quadrant. For reference, the block's halves and quadrants are depicted in Figure 9.

At the end of the block building procedure, the bulk of the order has been already packed into blocks (see Figure 10 where a large heterogeneous order has been organized into blocks), and what is left is to handle residual items that do not conform to layers or blocks (residuals).

3) PHASE 2: PACKING RESIDUALS

Residual packing represents the second phase of the algorithm. At this stage, the algorithm must pack a usually very heterogeneous set of items with respect to quantities and form factors that do not conform to layers and that do not easily fit together. Two problems need to be solved at this stage: decide where to place the items and evaluate different placing sequences. This is indeed the part of the algorithm where complexity substantially impacts its computational performance and solution quality.

The creation, evaluation and improvement of the quality of a specific placing sequence are done by a GA. The decision of where to place an item follows a specific placement strategy. The dynamic between the two is as follows: The GA generates potential placing sequences (i.e., ordered sequences in which the residuals are to be placed), and the placement strategy takes such sequences and generates a placement that does not violate any of the constraints.

Complexity arises on several fronts. First, for an arbitrary quantity of items q, there are $q!$ possible placing sequences (e.g., an order with 10 items has 10! or 3,628,800 possible placing sequences). The factorial growth of the number of theoretically possible palletizing solutions as a function of the number of items is the biggest contributor to the complexity of the 3D-PP problem. Secondly, for any placing sequence,

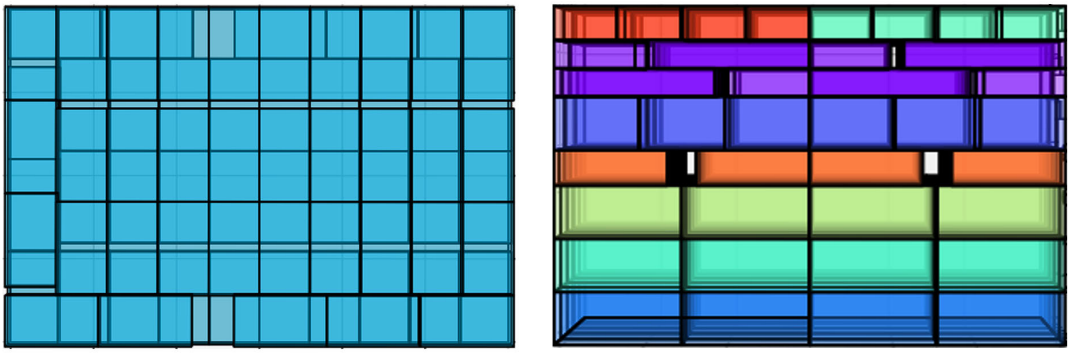


FIGURE 8. Interlocking of layers during block building displayed as (a) top view between two identical layers and (b) side view for multiple heterogeneous layers.

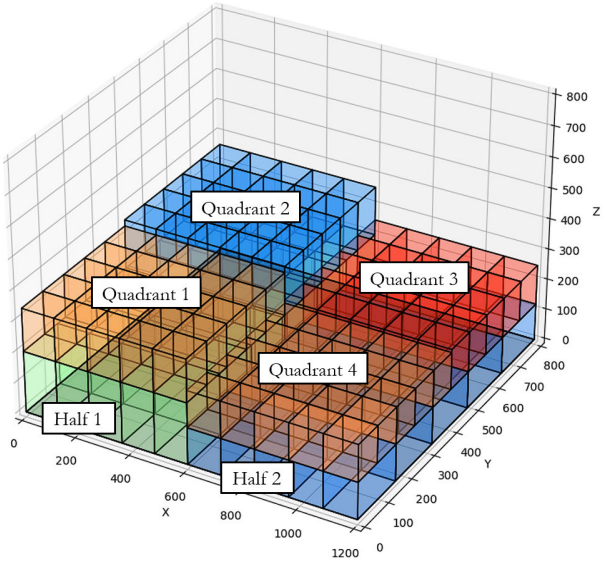


FIGURE 9. Visualization of block's halves and quadrants.

the placement strategy itself generates additional complexity. An item can potentially be placed in any available space on the available blocks.

The second phase of the algorithm reduces this complexity as follows. Instead of generating/encoding placing sequences for individual items, the GA's individuals encode the order in which items of a certain type are to be placed (Figure 12 shows how the problem is encoded in the GA's individuals as well as how the mating and mutation evolutionary operators are applied). This has two effects:

1) it partially addresses Constraint 7 which stipulates that items of the same type should ideally be packed together (and thereby by encoding item types rather than individual items implicitly presorts them by type before sending them to the placement strategy);
2) significantly reduces the search space when the residuals contain items of the same type (e.g., if a set of residuals would be composed of 50 items from 25 different types distributed such that every two items were of the same type, the search space would reduce from 50! to 25!).

TABLE 5. List of 10 custom individuals added to the randomly generated population pool during GA initialization.

Individual Index	The residual item ids are sorted based on
Individual 0	decreasing order of item weight
Individual 1	increasing order of item weight
Individual 2	decreasing order of item quantity
Individual 3	increasing order of item quantity
Individual 4	decreasing order of item base surface area
Individual 5	increasing order of item base surface area
Individual 6	decreasing order of item volume
Individual 7	increasing order of item volume
Individual 8	decreasing order of item (volume $\times$ quantity)
Individual 9	increasing order of item (volume $\times$ quantity)

However, 25! possible placing sequences is still a very large number and the fraction of this solution space that the GA could explore with reasonable performance is very small.

Proper initialization of the GA becomes very important. The algorithm therefore includes a pre-selected set of individuals in its first generation. These individuals are detailed in Table 5 and codify placement sequences in increasing and decreasing order of properties that are relevant for the palletization process including: weight, quantity, base surface area, volume and volume vs. quantity.

These individuals have shown to enable the GA to converge quicker to a satisfactory solution by conditioning the search space exploration around useful starting points that codify many of the manual placement strategies in place today in industry.

Evaluation of individuals within the GA is subject to two fitness functions. These functions are formulated as follows:

- *Fitness function 1*: Minimize the average item type heterogeneity on all the pallets used to pack the order.
- *Fitness function 2*: Maximize the average compactness of all the residual items placed on top of existing blocks and empty pallets for the current problem instance.

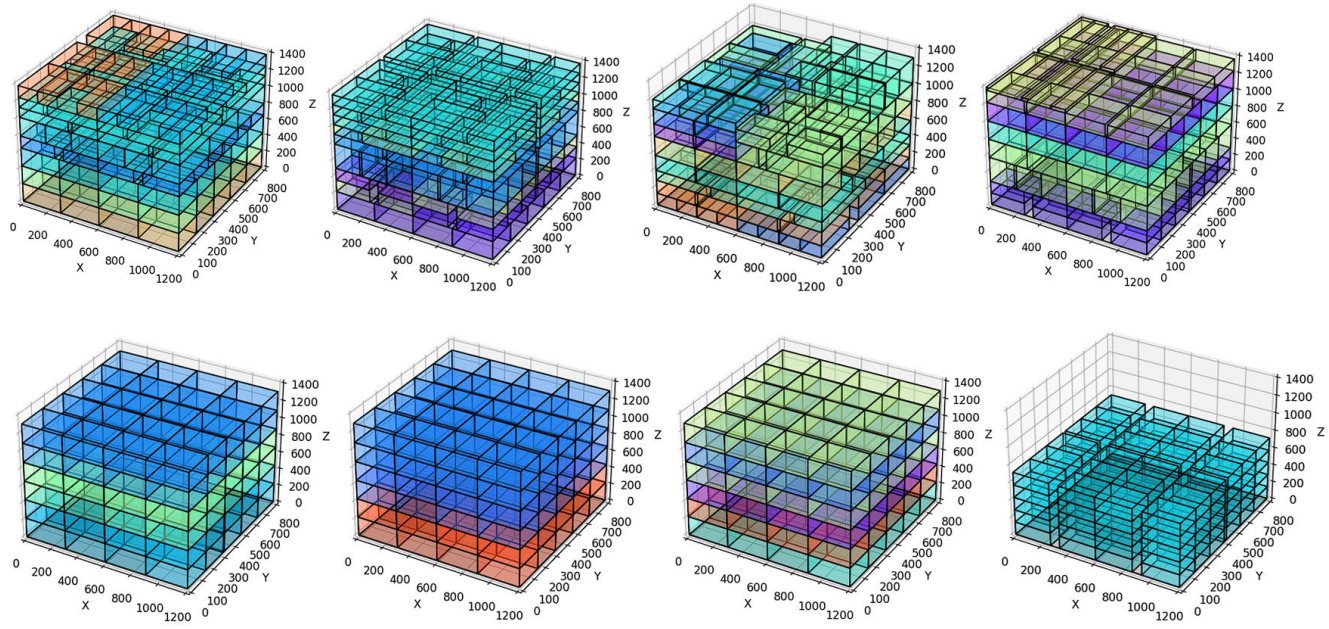


FIGURE 10. Phase-1 block generation for a large order.

Fitness function 1 considers the item types of the items pre-blocked into the pallets and therefore encourages solutions that are more compatible with Constraint 7.

Fitness function 2 promotes tighter packing of the residual items. Compactness evaluates how tightly items are packed on a pallet. It is a measurement of the maximum surface area of an item that is in contact with other items or the pallet. The compactness metric is normalized between 0 and 1. If an item has a compactness value of 1 that means all 6 surfaces of the cuboid item are in full contact with other items in the pallet, and 0 when vice versa. Fitness function 2 evaluates the compactness of only the residual items since the rest of the items are already packed compactly using the constructive heuristics described earlier.

Both fitness functions are therefore directly influenced by the placement strategy used to pack the residuals. The GA must apply the placement strategy to every placement sequence in order to evaluate the results.

The GA uses a modified definition of the Extreme Points (EPs) placement strategy documented in [64].

The general idea of extreme points is that when an item i with dimensions $[l_i, w_i, h_i]$ is placed on a point $[x_i, y_i, z_i]$, it creates additional points (called EPs) where new items can be placed. Generically, if an item is positioned at $[x_i, y_i, z_i]$ on a block or pallet, the following three new EPs are made available for subsequent placements $[l_i + x_i, y_i, z_i]$, $[x_i, w_i + y_i, z_i]$ and $[x_i, y_i, h_i + z_i]$.

In the present case, at the beginning of phase 2, the EPs corresponding to the vertices of the top surfaces of the items in all blocks are immediately made available to the placing strategy. Then, after placing each residual, three EPs per placed residual are created as shown in Figure 11. Each EP may only support one placement. The list of available

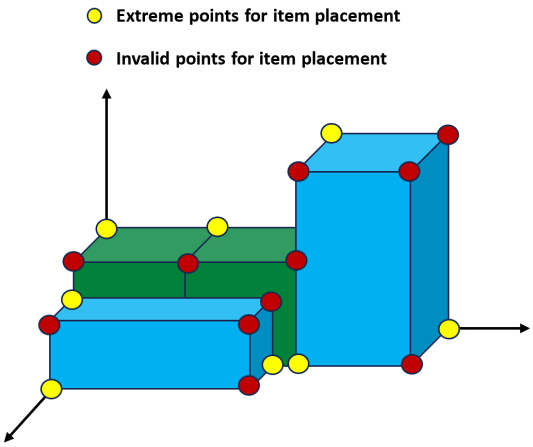


FIGURE 11. Visualization of the extreme points generated by placement of items inside a pallet.

EPs is therefore updated after each placement and does not include duplicated points. EPs are ranked based on their height. Lower EPs score higher to discourage the GA from building columns of items. If two EPs are at the same height then the one closer to the origin of the pallet is ranked first.

For every placing sequence the placement strategy will:

- attempt to place an item in the highest scoring EP and check if the placing violates any of the hard constraints, if it does not, the list of EPs is updated and the next item is placed on the new highest ranking EP;
- if the placement fails, the algorithm will attempt to place the item in the secondary orientation;
- if the placement on the highest ranking EP fails regardless of the item orientation, the second highest

ranked EP is selected and tested, and so on and so forth, until no more EPs are left.

If the placement strategy runs out of valid EPs, the individual corresponding to that placing sequence is given maximum penalizing scores on both fitness functions. If the GA is not able to find a valid solution, i.e., all individuals are unviable, after a certain number of generations, the GA fails and restarts with an additional empty pallet added to the list of available block/pallets.

Failing and restarting the GA is strictly necessary to fulfill Constraint6. Simultaneously, the placing strategy ensures that Constraint 5 is respected. Furthermore, in all cases, the items' positions determined by the placing strategy, during the execution of the GA, are always encoded into an individual as metadata. Each individual in the GA has therefore an active part that participates in the optimization process and a passive part that stores additional information about the placing solution that the individual represents (position and orientation) as well as relevant information about the item types it must place (namely:type, quantity, dimensions and height). The previous is important because, as a multi-objective optimization problem, the result of the evolutionary runs is a Pareto front of non-dominated solutions representing different trade-offs. Currently, the algorithm will prioritize, from the Pareto front, solutions benefiting compactness over heterogeneity.

By the end of the second phase, the algorithm is guaranteed to have found a solution.

C. IMPLEMENTATION AND PARAMETRIZATION OF THE GA

The base GA considered in the algorithm described before uses the DEAP Library [65] implementation of the Mu plus Lambda evolutionary strategy with the NSGA-II [66] selection operator. The Mu plus Lambda algorithm carries out selection from both the existing population and the offspring. While other evolutionary and selection strategies are possible, the authors have found, by experience, that the above combination delivers good quality results. The quality of the solutions obtained is in fact much more influenced by the placement strategy rather than the evolutionary and selection strategies due to the extremely large solution space of the problem.

Nevertheless, tuning the evolutionary strategy parameters will still contribute to better quality solutions. In the present case, the following values are used:

- mu = 15, mu is the number of individuals to select for the next generation;
- lambda = 30, lambda is the number of individuals on the offspring at each generation;
- crossover probability = 0.5, is the probability that an individual in the offspring is produced by crossover;
- mutation probability = 0.2, is the probability that an individual in the offspring is produced by mutation;
- population size = 100, the initial size of the population;

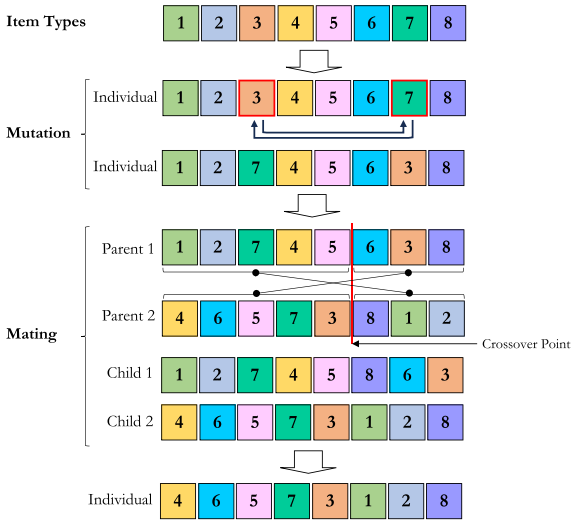


FIGURE 12. GA individuals encoding, mate and mutate operations.

- max stagnation = 5, the base implementation from DEAP was modified to include an early stoppage condition if evolutionary stagnation occurs, in this case, if the fitness values of the solutions are not improving after max stagnation consecutive rounds, the algorithm is stopped;
- ngen = 30, is the maximum number of evolutionary cycles the algorithm will run if stagnation does not occur.

Stagnation checking is active even for infeasible solutions, as experience has shown that if the GA is not able to find a feasible solution within the first max stagnation evolutionary rounds it probably never will on subsequent rounds.

To explicitly connect the GA implementation to the previously described heuristics and constraints, it is worth mentioning that the evaluation of one individual entails applying the placement strategy to that individual and, after the placement strategy has been executed, the GA will score the individual on solution heterogeneity and compactness. So it is effectively the placing strategy that guarantees the fulfillment of the hard constraints 2 and 6, while the GA's selection strategy favors the softer, customer-based constraints 7 and 8.

Mating (crossover) and mutation operations are executed in the following way (Figure 12). Given the nature of the encoding of the individuals, which requires that genes within an individual cannot repeat (that would mean the same item type being packed twice, which violates the problem specification), mating and mutation operations operate as follows:

- Mutation - a pair of genes is randomly selected from the individual and the genes are swapped.
- Mating - one crossover point is randomly selected, and the genes are exchanged between both individuals. This eventually leads to repeated genes and genes that were not included in the solution. The repeated genes are then

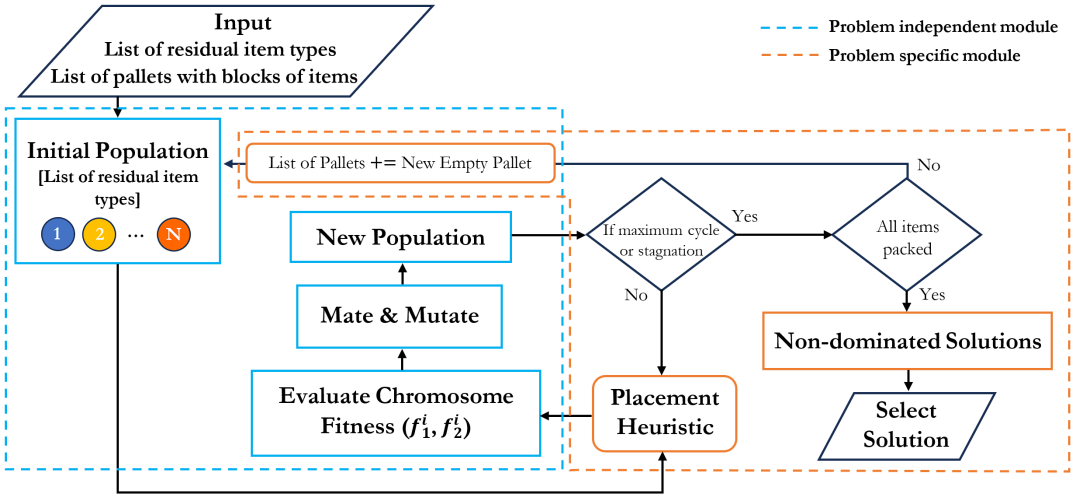


FIGURE 13. Phase-2 GA Architecture.

replaced by the genes that had not been included in the solution.

For reference, the general architecture of the GA is depicted in Figure 13, which shows the delimitation of the standard GA implementation and the custom placement strategy heuristics specific to the 3D-PP developed in the paper.

V. RESULT ANALYSIS AND DISCUSSION

To demonstrate the performance of the proposed solution, the algorithm was benchmarked using the two datasets described in Section III. The palletization of the orders of both datasets was carried out on a workstation with an Intel Xeon W-1250 CPU with a maximum clock speed of 3.30GHz and 16 gigabytes of RAM. The proposed solution was coded using Python V3.12 and executed on a Windows 11 version 22H2 operating system.

A. PERFORMANCE ASSESSMENT CRITERIA

The performance of the algorithm was analyzed using three key metrics: average volume utilization, average compactness and execution time.

The average volume utilization per pallet demonstrates how efficiently the loading space is utilized by the algorithm. The more the available volume in a pallet is filled, the better; after all, companies pay for such volume during transport. This metric can be deceiving in extremely low volume orders, normally used in more academic datasets, whereby the total number of items is not sufficient to fill the volume available in the pallet. Otherwise, for more realistic orders, it is a good estimate of the quality of the solution.

It is worth noting here that the algorithm actually does not optimize for volume utilization but rather for compactness. For realistic orders the bulk of items can be, from a volume utilization perspective, very efficiently packed into blocks. Note that when creating layers, the algorithm requires full and half layers to be at least 90% filled and quarter layers to be at least 85% filled, which results in extremely compact and volume efficient arrangements. The bulk of the order developed into blocks determines, to a very large extent, the number of pallets used. The algorithm is allowed to add more pallets dynamically to fulfill the complete packing of the order and after the first phase, there is usually usable free volume on the existing blocks/pallets. The usage of the GA to optimize compactness ensures that the available volume is explored and that residuals will ideally be packed against existing items from the existing block, preferably of the same kind. This indirectly, but significantly, optimizes volume utilization as well.

On that line, one also evaluates the compactness of the solution with a secondary purpose. Compact arrangements are usually stabler during transport and will do without being wrapped in plastic, with important sustainability implications. Compactness can therefore be used as an indicator of whether or not a packed pallet needs additional support structures.

Finally, execution time creates a kind of make or break deal on the practical utilization of the algorithm. Longer execution times may or may not be acceptable depending on the number of orders that need to be processed per unit of time (seconds, minutes, hours, days, weeks, etc.). Measuring the execution time allows for identifying the use cases that are feasible for the current version of the algorithm.

B. BENCHMARKING WITH INDUSTRY DATA (DATASET 1000)

The algorithm was tested first against the Dataset 1000 detailed in [5] and briefly described in Section III. To help position and characterize the results on such a rich but complex dataset, the dataset was segmented into five classes corresponding to 5 entropy intervals:

- Entropy Interval 1 - [0, 0.2];

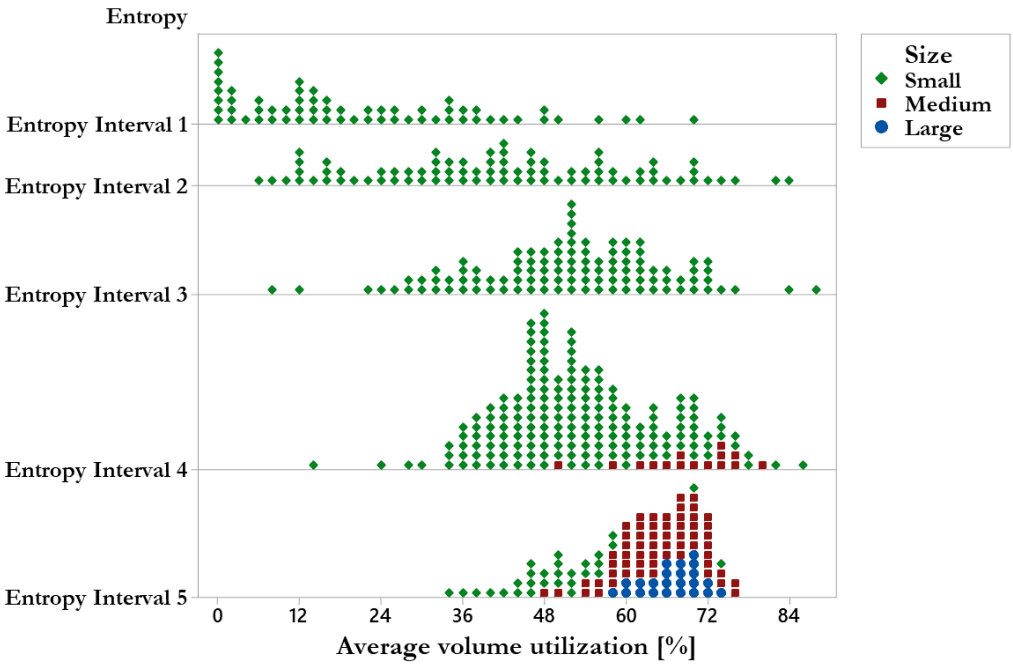


FIGURE 14. Distribution of volume utilization for different order sizes and entropy intervals from [5] (each symbol represents up to 2 observations).

- Entropy Interval 2 -]0.2, 0.4];
- Entropy Interval 3 -]0.4, 0.6];
- Entropy Interval 4 -]0.6, 0.8];
- Entropy Interval 4 -]0.8, 1];

The results of the algorithm on the above-mentioned dataset are summarized in Figures 14, 15 and 16. Collectively, the three figures provide a good insight into the expectable performance of the algorithm on real datasets. Before the results are analysed it is worth recalling that the algorithm is guaranteed to always generate a solution. This is important because many algorithms described in the literature do not provide such guarantees, which renders their practical utilization limited.

One starts the analysis by evaluating Figure 14, which also reveals additional relevant information about the dataset by showing that the order complexity (interpreted as entropy) is not a function of the number of items alone. Figure 14 breakdowns the distribution of orders by order size along the different complexity classes. Small orders have less than 600 items on them, medium orders have between 600 and 1300 items on them, and large orders have more than 1300 items. While medium and large orders predominantly occupy the fifth entropy interval, quite many small orders denote high complexity as well.

Average volume utilization has modest values in low entropy orders because many of these orders have a very small amount of items whose volumes do not fill a single pallet. The same is valid for higher entropy orders with few items on them. In effect, the Dataset 1000 contains a relatively high number of such orders. They are frequent in the F&B industry and therefore worth studying, mainly occurring when products are shipped directly to small businesses or when samples are being exchanged between operational units of the same company.

Complex, medium and large volume orders are usually packed with good performance, with a large portion of such orders utilizing more than 60% of the available pallet volume. Here, it is important to re-stress that these results fulfill the 8 constraints previously detailed. This is relevant because the algorithm enforces extremely stringent support and stability constraints. Many of the algorithms described in the literature, for example, do not test for support or stability and mainly concentrate on creating non-colliding geometric arrangements of the items. Simultaneously, the results reflect what is possible with real form-factor measurements. At this point, one may reflect on the fact that above 60% is acceptable but perhaps not enough for real industrial applications and that in industry, pallets must surely be packed with a higher efficiency today. Both reflections are true but with important caveats. In practical industrial scenarios, additional load carriers are incorporated into the same pallet to even out surfaces and create additional supporting layers. The previous means that a certain volume in the order is actually occupied by the load carrier, which is a non-value adding volume. Load carriers usually take the form of pallets interleaved between item layers. A standard Euro pallet has a maximum usable shipping height normally restricted to 1400 *mm* counting from the top surface of the pallet, with a total usable volume

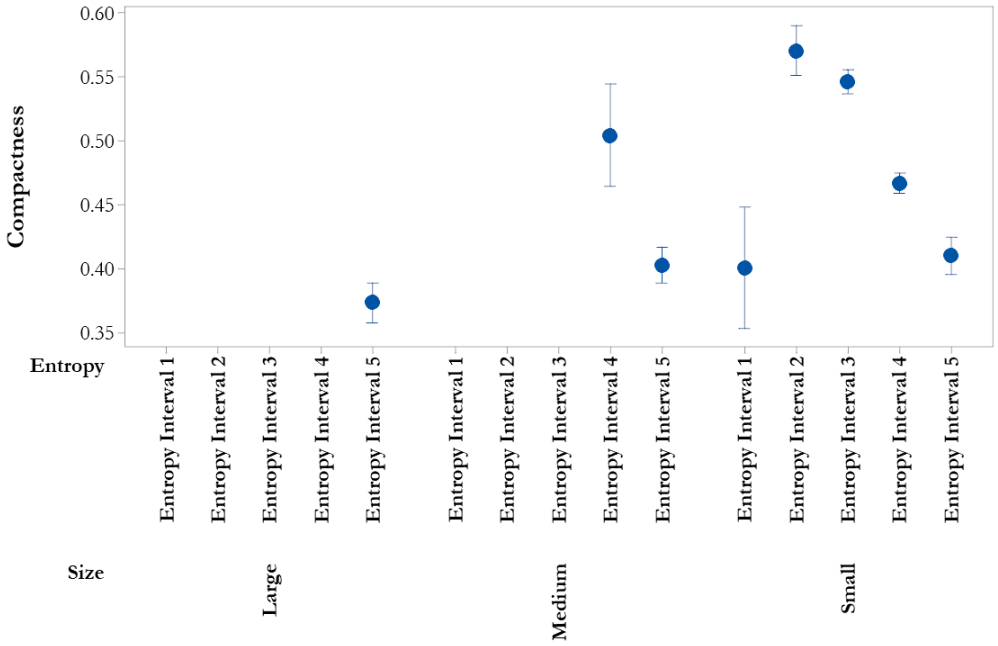


FIGURE 15. Comparison of mean compactness among different problem instances from [5]. Individual standard deviations are used to calculate the intervals.

of 1344 dm^3. The Euro pallet itself is 1200 *mm* by 800 *mm* of base area with a height of 144 *mm*. Its volume is therefore 138.24 dm^3. This means that for every load carrier that is included approximately 10% of volume is lost to the load carrier. The algorithm can be modified to include load carriers and therefore improve its packing efficiency. Conceptually, a load carrier is just an item with the same dimensions as a pallet.

It is now worth inspecting Figure 15.

Figure 15 illustrates that the mean compactness score improves when the number of items within the orders decreases. Overall, the average compactness lies between 35-60%. It is important to stress that compactness is only evaluated for the residuals and does not measure the highly compact and stable arrangements of the blocks created in the first phase of the algorithm. For smaller orders, the algorithm finds more side surfaces to lean residuals against. Some of the real small orders are entirely packed in phase 2, no layers or blocks are possible. In this case, the residuals are tightly packed together. On large orders, the residuals are usually placed on top of existing blocks and there is naturally less side contact surface to support against. A similar effect is seen in small orders.

Finally, it is worth analyzing the execution efficiency of the proposed algorithm (Figure 16).

Execution efficiency is mainly dictated by the nature of the residual items. In fact, the execution of phase one of the algorithm, even for large volume orders, occurs in under 10 seconds. When one inspects, however, the overall execution times, the majority of the values are very far from the first phase execution times. Order entropy and the variety of item types influence the execution time more profoundly than order volume. Higher entropy orders, including items of many different types, usually cause the GA to have to re-start and add new pallets, sometimes as much as four times in more complicated cases. On larger orders, there are many more EPs that need to be tested and the likelihood that the placing strategy fails near completion is higher. In many cases, this means that the placing strategy is able to tentatively place the majority of the residuals only to fail when 2% to 5% of the residuals are left to place. The placement strategy is applied to all the individuals in each generation during the execution of the GA. In these complex scenarios where the strategy fails consistently towards the end, the penalty in execution efficiency is very high with some extremely complex orders taking as much as 17500 seconds to execute. The placement strategy has been identified as the bottleneck in the process due to the many different constraints it needs to satisfy.

To better understand the execution efficiency of the proposed algorithm, a distribution of the number of orders solved in different time intervals is described in Table 6. A large portion of the orders (59.03%) consisting of low to medium complexity can be solved in under 10 minutes. A total of 323 orders with medium and high complexity can be solved in 10-60 minutes. Additionally, high complexity orders (5.52%) are solved within 60-120 minutes. Extremely high complexity orders, which are less frequent (3%), require more than 120 minutes to solve.

The authors believe that the profiling of the order into entropy classes and the evaluation of the algorithm against

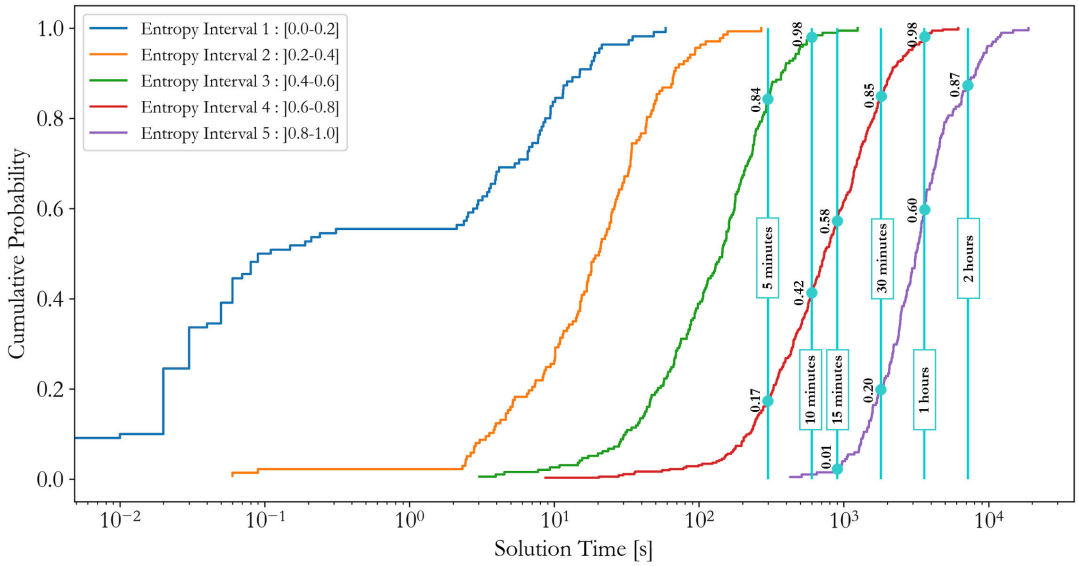


FIGURE 16. Empirical cumulative distribution of computational time for different problem instances from [5].

TABLE 6. Execution time distribution for the 996 orders in the Dataset 1000.

Time interval [minutes]	Number of orders solved
0-5	471
5-10	117
10-15	60
15-30	135
30-60	128
60-120	55
120+	30

them provides an innovative and useful way of benchmarking the 3D-PP algorithms. One of the advantages is that when considering a new dataset, it is possible by comparison with Figure 16 to build a quick idea on the expectable performance of the algorithm in the new dataset or industrial application.

The palletization results obtained for the dataset are available in https://github.com/luferi/3DBPP.

C. BENCHMARKING WITH AN ACADEMIC DATASET (BR1-7 DATASETS)

In this subsection, the performance of the proposed solution is compared against other 3D-PP algorithms that have been proposed in the literature. It is difficult to find performance results and indicators for multi-container problems with a relatively high number of constraints. The BR datasets, described in Section III are single container benchmarks that are commonly used in the literature. As a single container dataset, the majority of the algorithms tested against it evaluate the volume utilization achieved but do not satisfy the complete shipment constraint (Constraint 6). Instead, the problem is usually treated as a variation of the knapsack problem, where the objective is to combine items such that the used pallet volume is maximized. The authors have selected a total of 7 algorithms, published after the year 2000, benchmarked using the BR datasets and following layer/block/wall building approaches:

- HBMLS → Heuristic block-loading algorithm based on multi-layer search [53]
- WGMA → Wall generation meta-heuristic algorithm [18]
- CLTRS → Container loading by tree search algorithm [54]
- VNS → Variable neighborhood search [67]
- BSG-VCS → Beam search algorithm [68]
- GRASP → Greedy random adaptive search [36]
- HGA → Hybrid genetic algorithm [69]

Their performance, with respect to average volume utilization, on the BR datasets is detailed in Table 7. Virtually all these algorithms are capable of using at least 90% of the available packing volume.

TABLE 7. Comparing the performance of different 3D-PP algorithms benchmarked on BR1-7 datasets in terms of percentage of average volume utilization.

Algorithms	BR1	BR2	BR3	BR4	BR5	BR6	BR7
HBMLS	94.92	**95.48**	**95.69**	**95.53**	**95.44**	**95.38**	**95.00**
WGMA	89.24	89.44	90.71	90.44	90.22	88.03	86.67
CLTRS	**95.05**	95.39	95.45	95.18	94.96	94.72	94.26
VNS	94.85	95.10	94.97	94.52	94.19	94.61	93.38
BSG-VCS	94.74	95.38	95.65	95.48	95.33	95.23	94.66
GRASP	89.07	90.43	90.86	90.42	89.67	89.71	88.05
HGA	87.81	89.40	90.48	90.63	90.73	90.72	90.65
Proposed Solution$^{(a)}$	61.92	55.61	52.91	51.45	51.26	51.15	50.96
Proposed Solution$^{(b)}$	49.62	49.05	49.74	49.71	49.50	48.55	46.16

(a) Relaxed support constraint
(b) Strict support constraint

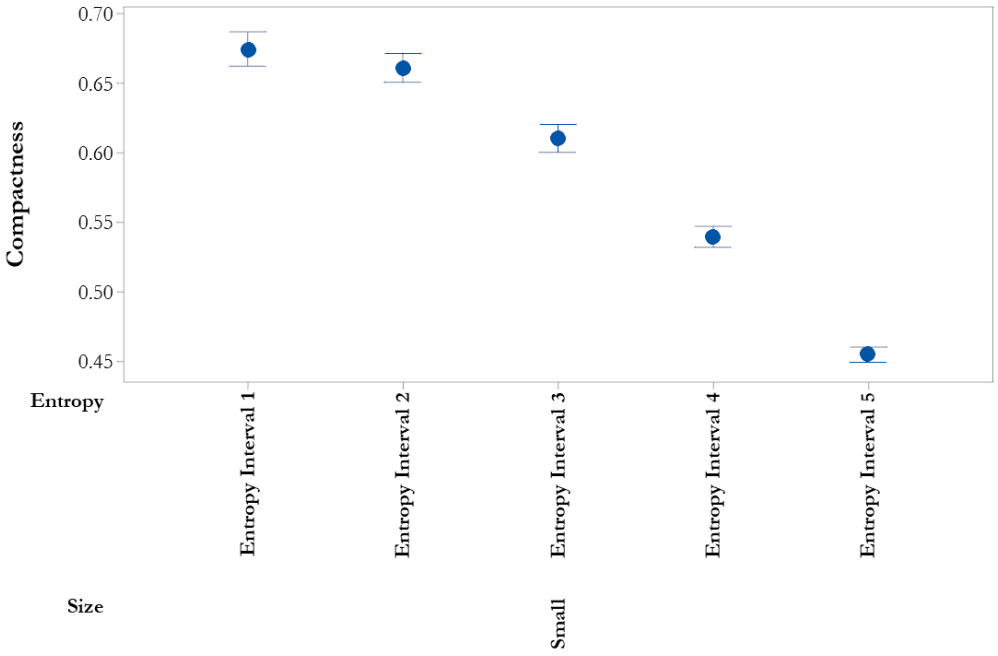


FIGURE 17. Comparison of mean compactness among different problem instances from the BR1-7 datasets. Individual standard deviations are used to calculate the intervals.

Table 7 also shows the results attained by the proposed solution with two different configurations, strict and relaxed support and complete shipment constraints. In the first (strict support) the algorithm applies fully all the constraints specified.

Not surprisingly, the volume utilization results obtained are all near 50% of volume utilization. There is a very clear explanation for that. The proposed solution guarantees that all the items are packed. In the BR dataset, the total number of items is roughly equal to or slightly less than the usable volume in the container. A perfect packing, which is not possible due to a combination of the item's form factor, would yield close to 100% volume utilization. In this context, the current algorithm will, in all cases, use one additional pallet. Mathematically this yields an average volume utilization of about 50% since all the items that theoretically fit in one single container are now fully packed in two containers.

In the relaxed support constraint configuration, the algorithm is setup in a way that closely resembles the objectives and packing behavior of the other algorithms it is compared with. In this case, the support constraints are removed and items are mainly placed taking into account collision free placement and stability criteria. Additionally, for the relaxed case, the numbers reflect that of the most filled pallet, rather than the average of the used volume as in the previous case. One notices some improvement, up to 10% in BR1. However, the algorithm will still consider pallet stability constraints. Because the BR datasets do not include weight information, the authors have assigned a unitary weight to all the packages. In this case, their relative proportion may create imbalances in the center of mass, which leads the algorithm to prefer certain less-than-optimal solutions.

TABLE 8. Comparing the average computing time (in seconds) of 100 problem instances in each of the BR1-7 datasets using different 3D-PP algorithms.

Algorithms	BR1	BR2	BR3	BR4	BR5	BR6	BR7
HBMLS	14.1	34.18	79.43	115.59	155.1	217.88	327.88
VNS	2.98	5.6	11.06	15.12	22.62	31.71	58
BSG-VCS	150	150	150	150	150	150	150
GRASP	8	12	25	28	40	59	64
Proposed Solution	54.55	98.29	183.84	244.3	283.97	362.35	457.45

The results of the proposed solution on the BR datasets are further analyzed in Table 8, Figures 17 and 18. Table 8 compares the average execution time of the proposed solution with the execution time of other algorithms documented in the literature, for the purpose of solving each instance of the BR dataset. As mentioned before, execution time is proportional to the quality of the software implementation and the available computational power. However, it is also greatly affected by the number of constraints and optimization variables that are included in the algorithm. The analysis of Table 8 shows, as expected, that the proposed solution is slower on the selected datasets, in comparison to other previously published results. However, a direct comparison is unfair due to the higher number of constraints, and subsequent improved real-world quality of the palletizing solutions, considered in the proposed solution. One turns, therefore, the attention to the assessment of the absolute value of the numbers. The worst-case scenario for the BR datasets

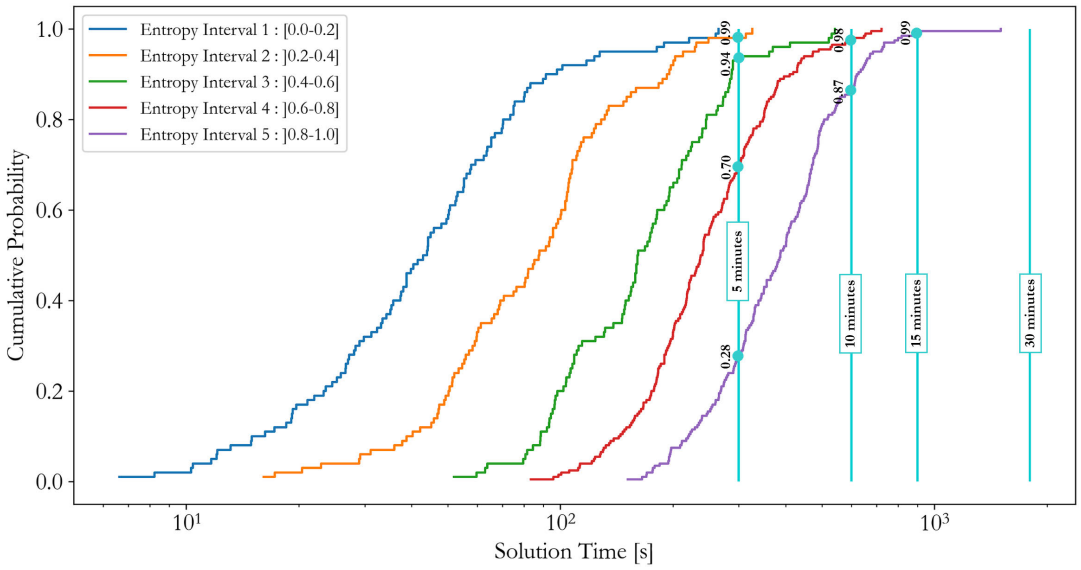


FIGURE 18. Empirical cumulative distribution of computational time for different problem instances from the BR1-7 datasets.

is about 8 mins of execution time, which still allows real-time execution by pre-calculating the orders.

Average execution times portray a limited picture of the problem, without an indication of the distribution of such execution times. In order to better characterize the complexity of the problem instances in the BR datasets a similar approach to Section V-B was used. All 700 problem instances are distributed into 5 entropy intervals. All the orders in this dataset are considered small orders as they are all under 600 items. In the BR dataset, orders are usually much more homogeneous in comparison with the orders in the Dataset 1000. This enables the algorithm to perform the majority of the packing using the first stage. Figure 18 shows that solutions for high entropy orders are found significantly faster than in the case of the high entropy orders in Dataset 1000. Because the second stage of the algorithm has fewer items to pack, this also results in more compact arrangements, as shown in Figure 17.

Figure 17 also shows, however, that the average compactness reduces when the heterogeneity of the problem instances increases from interval 3 to 5. As problem instances get more heterogeneous, the first phase of the algorithm packs a smaller number of items into layers and produces a large number of heterogeneous residuals. Heterogeneous residual items have a high deviation in their contact surface area. This makes it difficult for the GA to pack them compactly.

D. INTEGRATION INTO INDUSTRIAL ENVIRONMENTS

The algorithm takes input data in the format described in [5]. Without repeating the details of this previous contribution, the authors can mention that typical data can be presented to the algorithm from a Comma-Separated Values File (CSV file) or as a stream of strings encoding the same information. Relevant data that the algorithm needs to operate includes: order number, product id, quantity of products of a given id in the order, product dimensions and product weight. As mentioned before, items are assumed to be of cuboid shape. The algorithm outputs the optimized placing sequence including the Cartesian position and orientation of each item in the shipment as detailed in Figure 4.

Industrial integration of the algorithm would require transforming existing order data, typically stored in a company's ERP, into the required data format that the algorithm is designed to use. Then, the data must be transferred to the algorithm for processing. The results of algorithm must then be collected and used in any appropriate way for the system (for example sending the results to a automatic palletizing machine or displaying them in visual interface to guide a human operator in the process). This data transformation and usage pipeline can be automated in a number of different ways. The exact way depends on the Information Technology/Operational Technology infrastructures available at the company. Profiling of the orders into complexity classes will help the end user understand which performance can be expected from the algorithm as a function of the most frequent types of orders. The processing of extremely large and complex orders may have to be carried out ahead of time in order to cater for a higher computational execution time. The algorithm discussed in the paper purposely avoids such implementation details, is generic, and can be parameterized to fit each specific case.

VI. CONCLUSION

3D bin packing problems have a very high impact on any industrial/logistics operation where packing/palletizing and transport operations are included. Packing in a way that maximizes the usage of available volume is the golden

standard. However, packing is a multidimensional problem and the ability to maximize volume utilization is influenced by many factors. In the literature, there are a multitude of solutions for a multitude of well-known variations of 3D-PP. However, many of the discussed solutions only have an academic interest.

Real-world packing scenarios are first conditioned by how much a priori know-how is there about the items that need to be packed. In many scenarios, ordering patterns are quite stochastic and many items are bundled together into larger containers opportunistically (online problem), transported, and subsequently broken down again and distributed (for example, in large online retail operations). However, in many other industrial scenarios, customers will place well-defined orders including a variety of different items. The entire shipment contents are therefore known (offline problem). This contribution focused mainly on the latter. By studying and evaluating order profiles from real industry data, as well as considering a set of general but also customer-specific constraints that occur very frequently in real 3D-PPs, the authors propose and evaluate a potential solution. In particular, the proposed solution addresses two customer-specific needs: packing similar items together and creating packing solutions that are executable in fully automated, semi-automated or fully manual palletization processes.

The 3D-PP solutions generated by the proposed approach have a number of interesting features with a very high practical impact. For once, they provide a wide set of guarantees with respect to the support of the items and the stability of the container. In addition to these guarantees, they rely on a combination of strategies that promote the geometric interlocking of items. The latter has a very high sustainability impact on many occasions and may mean that the pallet can be transported, under certain conditions, without plastic wrapping or other additional supporting and stabilizing aids, potentially reducing the weight on the pallet and doing away with excessive plastic usage.

The proposed algorithm was tested and validated against a very large set of cases and it was demonstrated that solutions of good quality can be generated in a timely way, for the majority of the orders.

A number of improvements in the quality of the solutions are possible, namely in respect to volume utilization for introducing additional load carriers, which is a relatively common practice in the F&B industry. This direction has not been explored in the current version of the algorithm and is the subject of future work. Also in the scope of future work, the authors anticipate a set of relevant developments: the evaluation of additional packing strategies in the second phase of the algorithm but also the use of machine learning based approaches that can consume, in the learning process, the high quality solutions generated by the current approach. In that respect, the proposed approach serves a dual purpose: it can be used on its own, establishing a quality and performance baseline, but it may also be used to generate synthetic data for further AI/ML-based developments. Integration of machine learning techniques to the proposed solution can potentially improve its ability to optimize large-volume strongly heterogeneous order with acceptable performance.

ACKNOWLEDGMENT

The authors would like to thank the invaluable input of the partners of the FLAP project which contributed to the high quality results reported in this manuscript.

REFERENCES

[1] N. Chernov, Y. Stoyan, and T. Romanova, "Mathematical model and efficient algorithms for object packing problem," *Comput. Geometry*, vol. 43, no. 5, pp. 535–553, Jul. 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925772109001576

[2] S. Ali, A. G. Ramos, M. A. Carravilla, and J. F. Oliveira, "On-line three-dimensional packing problems: A review of off-line and on-line solution approaches," *Comput. Ind. Eng.*, vol. 168, Jun. 2022, Art. no. 108122.

[3] Amazon. (2022). *How We Work to Find the Perfect Fit for Your Product's Packaging*. [Online]. Available: https://www.aboutamazon.co.uk/news/sustainability/how-we-work-to-find-the-perfect-fit-for-your-products-packaging

[4] E. E. Bischoff and M. S. W. Ratcliff, "Issues in the development of approaches to container loading," *Omega*, vol. 23, no. 4, pp. 377–390, Aug. 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/030504839500015G

[5] L. Ribeiro and A. A. Ananno, "A software toolbox for realistic dataset generation for testing online and offline 3D bin packing algorithms," *Processes*, vol. 11, no. 7, p. 1909, Jun. 2023. [Online]. Available: https://www.mdpi.com/2227-9717/11/7/1909

[6] A. Bortfeldt and G. Wäscher, "Constraints in container loading—A state-of-the-art review," *Eur. J. Oper. Res.*, vol. 229, no. 1, pp. 1–20, Aug. 2013.

[7] H. Wu, S. C. H. Leung, Y.-W. Si, D. Zhang, and A. Lin, "Three-stage heuristic algorithm for three-dimensional irregular packing problem," *Appl. Math. Model.*, vol. 41, pp. 431–444, Jan. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0307904X16304887

[8] M. T. Alonso, R. Alvarez-Valdes, M. Iori, and F. Parreño, "Mathematical models for multi container loading problems with practical constraints," *Comput. Ind. Eng.*, vol. 127, pp. 722–733, Jan. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835218305527

[9] I. Deplano, C. Lersteau, and T. T. Nguyen, "A mixed-integer linear model for the multiple heterogeneous knapsack problem with realistic container loading constraints and bins' priority," *Int. Trans. Oper. Res.*, vol. 28, no. 6, pp. 3244–3275, Nov. 2021.

[10] C. Paquay, S. Limbourg, M. Schyns, and J. F. Oliveira, "MIP-based constructive heuristics for the three-dimensional bin packing problem with transportation constraints," *Int. J. Prod. Res.*, vol. 56, no. 4, pp. 1581–1592, Feb. 2018. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/00207543.2017.1355577

[11] T. A. M. Toffolo, E. Esprit, T. Wauters, and G. V. Berghe, "A two-dimensional heuristic decomposition approach to a three-dimensional multiple container loading problem," *Eur. J. Oper. Res.*, vol. 257, no. 2, pp. 526–538, Mar. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221716305707

[12] L. Junqueira, R. Morabito, and D. S. Yamashita, "Three-dimensional container loading models with cargo stability and load bearing constraints," *Comput. Oper. Res.*, vol. 39, no. 1, pp. 74–85, Jan. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054810001486

[13] J. Hasan, J. Kaabi, and Y. Harrath, "Multi-objective 3D bin-packing problem," in *Proc. 8th Int. Conf. Model. Simul. Appl. Optim. (ICMSAO)*, Apr. 2019, pp. 1–5.

[14] O. X. D. Nascimento, T. A. de Queiroz, and L. Junqueira, "Practical constraints in the container loading problem: Comprehensive formulations and exact algorithm," *Comput. Oper. Res.*, vol. 128, Apr. 2021, Art. no. 105186. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054820303038

[15] H. Zhao, Q. She, C. Zhu, Y. Yang, and K. Xu, "Online 3D bin packing with constrained deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, May 2021, vol. 35, no. 1, pp. 741–749. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16155

[16] D. V. Kurpel, C. T. Scarpin, J. E. P. Junior, C. M. Schenekemberg, and L. C. Coelho, "The exact solutions of several types of container loading problems," *Eur. J. Oper. Res.*, vol. 284, no. 1, pp. 87–107, Jul. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221719310136

[17] R. Hu, J. Xu, B. Chen, M. Gong, H. Zhang, and H. Huang, "TAP-Net: Transport-and-pack using reinforcement learning," *ACM Trans. Graph.*, vol. 39, no. 6, pp. 1–15, Nov. 2020, doi: 10.1145/3414685.3417796.

[18] E. F. da Silva, A. A. S. Leão, F. M. B. Toledo, and T. Wauters, "A matheuristic framework for the three-dimensional single large object placement problem with practical constraints," *Comput. Oper. Res.*, vol. 124, Dec. 2020, Art. no. 105058. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054820301751

[19] V. E. Ocloo, A. Fügenschuh, and O. M. Pamen, "A new mathematical model for a 3D container packing problem," Brandenburgische Technische Universität Cottbus-Senftenberg, Senftenberg, Germany, Tech. Rep., 2020.

[20] J. Olsson, T. Larsson, and N.-H. Quttineh, "Automating the planning of container loading for atlas copco: Coping with real-life stacking and stability constraints," *Eur. J. Oper. Res.*, vol. 280, no. 3, pp. 1018–1034, Feb. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221719306368

[21] F. Gzara, S. Elhedhli, and B. C. Yildiz, "The pallet loading problem: Three-dimensional bin packing with practical constraints," *Eur. J. Oper. Res.*, vol. 287, no. 3, pp. 1062–1074, Dec. 2020.

[22] R. R. Júnior, H. H. Yanasse, R. Morabito, and L. Junqueira, "A hybrid approach for a multi-compartment container loading problem," *Expert Syst. Appl.*, vol. 137, pp. 471–492, Dec. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417419304993

[23] R. D. Saraiva, N. Nepomuceno, and P. R. Pinheiro, "A two-phase approach for single container loading with weakly heterogeneous boxes," *Algorithms*, vol. 12, no. 4, p. 67, Mar. 2019. [Online]. Available: https://www.mdpi.com/1999-4893/12/4/67

[24] F. Wang and K. Hauser, "Stable bin packing of non-convex 3D objects with a robot manipulator," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 8698–8704.

[25] B. Mahvash, A. Awasthi, and S. Chauhan, "A column generation-based heuristic for the three-dimensional bin packing problem with rotation," *J. Oper. Res. Soc.*, vol. 69, no. 1, pp. 78–90, Jan. 2018.

[26] A. G. Ramos, E. Silva, and J. F. Oliveira, "A new load balance methodology for container loading problem in road transportation," *Eur. J. Oper. Res.*, vol. 266, no. 3, pp. 1140–1152, May 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221717309633

[27] C. Paquay, S. Limbourg, and M. Schyns, "A tailored two-phase constructive heuristic for the three-dimensional multiple bin size bin packing problem with transportation constraints," *Eur. J. Oper. Res.*, vol. 267, no. 1, pp. 52–64, May 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221717310214

[28] T. S. Li, C.-Y. Liu, P.-H. Kuo, N.-C. Fang, C.-H. Li, C.-W. Cheng, C.-Y. Hsieh, L.-F. Wu, J.-J. Liang, and C.-Y. Chen, "A three-dimensional adaptive PSO-based packing algorithm for an IoT-based automated e-fulfillment packaging system," *IEEE Access*, vol. 5, pp. 9188–9205, 2017.

[29] J. F. Correcher, M. T. Alonso, F. Parreño, and R. Alvarez-Valdes, "Solving a large multicontainer loading problem in the car manufacturing industry," *Comput. Oper. Res.*, vol. 82, pp. 139–152, Jun. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054817300126

[30] M. T. Alonso, R. Alvarez-Valdes, F. Parreño, and J. M. Tamarit, "Algorithms for pallet building and truck loading in an interdepot transportation problem," *Math. Problems Eng.*, vol. 2016, pp. 1–11, Dec. 2016.

[31] M. D. G. Costa and M. E. Captivo, "Weight distribution in container loading: A case study," *Int. Trans. Oper. Res.*, vol. 23, nos. 1–2, pp. 239–263, Jan. 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12145

[32] O. C. B. D. Araújo and V. A. Armentano, "A multi-start random constructive heuristic for the container loading problem," *Pesquisa Operacional*, vol. 27, no. 2, pp. 311–331, Aug. 2007, doi: 10.1590/s0101-74382007000200007.

[33] D. Pisinger, "Heuristics for the container loading problem," *Eur. J. Oper. Res.*, vol. 141, no. 2, pp. 382–392, Sep. 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221702001327

[34] W. Zhu and A. Lim, "A new iterative-doubling Greedy–Lookahead algorithm for the single container loading problem," *Eur. J. Oper. Res.*, vol. 222, no. 3, pp. 408–417, Nov. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221712003529

[35] M. T. Alonso, R. Alvarez-Valdes, M. Iori, F. Parreño, and J. M. Tamarit, "Mathematical models for multicontainer loading problems," *Omega*, vol. 66, pp. 106–117, Jan. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305048316000335

[36] A. Moura and J. F. Oliveira, "A GRASP approach to the container-loading problem," *IEEE Intell. Syst.*, vol. 20, no. 4, pp. 50–57, Jul. 2005.

[37] O. Faroe, D. Pisinger, and M. Zachariasen, "Guided local search for the three-dimensional bin-packing problem," *INFORMS J. Comput.*, vol. 15, no. 3, pp. 267–283, Aug. 2003.

[38] S. Nishiyama, C. Lee, and T. Mashita, "Designing a flexible evaluation of container loading using physics simulation," in *Proc. 3rd Int. Conf. Optim. Learn.* Cádiz, Spain: Springer, 2020, pp. 255–268.

[39] R. D. Saraiva, N. Nepomuceno, and P. R. Pinheiro, "A layer-building algorithm for the three-dimensional multiple bin packing problem: A case study in an automotive company," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 490–495, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896315003687

[40] J.-F. Tsai, P.-C. Wang, and M.-H. Lin, "A global optimization approach for solving three-dimensional open dimension rectangular packing problems," *Optimization*, vol. 64, no. 12, pp. 2601–2618, Dec. 2015.

[41] M. Hifi, I. Kacem, S. Nègre, and L. Wu, "A linear programming approach for the three-dimensional bin-packing problem," *Electron. Notes Discrete Math.*, vol. 36, pp. 993–1000, Aug. 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1571065310001277

[42] S. Martello, D. Pisinger, and D. Vigo, "The three-dimensional bin packing problem," *Operations Res.*, vol. 48, no. 2, pp. 256–267, Apr. 2000.

[43] O. Kundu, S. Dutta, and S. Kumar, "Deep-pack: A vision-based 2D online bin packing algorithm with deep reinforcement learning," in *Proc. 28th IEEE Int. Conf. Robot Human Interact. Commun. (RO-MAN)*, Oct. 2019, pp. 1–7.

[44] L. Duan, H. Hu, Y. Qian, Y. Gong, X. Zhang, Y. Xu, and J. Wei, "A multi-task selected learning approach for solving 3D flexible bin packing problem," 2018, *arXiv:1804.06896*.

[45] K. Karabulut and M. M. Inceoğlu, "A hybrid genetic algorithm for packing in 3D with deepest bottom left with fill method," in *Advances in Information Systems*, T. Yakhno, Ed. Berlin, Germany: Springer, 2004, pp. 441–450.

[46] I. Gimenez-Palacios, M. T. Alonso, R. Alvarez-Valdes, and F. Parreño, "Logistic constraints in container loading problems: The impact of complete shipment conditions," *TOP*, vol. 29, no. 1, pp. 177–203, Apr. 2021, doi: 10.1007/s11750-020-00577-8.

[47] R. Verma, A. Singhal, H. Khadilkar, A. Basumatary, S. Nayak, H. V. Singh, S. Kumar, and R. Sinha, "A generalized reinforcement learning algorithm for online 3D bin-packing," 2020, *arXiv:2007.00463*.

[48] C. T. Ha, T. T. Nguyen, L. T. Bui, and R. Wang, "An online packing heuristic for the three-dimensional container loading problem in dynamic environments and the physical Internet," in *Proc. 20th Eur. Conf. Appl. Evol. Comput.*, Amsterdam, The Netherlands. Cham, Switzerland: Springer, 2017, pp. 140–155.

[49] H. Zhao, Q. She, C. Zhu, Y. Yang, and K. Xu, "Online 3D bin packing with constrained deep reinforcement learning," 2020, *arXiv:2006.14978*.

[50] C. S. Chen, S. M. Lee, and Q. S. Shen, "An analytical model for the container loading problem," *Eur. J. Oper. Res.*, vol. 80, no. 1, pp. 68–76, Jan. 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/037722179400002T

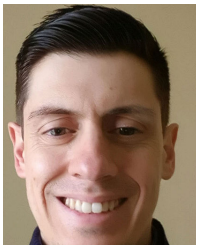
[51] S. Elhedhli, F. Gzara, and B. Yildiz, "Three-dimensional bin packing and mixed-case palletization," *INFORMS J. Optim.*, vol. 1, no. 4, pp. 323–352, Oct. 2019, doi: 10.1287/ijoo.2019.0013.

[52] A. Galrão Ramos, J. F. Oliveira, J. F. Gonçalves, and M. P. Lopes, "A container loading algorithm with static mechanical equilibrium stability constraints," *Transp. Res. B, Methodol.*, vol. 91, pp. 565–581, Sep. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261515302022

[53] D. Zhang, Y. Peng, and S. C. H. Leung, "A heuristic block-loading algorithm based on multi-layer search for the container loading problem," *Comput. Oper. Res.*, vol. 39, no. 10, pp. 2267–2276, Oct. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054811003078

[54] T. Fanslau and A. Bortfeldt, "A tree search algorithm for solving the container loading problem," *INFORMS J. Comput.*, vol. 22, no. 2, pp. 222–235, May 2010, doi: 10.1287/ijoc.1090.0338.
[55] S. Liu, W. Tan, Z. Xu, and X. Liu, "A tree search algorithm for the container loading problem," *Comput. Ind. Eng.*, vol. 75, pp. 20–30, Sep. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835214001776
[56] C.-F. Chien and J.-F. Deng, "A container packing support system for determining and visualizing container packing patterns," *Decis. Support Syst.*, vol. 37, no. 1, pp. 23–34, Apr. 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923602001926
[57] X. Zhao, J. A. Bennell, T. Bektağ, and K. Dowsland, "A comparative review of 3D container loading algorithms," *Int. Trans. Oper. Res.*, vol. 23, nos. 1–2, pp. 287–320, Jan. 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12094
[58] T. Tian, W. Zhu, A. Lim, and L. Wei, "The multiple container loading problem with preference," *Eur. J. Oper. Res.*, vol. 248, no. 1, pp. 84–94, Jan. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221715006232
[59] C. D. Tarantilis, E. E. Zachariadis, and C. T. Kiranoudis, "A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 2, pp. 255–271, Jun. 2009.
[60] H. Hu, X. Zhang, X. Yan, L. Wang, and Y. Xu, "Solving a new 3D bin packing problem with deep reinforcement learning method," 2017, *arXiv:1708.05930*.
[61] A. Laterre, Y. Fu, M. K. Jabri, A.-S. Cohen, D. Kas, K. Hajjar, T. S. Dahl, A. Kerkeni, and K. Beguir, "Ranked reward: Enabling self-play reinforcement learning for combinatorial optimization," 2018, *arXiv:1807.01672*.
[62] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *J. Oper. Res. Soc.*, vol. 64, no. 12, pp. 1695–1724, Dec. 2013, doi: 10.1057/jors.2013.71.
[63] M. G. Epitropakis and E. K. Burke, *Hyper-Heuristics*. Cham, Switzerland: Springer, 2018, pp. 1–57, doi: 10.1007/978-3-319-07153-4.
[64] T. G. Crainic, G. Perboli, and R. Tadei, "Extreme point-based heuristics for three-dimensional bin packing," *INFORMS J. Comput.*, vol. 20, no. 3, pp. 368–384, Aug. 2008.
[65] F.-A. Fortin, F.-M. De Rainville, M.-A. G. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 2171–2175, 2012.
[66] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[67] F. Parreño, R. Alvarez-Valdes, J. F. Oliveira, and J. M. Tamarit, "Neighborhood structures for the container loading problem: A VNS implementation," *J. Heuristics*, vol. 16, no. 1, pp. 1–22, Feb. 2010.
[68] I. Araya, K. Guerrero, and E. Nuñez, "VCS: A new heuristic function for selecting boxes in the single container loading problem," *Comput. Oper. Res.*, vol. 82, pp. 27–35, Jun. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054817300023
[69] A. Bortfeldt and H. Gehring, "A hybrid genetic algorithm for the container loading problem," *Eur. J. Oper. Res.*, vol. 131, no. 1, pp. 143–161, May 2001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221700000552

ANAN ASHRABI ANANNO received the B.S. degree in mechanical engineering from Rajshahi University of Engineering & Technology (RUET), Rajshahi, Bangladesh, in 2019, and the M.S. degree in mechanical engineering from Linköping University, Linköping, Sweden, in 2022, where he is currently pursuing the Ph.D. degree in industrial engineering. He has (co)authored more than 25 papers and book chapters published in journals, books, and conference proceedings. His research interests include discrete optimization, multi-objective design optimization, industrial cyber-physical systems, and design automation.



LUIS RIBEIRO (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in electrical and computer engineering, with specialization in robotics and computer integrated manufacturing from NOVA University Lisbon, Portugal, in 2007 and 2012, respectively. He is an Associate Professor (Biträdande Professor) in manufacturing engineering, with a specialization in industrial cyber-physical systems, with Linköping University, Sweden. His Habilitation (SE Docent) is in manufacturing engineering from Linköping University. He has participated in several national and international research projects in the field of intelligent and plug and produce production systems and he was a Freelance Consultant and a Software Engineer in the automotive industry. He has (co)authored more than 80 papers and chapters published in journals, books, and conference proceedings. He is the author of the book *System Design and Implementation Principles for Industry 4.0*. His research interests include the modeling, development, and implementation of intelligent software solutions-based in multi-agent systems, service-oriented architectures, and cloud for the fast reconfiguration, execution, and ramp-up of batch-size-one production systems. He is a member of the Swedish Advisory Production Council. He is also the Vice-Chair of the IEEE Technical Committee on Industrial Agents.