Journal of Heuristics (2026) 32:11
https://doi.org/10.1007/s10732-026-09586-5

Can machine learning help in solving the pallet loading optimization problem?

Mauro Dell'Amico[1,2] · Giorgia Franchini[3] · Matteo Magnani[1] · Luca Zanni[3,4]

Received: 16 May 2024 / Revised: 4 August 2025 / Accepted: 9 February 2026

Abstract
The Distributor's Pallet Loading Problem aims to optimize the loading of different 3D boxes on the minimum number of pallets. We consider an Integer Linear Programming (ILP) model for the problem that includes constraints deriving from real applications, such as stability and compression limits. In order to solve the ILP problem efficiently, we propose a method that exploits Machine Learning algorithms to classify predetermined layers of boxes, based on their "importance" of being used for an ILP solution. This classification is used to heuristically limit the number of layers taken into account by the ILP solver. We demonstrate the effectiveness of our approach by comparing the ILP solution with and without the Machine Learning component. The numerical results show that the proposed Machine Learning matheuristic approach achieves optimized pallet loading solutions in significantly reduced computational time.

Mauro Dell'Amico, Giorgia Franchini, Matteo Magnani and Luca Zanni contributed equally to this work.

✉ Matteo Magnani
matteo.magnani@unimore.it

Mauro Dell'Amico
mauro.dellamico@unimore.it

Giorgia Franchini
giorgia.franchini@unimore.it

Luca Zanni
luca.zanni@unimore.it

1 Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Via Amendola 2, Reggio Emilia 42122, Italy

2 Centro Interdipartimentale En&Tech, University of Modena and Reggio Emilia, Piazzale Europa 3, Reggio Emilia 42124, Italy

3 Department of Physics, Informatics and Mathematics, University of Modena and Reggio Emilia, Via Campi 213/B, Modena 41125, Italy

4 Institute of Informatics and Telematics, National Research Council, Pisa 56124, Italy

Published online: 24 February 2026

Keywords Distributor's Pallet Loading Problem · Combinatorial Optimization · Integer Linear Programming · Machine Learning · Random Forest · Support Vector Machine for Regression

1 Introduction

The **Distributor's Pallet Loading Problem** (DPLP) involves the efficient loading of parallelepiped-shaped boxes onto pallets in an optimal manner. It has garnered significant attention from the operational research community and from various industries involved in logistics, transportation, storage, and small to medium-sized packaging production. The primary objective is to determine the best approach for loading of boxes onto a minimal number of pallets. The boxes packed must be arranged into layers, where a layer is defined as a grouping of boxes that neither overlap on their supporting surface nor exceed the dimensions of the pallets. We take into account several real-life constraints, including:

- **Dimension Limit**: Ensure that the dimensions of a loaded pallet does not exceed predefined size limits, allowing efficient stacking and handling.
- **Weight Limit**: Restrict the total weight of all boxes loaded onto a pallet to not exceed a given weight threshold, ensuring safe transport and handling.
- **Stackability Constraint**: The property of a layer to support other layers stacked above it.
- **Stability Constraint**: Ensure that layers stacked below another layer has an area large enough to support the above layer.
- **Compression Constraint**: Establish the maximum weight that can be placed on a box, preventing compression-related damage.

Our problem assumes the possibility of rotating boxes 90 degrees on their supporting surface, parallel to the dimensions of the pallet's base. The problem asks to minimize the number of pallets used to achieve cost savings in material transportation and storage. The DPLP with layering is a special case of the three-dimensional bin packing problem (3D-BPP) (Martello et al. 2000), which is a classical combinatorial problem, with the added constraints of stability, stackability and compression.

An effective method to solve complex combinatorial problems is based on extended mathematical formulations and branch-and-prize (Barnhart et al. 1998). In this approach, a master model solves a covering problem where the variables are exponentially many, therefore a column generation approach for solving the LP relaxation and an overall branch-and-price for finding an optimal integer solution are necessary. For DPLP a possible model associates the variables to all the possible layers, and the master selects the layers to be used in each pallet by ensuring that each box is loaded exactly once, and the above side constraints are satisfied. A common way to design approximate algorithms for hard problems is to heuristically limit the set of variables (layers) and optimally solve the resulting compact model (Elhedhli et al. 2019; Mahvash et al. 2018). The primary objective of this paper is to investigate the potential of Machine Learning (ML) to achieve an effective selection of useful layers by employing prediction algorithms, namely Random Forest, Support Vector Regression, and an

hybrid approach combining both. In this case, regression ML models are employed to identify which layers are truly relevant for solving a given instance. The goal is not to enhance the quality of the solution but to reduce the solver's computation time by limiting the number of layers available for the heuristics involved in finding the solution to each instance. This integration enables a faster algorithm, efficiently narrowing the solution space while maintaining high solution quality. While prior research has applied Machine Learning to combinatorial optimization, our approach distinguishes itself by specifically combining Machine Learning with integer linear programming to improve variable selection while including real-world logistics applications and real constraints, where time efficiency is critical.

This work starts from our previous work (Magnani et al. n.d.), in which we presented an ML method focused on a simplified version of the problem with exactly 5 box types for each instance and at least 5 boxes for each type. In this work we introduce a new representation which allows to manage box type variability. The new representation is able to generalize from the training set, i.e. to solve instances with a larger number of box types with respect to the number used in the training. Moreover, we improve the sampling and balancing strategies for training the regression models, combining ensemble methods and balancing techniques to enhance performance. Overall, this work moves beyond a proof of concept to a robust and scalable solution applicable to real-world scenarios.

The paper is organized as follows. In the next section, we provide an overview of the existing literature that addresses the challenges posed by the 3D bin packing, container loading, and manufacturer's pallet loading problems, all of which share similarities with the distributor's loading pallet problem. In Section 3 we present a detailed formal description of the DPLP, we delve into an Integer Linear Programming (ILP) that models the problem and finally we present a matheuristic approach for solving the DPLP based on solving the ILP with a small set of variables, heuristically generated. Section 4 introduces the reader to the ML-based matheuristic approach designed to enhance the efficiency of the ILP solution by classifying variables (layers) based on their likelihood of being included in an optimal solution. The computational experiments and concluding remarks are then presented in the last two sections of this work.

2 Related works

Over the years, the packing problem has garnered considerable attention from both academic researchers and industry practitioners due to its applicability across various sectors. This wide scope has led to the exploration of different variants of the problem and generalizations that are relevant in different domains.

Among these, the DPLP has been a focal point for various researchers. The DPLP with layers is a special case of the three dimentional bin packing problem (3D-BPP), which involves efficiently packing a set of three-dimensional objects into a limited number of three-dimensional bins or containers. The objective of solving the 3D bin packing problem is to find an efficient packing arrangement that minimizes wasted space and maximizes the utilization of available resources while adhering to the spec-

ified constraints. This problem is known to be NP-hard, and various heuristics (Lodi et al. 2002; Crainic et al. 2008; Paquay et al. 2018) are commonly used to find good solutions in practice. For detailed information on packing problems, (Dowsland and Dowsland 1992) offers a classification of various types, while recent approaches for solving online and offline 3D bin packing problems are summarized in Ali et al. (2022).

The container loading problem is similar to the 3D-BPP but focusses specifically on packing a set of items into one or more containers, typically with the goal of optimizing some objective function, such as maximizing the used space, or optmizing the load balance into the cotainer. Many studies deal with this problem (Alonso et al. n.d.; Mart Nez et al. 2016; Moura and Oliveira 2005; Egeblad et al. 2010; Toffolo et al. 2016; Ranck et al. 2019; Saraiva et al. 2015). These works use various strategies, from ILP strategies (Alonso et al. n.d.) and GRASP methods (Mart Nez et al. 2016; Moura and Oliveira 2005) to heuristic approaches (Egeblad et al. 2010; Toffolo et al. 2016) and a combination of heuristics and MILP methods (Ranck et al. 2019) to resolve the problem. (Saraiva et al. 2015) presents a greedy strategy based on layers to tackle this variant of the problem. Common constraints used in container loading and packing problems are extensively discussed in Bortfeldt and Scher (2013) and Zhao et al. (2014).

A similar problem to the DPLP is the so called Manufacturer's Pallet Loading Problem (MPLP), which asks to minimize the number of bins used for loading identical boxes. Some studies have been done for the MPLP: we refer the reader to Morabito and Morales (1998); Silva et al. (2014); Gunawardena et al. (2021) for different solutions to this problem. The DPLP is a generalization of the 3D-BPP, container loading and MPLP problems, which asks to load non-identical boxes on the fewest possible number of pallets, satisfying constraints such as load balancing and compression constraints, as well as stability for transportation. Real-world packing problems, similar to the one addressed in this paper, have been explored in studies such as Ancora et al. (2020); Bischoff et al. (1995); Bischoff and Ratcliff (1995); Erbayrak et al. (2021); Gzara et al. (2020); Piyachayawat and Mungwattana (2017); Scheithauer and Terno (1999). Different methodologies have been used, ranging from hybrid genetic strategies (Ancora et al. 2020) to heuristic approaches (Bischoff et al. 1995; Bischoff and Ratcliff 1995; Piyachayawat and Mungwattana 2017; Erbayrak et al. 2021; Scheithauer and Terno 1999) that encompass greedy, genetic, and differential evolution algorithms, as well as exact MILP and branch-and-bound methods. In contrast, (Gzara et al. 2020) and (Harrath 2021) present innovative layer-based column generation and heuristic strategies. Moreover, related works such as Iori et al. (2020) incorporate visibility and contiguity constraints, leveraging GRASP algorithms, while Tresca et al. (2022) utilizes two MILP formulations for layer creation and palletization. These studies focus on minimizing the empty spaces formed during layering and determining optimal layer-loading sequences to minimize the number of pallets used. In the study by Dell'Amico and Magnani (2021), an heuristic approach was introduced to create layers made of boxes which are selected and packed into a minimum number of pallets using an ILP model.

While various approaches have been explored in the realm of packing problems, the incorporation of real-world constraints remains an area that warrants further exploration and development in the field. It should be noted that relatively less research has

been dedicated to the mathematical modeling of packing problems with real-world constraints such as weight and compression limits, stackability criteria, which are focal points in this paper.

On the other hand, ML-based regressors have been developed in the literature in recent years, the purpose of which is to learn how to associate different examples of a data set with a score that describes their goodness in the specific application. Such regressors, or performance predictors, have been developed in the field of imaging to be able to associate images with a goodness index (Bonettini et al. 2023), in the field of Deep Learning to discriminate between Neural Networks with good performance and not (Franchini et al. 2023), and in the field of hardware to indicate the efficiency of the various accelerators with respect to the assigned operations (Masola et al. 2023). In a recent literature work (Magnani et al. n.d.), the authors tackle the optimization of pallet loading, providing a starting point for our study. Machine Learning is transforming the field of combinatorial optimization (Bengio et al. 2021). By combining Machine Learning with meta-heuristics, solution processes are becoming faster and more efficient (Mamaghan et al. 2021). Techniques like reinforcement learning (Barrett et al. 2020; Mazyavkina et al. 2021) and neural networks (Bello et al. 2016) offer flexible solutions to tackle combinatorial problems. Notably, Machine Learning-based approaches have been successfully applied to the pallet loading problem (Aylak et al. 2021; Layeb and Omri 2024). More broadly, similar algorithmic methodologies can be found in the works on Machine Learning for column generation and selection algorithms in solving MILP (Furian et al. 2021; Lodi and Zarpellon 2017; Morabit et al. 2021).

3 Formal description and a matheuristic algorithm

In this section, we present a formal description of the DPLP and outline a matheuristic approach to address it. We begin by formally defining the problem and its key constituents, followed by the description of an ILP that model the loading of pallets. Finally, we describe the components of the matheuristic algorithm for solving the DPLP that limits through heuristics the dimension of the set of variables of the ILP, in order to speed up its solution.

3.1 Formalization of the DPLP

The DPLP involves efficient arranging of three-dimentional boxes onto pallets, with the aim of minimizing the number of pallets used. Boxes belong to a set $\mathscr{B}$ categorized into types within set $\mathscr{I}$, each type $i \in \mathscr{I}$ comprising n_i identical boxes. The boxes of type $i \in \mathscr{I}$ are characterized by the following attributes: width $w_i \in \mathbb{N}$, depth $d_i \in \mathbb{N}$, height $h_i \in \mathbb{N}$, weight $p_i \in \mathbb{R}^+$, and compression index $c_i \in \mathbb{R}^+$. A box $j \in \mathscr{B}$ corresponds to type $i(j) \in \mathscr{I}$.

An infinite set $\mathscr{P}$ of identical pallets is available, each with a loading surface with width $W \in \mathbb{N}$ and depth $D \in \mathbb{N}$, and it can be loaded with boxes up to the maximum

height $H \in \mathbb{N}$. Dimensions of box types satisfy constraints: $w_i \leq W$, $d_i \leq D$, and $h_i \leq H$.

We call *layer* a two-dimensional packing of a subset of box types, with repetitions, within the loading surface $W \times D$. Let $\mathscr{L}$ be the set of all possible layers. A layer $l \in \mathscr{L}$ includes a set $\mathscr{B}_l$ of assigned box types and one of the feasible two-dimentional box arrangements. Set $\mathscr{L}$ is a subset of the power set $\mathcal{P}(\mathscr{B})$ because not all combinations of box types have a feasible two-dimensional packing. The height of the layer is denoted by H_l and is the maximum height of the boxes within it, while A_l denotes the total net area occupied by the boxes in the layer i.e., the total area of the boxes, disregarding the holes between them. Box types can be arranged in a layer by allowing them to rotated 90 degrees, but not permitting vertical rotations. The layers overall selected for the solution will load each box type i n_i times on the pallets. In fact, each layer can be used multiple times for solving the problem, being loaded at different levels on each pallet.

In order to allow physical loading of layers one on the other, some practical rules must be respected. The loading of layers must respect a **height** limit, meaning that the sum of the heights of the layers loaded on each pallet must not exceed the maximum height H. Similarly, the sum of the weight of the loaded layers on each pallet must respect a **weight** limit $P \in \mathbb{R}^+$. The **stackability** constraint imposes that height difference between the box types in a layer must not exceed a parameter $\Delta h \in \mathbb{N}$, except for the layer loaded on the top level on a pallet. In practice, if two box types i and j are loaded on the same layer, their corresponding heights h_i and h_j must satisfy:

$$|h_i - h_j| \leq \Delta h. \tag{1}$$

The **stability** constraint ensures suitability when loading one layer over another by imposing a requirement for the area ratio. For two layers l and m, where m is loaded immediately above l, it demands

$$\alpha A_l \geq A_m \tag{2}$$

asserting that the upper layer's area (A_m) must not exceed $\alpha \geq 1$ times the lower layer's area (A_l).

The **compression** constraint requires each layer to be able to support the layers loaded above it, in terms of weight. We denote by C_l the compression factor, which represents the maximum weight that a layer l can support: with the definition $C_l = A_l \min_{i \in \mathcal{B}_l} c_i$ the compression constraint results into the inequality:

$$C_l \geq \sum_{k \in U} \sum_{i \in \mathscr{B}_k} p_i \tag{3}$$

where U represents the set of layers loaded above layer l, on the same pallet.

Resuming, the DPLP aims to load all boxes into minimal pallets while adhering to the following multiple constraints.

c1. **Numerosity constraint**: All $|\mathscr{B}|$ boxes must be packed.

c2. **Height constraint**: The cumulative layer's height on each pallet must not exceed H.
c3. **Weight limit**: The total weight of the layers on each pallet must not exceed P.
c4. **Stackability constraint**: Except for the layers loaded at the top level, each layer must comply with the height difference constraint (1).
c5. **Stability constraint**: Layers must satisfy the area ratio condition between the overlaid layers (2).
c6. **Compression constraint**: Each layer must verify the compression inequality (3).

3.2 Integer linear programming formulation

We introduce a mathematical model with an exponential number of variables, that aims to optimally load layers using the minimum number of pallets while packing all the given boxes. The layers are described by a matrix B of size $|\mathscr{I}| \times |\mathscr{L}|$, where b_{il} represents the number of boxes of type i packed in layer l. Two sets of binary variables are used:

$$y_p = \begin{cases} 1 & \text{if pallet } p \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{lpk} = \begin{cases} 1 & \text{if layer} l \text{ is stacked on pallet} p \text{ at level } k \\ 0 & \text{otherwise} \end{cases}$$

The number of pallets to be considered i.e., the cardinality of $\mathscr{P}$, can be bounded by the number of boxes, or, more effectively, by any upper bound on the optimal solution value. The number of levels (layers) on a pallet can be bounded by $\kappa = \lfloor H/min_{i\in\mathscr{I}} h_i \rfloor$. We split the set of layers $\mathscr{L}$ in two subsets $\mathscr{L}^{\leq} = \{l \in \mathscr{L} : (1)$ holds for all $i, j \in \mathscr{B}_l\}$ and $\mathscr{L}^{\geq} = \mathscr{L} \setminus \mathscr{L}^{\leq}$. Finally, let $P_l = \sum_{i\in\mathscr{B}_l} p_i$ represent the weight of each layer l. The ILP for modeling the stacking layers on pallets is expressed as follows:

$$\min \sum_{p\in\mathscr{P}} y_p \tag{4}$$

$$\sum_{k=1}^{\kappa} \sum_{p\in\mathscr{P}} \sum_{l\in\mathscr{L}} b_{il} x_{lpk} = n_i \qquad i \in \mathscr{I} \tag{5}$$

$$\sum_{k=1}^{\kappa} \sum_{l\in\mathscr{L}} H_l x_{lpk} \leq H y_p \qquad p \in \mathscr{P} \tag{6}$$

$$\sum_{k=1}^{\kappa} \sum_{l\in\mathscr{L}} x_{lpk} P_l \leq P y_p \qquad p \in \mathscr{P} \tag{7}$$

$$\sum_{h=k+1}^{\kappa} \sum_{\ell\in\mathscr{L}} x_{\ell ph} \leq \kappa(1 - x_{lpk}) \qquad l \in \mathscr{L}^{\geq}, p \in \mathscr{P}, k \in \{1, \ldots, \kappa - 1\} \tag{8}$$

$$x_{lpk} + x_{mp(k+1)} \leq 1 \qquad l, m \in \mathscr{L}, A_m > \alpha A_l,\ p \in \mathscr{P},\ k \in \{1, \ldots, \kappa - 1\} \tag{9}$$

$$\sum_{h=k+1}^{\kappa} \sum_{\ell \in \mathscr{L}} x_{\ell ph} P_\ell \leq C_l + P(1 - x_{lpk}) \qquad l \in \mathscr{L}^{\leq}, p \in \mathscr{P}, k \in \{1, \ldots, \kappa - 1\} \tag{10}$$

$$\sum_{l \in \mathscr{L}} x_{lpk} \leq 1 \qquad p \in \mathscr{P}, k \in \{1, \ldots, \kappa\} \tag{11}$$

$$\sum_{l \in \mathscr{L}} (x_{lpk} - x_{lp(k-1)}) \leq 0 \qquad p \in \mathscr{P}, k \in \{2, \ldots, \kappa\}, \tag{12}$$

$$y_p \leq y_{p-1} \qquad p \in \{2, \ldots, |\mathscr{P}|\} \tag{13}$$

$$y_p \in \{0, 1\} \qquad p \in \mathscr{P} \tag{14}$$

$$x_{lpk} \in \{0, 1\} \qquad l \in \mathscr{L}, p \in \mathscr{P}, k \in \{1, \ldots, \kappa\} \tag{15}$$

The aim of objective function (4) is to minimize the number of pallets used in the packing of the boxes. Requirement **c1** is implemented by constraints (5), which ensures that each box type is packed exactly n_i times on a pallet. Constraints (6) enforces requirement **c2** by restricting the sum of the heights of the layers loaded on a pallet to be less than or equal to the height H. While the weight limit **c3** is enforced by (7), the stackability requirement **c4** is satisfied by constraints (8). Constraints (9) and (10) guarantee both requirements **c5** and **c6**. The constraint (11) limits the packing of at most one layer per level. In order to break symmetries, constraints (12) ensure that level k is loaded only if level $k - 1$ has been loaded, and constraints (13) impose that pallet p is used only if pallet $p - 1$ has been used. The definition of the domains of the variables follows.

3.3 A Matheuristic for DPLP

Solving the ILP outlined in Section 3.2 is impractical due to the exponential size of the set $\mathscr{L}$. To address this challenge, instead of a complete column generation approach, we propose an algorithm that heuristically generates a subset $\mathcal{L} \subset \mathscr{L}$, that contains a limited (polynomial) number of layers, to serve as input for the ILP instead of the entire set $\mathscr{L}$. The algorithm generates the subset $\mathcal{L}$ which comprises layers of box types, with the majority satisfying the inequality (1) and all exhibiting feasible 2D packing. Then, the set $\mathcal{L}$ is given as input to the ILP for finding a solution to the problem. Resuming, the algorithm for solving DPLP (MH-PL), which was firstly introduced in Dell'Amico and Magnani (2021), is composed by two main stages:

1. Build set $\mathcal{L}$ with Algorithm 1 BUILDLAYERS;
2. Solve the ILP (4)-(15), given as input the set $\mathcal{L}$.

The solution of DPLP is not guaranteed to be optimal, because of the limited size of set $\mathcal{L}$. The MH-PL can quickly solve small and medium instances where we have few box types (up to 7) and their count does not exceed 200. We refer the reader to Dell'Amico and Magnani (2021) for further details on performance of the algorithm.

Table 1 Two-dimensional bin packing algorithms for packing layers

	Algorithm	Methodology	Sorting	Rotation allowed
1	Maximal rectangle	Best-fit	Decreasing area	Yes
2	Maximal rectangle	Best-fit	Decreasing area	No
3	Maximal rectangle	Best-fit	Decreasing diagonal	Yes
4	Maximal rectangle	Best-fit	Decreasing diagonal	No
5	Maximal rectangle	Best-fit	Decreasing longest side	Yes
6	Maximal rectangle	Best-fit	Decreasing longest side	No
7	Skyline	Best-fit	Decreasing area	Yes
8	Skyline	Best-fit	Decreasing area	No
9	Skyline	Best-fit	Decreasing diagonal	Yes
10	Skyline	Best-fit	Decreasing diagonal	No
11	Skyline	Best-fit	Decreasing longest side	Yes
12	Skyline	Best-fit	Decreasing longest side	No

The Algorithm 1 BUILDLAYERS for constructing the set $\mathcal{L}$ involves a two-phase approach. Initially, box types are divided into "families" where the box types have heights that almost satisfy the condition (1) on height differences, through Algorithm 2 CREATEFAMILIES. In the second phase, classical 2D packing heuristic algorithms (Lijun et al. 2009) named *maximal rectangles* and *skyline* are used to feasibly pack boxes onto layers. To create different layers, various versions of these algorithms are implemented, using the *Best-fit* strategy and allowing (or not) boxes to rotate by 90 degrees, sorting them by Decreasing area of the surfaces of the boxes, or sorting by the longest side or major diagonal of the boxes before running the algorithms. In Table 1, we summarize the versions of algorithms that we have chosen to use for layer construction.

With a_{max} we denote the number of 2D heuristic algorithms, and with $2D\text{-}H_a$ the a-th heuristic algorithm as in Table 1. Each algorithm is applied to every family in turn, creating layers that are added to the current set of layers $\mathcal{L}$ if not yet present. In our case, we chose $a_{max} = 12$. The loop in line 4 of Algorithm 1 calls Algorithm 2 multiple times, thus creating different families of box types. Algorithm 2 takes as input the number of families f we want to form and generates a partition of the set of box types $\mathscr{I}$, each of which is assigned to one of the f families $\mathscr{I}(1), \ldots, \mathscr{I}(f)$.

The Algorithm 2 starts by dividing the interval $[1; |\mathscr{I}|]$ into f subintervals, with the first $f-1$ having identical widths $\lceil |\mathscr{I}|/f \rceil$, and possibly the f-th one with a smaller width than the others. A pivot box type index $\nu(i)$ is randomly selected from each interval i (with $i \in 1, \ldots, f$). Each family $\mathscr{I}(i)$ is defined as the set of box types with absolute height difference from $h_{\nu(i)}$ not exceeding $\frac{1}{2}(1+\gamma)\Delta h$, where γ is a small random value. After this initial selection of box types, possible intersections are removed, and box types not inserted into any subset, if any, are assigned to the set with the closest pivot height. Each generated family consists of box types that "almost" satisfy the condition (1).

Algorithm 1 Algorithm for building layers

procedure BUILDLAYERS(boxes $\mathscr{B}$ and their types $\mathscr{I}$, layers' sizes)
  Sort the box types in $\mathscr{I}$ by non-decreasing heights (i.e., $h_1 \le h_2 \le \ldots, \le h_{|\mathscr{I}|}$)
  $\mathcal{L} = \emptyset$;
  for $f = f_{\min}, \ldots, f_{\max}$ **do**
    $(\mathscr{I}(1), \ldots, \mathscr{I}(f))$=CREATEFAMILIES($f$);
    for $i = 1, \ldots, f$ **do**
      for $a = 1, \ldots, a_{\max}$ **do**
        pack the boxes in $\mathscr{I}(i)$ with $2D$-H_a heuristic giving layers L
        $\mathcal{L} = \mathcal{L} \cup L$;
      end for
    end for
  end for
end procedure
return $\mathcal{L}$

Algorithm 2 Procedure for creating the box type families

procedure CREATEFAMILIES(number of families f)
  Let $n = |\mathscr{I}|$; Choose randomly $\gamma \in [0.3, 0.5]$
  for $j = 1, \ldots, f$ **do**
    Let $\nu(j) =$ RANDOM $((j-1)\lceil n/f \rceil, \min(j\lceil n/f \rceil, |\mathscr{I}|))$
    $\mathscr{I}(1) = 1, \ldots, \bar{\imath}$ with $\bar{\imath} = \arg\max\{h_i : h_i \le h_{\nu(1)} + \frac{1}{2}(1+\gamma)\Delta h\}$
  end for
  for $j = 2, \ldots, f-1$ **do**
    $\mathscr{I}(j) = \underline{\imath}, \ldots, \bar{\imath}$ with $\underline{\imath} = \arg\min\{h_i : h_i \ge h_{\nu(j)} - \frac{1}{2}(1+\gamma)\Delta h\}$
    $\bar{\imath} = \arg\max\{h_i : h_i \le h_{\nu(j)} + \frac{1}{2}(1+\gamma)\Delta h\}$
  end for
  $\mathscr{I}(f) = \underline{\imath}, \ldots, n$ with $\underline{\imath} = \arg\min\{h_i : h_i \ge h_{\nu(f)} - \frac{1}{2}(1+\gamma)\Delta h\}$
  for $j = 1, \ldots, f-1$ **do**
    if $\mathscr{I}(j) \cap \mathscr{I}(j+1) \neq \emptyset$ **then**
      $\mathscr{I}(j+1) = \mathscr{I}(j+1) \setminus \mathscr{I}(j)$ ▷ exclude intersections
    end if
  end for
  for $i \in \mathscr{I}$ **do**
    if $i \notin \bigcup_{j=1}^{f} \mathscr{I}(j)$ **then**
      assign i to the set with nearest pivot height
    end if
  end for
end procedure
return $(\mathscr{I}(1), \ldots, \mathscr{I}(f))$

4 The hybrid machine learning integer linear programming method

The solution of medium and big instances of DPLP with MH-PL becomes excessively time-consuming when we solve the ILP outlined in Section 3.2 with $|\mathcal{L}| \ge 100$. This issue becomes particularly evident when there is a medium-large number of box types ($|\mathscr{I}| \ge 10$) and a large number of boxes ($|\mathscr{B}| \ge 200$).

To address this issue and expedite the solution process, we propose the introduction of a Machine Learning-based matheuristic approach. This methodology is specifically designed to improve the efficiency of solving the DPLP through algorithm MH-PL.

Our approach leverages ML methods to select “useful" layers, which are more likely to be included in an optimal solution, and include them in a limited set. This restricted set is provided as input to the ILP to generate effective solutions within a reasonable time frame. We call MLrILPthe ILP model (4)-(15) with $\mathcal{L}$ given by the layers selected by the ML method.

Specifically, we construct a data set consisting of several DPLP instances solved though algorithm MH-PL. Then, we train selected ML methods to learn the potentialities of the layers to be part of a high quality solution for a DPLP instance. Finally, we use the trained methods to select, for each instance, a subset, of smaller percentage size, composed of the “good" layers and solve the corresponding ILP.

Our goal is to show that the ML methods are able to clever select the layers, so that the ILP solution is faster and the quality of the final solution is not worse than the solution without any reduction on the set of variables (layers).

4.1 ML techniques

For the ML algorithms, we considered two well-known methodologies: Random Forest (RF) (Breiman 2001) and Support Vector Machines with Regression (SVR) (Burges 1998), along with a hybrid method that combines RF and SVR using the mean of the predicted labels, as suggested in Franchini et al. (2023).

RF is an ensemble learning technique that combines multiple decision trees to make predictions. It is particularly effective for handling high-dimensional data sets and provides robustness against outliers and noisy data. One of the advantages of RF is that it offers insights into feature importance, which aids in understanding the contributions of different variables. In our implementation, RF was trained using the Bagging methodology (Breiman 1996) to increase the independence among the individual decision trees. A fundamental role in the RF model is the choice of the number of binary trees n_T to be put together to form the forest. The choice of n_T is crucial since an excessive number of trees could only increase the computational cost without an increase in performance, due to the loss of independence among the various predictors.

SVR is a supervised learning algorithm that finds the best-fitting hyperplane in a high-dimensional feature space. It excels at capturing non-linear relationships between variables by utilizing kernel functions. SVR is especially useful when dealing with complex relationships and can provide accurate predictions. We tried SVR with different kernels and we found the best result by exploiting the Gaussian kernel. Thus, our SVR model involves three key hyperarameters, used to balance model complexity and generalization capability (Burges 1998):

- ε, which defines the width of the “epsilon-insensitive zone” within which errors are ignored in the loss function, controlling the tolerance for error.
- C represents the regularization parameter, balancing the trade-off between the complexity of the decision function and the data fidelity term.
- σ, used in the Gaussian kernel function.

The hybrid method (HYB) combines the predictions of RF and SVR by taking the mean of the predicted labels. This approach leverages the strengths of both RF and

SVR and can potentially improve the overall performance of each of the other two regressors, through the well-known mechanism of ensemble methods.

4.2 Data set for ML methodology: features and labels of data set

The ML methodology data set is constructed by solving numerous instances using the MH- PL Algorithm. It contains information about the layers and the number of times these layers were used to solve instances. Each entry in the data set corresponds to a pair of instance-layer, representing one instance and one layer, with the information on how many times that layer is used to solve that instance. Both instances and layers are represented by vectors of $|\mathscr{I}|$ integer numbers, where each element denotes the quantity of a specific box type present in an instance and in a layer. An element in the data set is described by $(2\Phi + 5)$ features and a single label, where $\Phi = |T|$, where $T \supseteq \mathscr{I}$ is the set containing all possible box types that can define an instance. The first Φ features represent the quantities of box types present (or not present) in the instance, and the second Φ features represent the quantities of boxes representing one layer. Additionally, 5 features are incorporated, encompassing the sum of box quantities, height, weight, compression, and area of each layer. One label denotes the number of times the layer is used to solve the corresponding instance. Labels are normalized between 0 and 1 by dividing all of them by the maximum label value. This is a way of placing all labels into the same range and giving them probabilistic meaning, even if their sum is not equal to 1. This is very important for generating a ranking of the best layers for each instance in the inference phase. We chose regression models over classification models because regression allows us to quantify the importance of each layer by capturing how frequently it appears in the solution. This provides more granular insights compared to binary classification, which only predicts whether a layer is used or not. With this structure we can represent all instances that have dimension up to the Φ value and which may change up to this maximum value.

In our previous work (Magnani et al. n.d.), the features were able to represent only instances with a pre-fixed number of box types, namely 5 in our experiments. This constraint prevents the possibility of generalization from the training set, that is leveraging knowledge acquired from solving simple instances with few box types to make predictions on instances with a larger number of box types.

4.3 Balanced and unbalanced data sets

The data set construction leads to a large imbalance between the labels. The vast majority of the labels in the data set are equal to 0, meaning that the layer considered is not used in that particular instance. The minority of elements in the data set have non-zero labels, meaning that that particular layer has been used one or more times to solve that instance. We will call *unbalanced* the original data set.

In classification and regression scenarios, dealing with imbalanced data–where one class is inadequately represented–presents distinct challenges for ML algorithms and has garnered significant attention within the community (He and Garcia 2009; Krawczyk 2016). The imbalance affects the performance of commonly used ML tech-

niques, such as RF and SVR, as they often assume that classes are equally distributed. The imbalance is an important issue in healthcare and in diagnosis prediction models, especially, where individuals affected by the disease of interest are the minority of the sample (Khalilia et al. 2011; Majid et al. 2014; Gerych et al. 2019; Gentili et al. 2024). The problem of imbalanced classes can be solved using data-level methods, such as random resampling, which modifies the number of training samples to decrease the imbalance. Random *undersampling* discards samples from the majority group, reducing training time and redundant information. On the other hand, random *oversampling* duplicates samples from the minority class, which can increase the risk of overfitting and significantly extend the training time. To avoid these drawbacks, and in line with both established findings in the literature (Wongvorachan et al. 2023) and our own experimental results (presented in Section 5), we chose to apply undersampling when training the ML predictors. Specifically, we constructed an undersampled data set by randomly selecting a subset of samples from the majority class. This process ensured a balanced representation between the majority and minority classes, which improved the efficiency of the training process and mitigated potential bias without compromising model performance. In the following, we will refer to the reduced data set as the *balanced* one.

In the case of SVR we consider also a technique that combine ensemble and balancing techniques. In particular we refer to an adaptation of Chan and Stolfo (1998) work that conducted a series of initial experiments to determine an optimal regression distribution, followed by sampling techniques that generate multiple training sets with the desired regression distribution. Typically, each training set includes all instances of the minority class and a subset of instances from the majority class. Importantly, each majority-class instance is guaranteed to appear in at least one training set, ensuring no data is wasted. The SVR model is then applied to each training set, and ensemble technique is utilized to create a composite learner from the resulting regressors by calculating the mean value of the predicted labels by the SVR classifiers. The resulting SVR ensemble (Yan et al. 2003) demonstrated superior performance compared to both undersampling and oversampling methods.

For this study, we apply RF, SVR and HYB methods to the data set and solve instances with different reduction percentage on the original ensemble of layers.

4.4 The ML-based matheuristic approach

We finally present the ML-based matheuristic algorithm (ML-PL) to solve the DPLP. Given an instance of the problem and a reduction percentage p ($0 < p < 100$), then ML-PL works in five main steps, for each of the three methods (RF, SVR and HYB):

1. Build the set of layers $\mathcal{L}$ with the Algorithm 1 BUILDLAYERS;
2. Assign labels to layers in the set $\mathcal{L}$, one for each of the ML methods, with values reflecting their likelihood of being included in an optimal solution;
3. Sort layers based on their labels, in decreasing order;
4. Maintain in set $\mathcal{L}$ only layers that come before the p-th percentile of the set.
5. Solve the instance MLrILP given by the reduced set of layers.

In the upcoming section, we will investigate computational experiments conducted to evaluate the performance and effectiveness of the ML-PL algorithm compared to MH-PL using balanced and unbalanced versions of the data set for training each method, together with different ML methods and a reduction percentage in the set $\mathcal{L}$.

5 Computational experiments

To assess the efficacy of the proposed ML-PL algorithm, we performed experiments with various instances, comparing the results achieved with the different ML methods used (RF, SVR, or HYB), varying the size of the instances and the reduction percentages, and using balanced and unbalanced data set during the training phase. This section outlines our findings. All the experiments were carried out on a PC featuring an Intel(R) Core(TM) i9-11900K 3.50GHz processor, 64 GB of RAM, and running the Ubuntu 22.04.1 LTS Operating System. Algorithm implementation was carried out in Python and Matlab. Data set processing and ML methods training were executed using Matlab, while data set building and ILP solution with reduced layers' set were executed using Python. The ILP model was solved using Gurobi 11.0, employing a time limit variable based on the size of the instance, following the linear rule *time limit* $= |\mathscr{B}|$ seconds. Pallet dimensions are those of the standard europallet sizes: 1200 mm in length, 800 mm in width, and a maximum height of 1650 mm. Considering the pallet base height, typically 150 mm, the total height was set to 1800 mm, meeting the standard transportation height requirement. The maximum pallet weight was restricted to 1500 kg, which is the standard maximum weight limit for europallet. Finally, Δh was configured to 20 mm and α, in Equation (9), had a value of 1.1. The settings of these two parameters guarantee stability during transport of small-medium sized boxes.

Boxes dimensions, compression index and weight are taken from a food and beverage logistic company data set and, for our experiments, the cardinality of the set containing all box types is $|T| = 50$ and the cardinality $|\mathscr{I}|$ of box types for each instances varies from 5 to 10, reflecting real transportation problems for the company.

The complete data set used in our experiments is available online at https://github.com/MatteoMagnani95/DPLP.

In order to compare the performances of the two algorithms, MH-PL and ML-PL, we focused on the percentage of average time saved (%AT), the standard deviation of the percentage time saved (%SD), the number of unsolved instances (#unf), and instances that reached the time limit (#TL) set for the solver. The %AT was computed by averaging the percentage differences of computational times between instances solved with MH-PL and ML-PL over the computational time taken by MH-PL. Specifically, given N instances and having denoted as o_i and r_i the computational times for solving instances with MH-PL and ML-PL respectively, the %AT was derived using the formula:

$$\%\text{AT} = \frac{100}{N} \sum_{i=1}^{N} \frac{o_i - r_i}{o_i}.$$

Table 2 Cardinalities of the data sets used for the training phase

Box Count	100	200	300	400	500	600	Total
Number of solved instances	12000	120	120	120	120	120	12600
Number of samples	676084	6592	6858	7134	6941	6904	710513

The calculation of the solution time (r_i) was done as follows:

- If the MLrILP instance i has been optimally solved, then:
  - if the solution with reduction MLrILP solution was equal to the solution without reduction on layer set, the solution time (r_i) was equal to the solver's time for solving the instance.
  - if the number of pallets was not the same as the solution obtained with all layers, then r_i was calculated adding to the solver's time for solving the instance with the reduction on layer set the solver's time for solving the instance with the complete layers' set $\mathscr{L}$.
- If the MLrILP instance i was unfeasible or the time limit was reached without an optimal solution, the solution time for the reduced instance was determined by adding the time required to solve the instance without removing any layer to the computational time taken to demonstrate the unfeasibility or adding the time limit.

Similarly, with the same values o_i and r_i, we defined %SD with the formula:

$$\%\mathrm{SD} = \sqrt{\frac{\sum_{i=1}^{N}(100\frac{o_i - r_i}{o_i} - \%AT)^2}{N}}.$$

The #unf indicated the count of instances proven to be unfeasible because of the reduction of the set of layers within the time limit, while #TL represented the instances that reached the time limit.

5.1 Experiments on instances of different dimensions

For the evaluation of the proposed algorithm, we generated a data set for training the ML methods, solving a total of 12600 instances. A total pool of $|T| = 50$ distinct box types was defined, from which each instance was created by selecting a subset of 5 to 10 box types. The instances also varied in terms of the number of boxes, with total box counts ranging from 100 to 600. We generated many simple instances, more than medium and difficult ones, with 5 box types and a box count of 100, to expedite the construction of the training data set. This choice was deliberate as instances with fewer box types and counts tend to be simpler to solve compared to instances with numerous box types and counts.

Additionally, we solved few more complex instances to ensure a more comprehensive training data set for the ML methods.

In Table 2, for each box count we present the number of solved instances and the number of samples that were generated in the data set, each sample corresponding to one layer constructed for solving one instance (refer to Section 3.3).

These data sets serve as a training set. It is important to note the considerable imbalance in our data sets, with 119001 samples having a non-zero label over 710513 total. This imbalance derives from the fact that many layers generated in this process are not utilized in any instance. On average, less than 17% of the generated layers are employed in solving an instance, resulting in the majority of layers being labeled as 0 for each instance.

The training of RF model took approximately 3575 seconds, while the SVR model required 159726 seconds for the unbalanced strategy. Regarding the balanced data set, the RF model took 664 seconds for training and 23830 seconds for training the SVR model. For the HYB model, it is reasonable to consider a combined time equal to the sum of both individual models on both strategies.

Given that the predictive models were trained in their regression versions, we can set a threshold to precisely select a specific portion of the data set.

5.1.1 ML hyperparameters setting and performance

Assuming that a truly definitive test involves fully solving the instances and evaluating their timing and metrics, in this section we present preliminary results and settings based solely on the performance analysis of the ML methods: RF, SVR, and HYB.

Using the data set described above, we conducted preliminary experiments to tune the RF and SVR hyperparameters. Regarding RF, we focused on the number of trees n_T; we set $n_T = 100$, since we verified that a larger number involved increased computational costs without enhancing performance. As far as SVR is concerned, we tried different values for the hyperparameters C, ε and σ described in Section 4.1. In the end, the best setting was: $C = 0.3706$, $\varepsilon = 0.0371$ and $\sigma = 8.6905$.

As mentioned above, ML models, once trained, must be able, assigned an instance, to return a ranking of the most promising layers to solve it. With this main objective in mind, we can make two types of mistake. The most serious error that can be made is to not include among the first portion of the layers a layer that we will then need to solve the instance; we will denote this error as Type 1 error. The other error, Type 2, is considerably less serious as it involves including an unnecessary layer among those considered by the solver for that instance. We regard the Type 1 error as more severe because it could result in infeasibility, a scenario that cannot arise with a Type 2 error. In order to evaluate the ML algorithms according to the various reduction percentages of the data set, we counted the layers with a non-zero label that were not included in the reduced data set (Type 1 error). The test set consisted of 8443 samples of which 2106 having non-zero label.

In Table 3 we provide the number of Type 1 errors on the test set, for the three ML methods, RF, SVR and HYB, trained on the balanced set, at varying reduction percentages (20%, 30%, 40%, 50%, 60%, and 70%). In addition, we report the percentages of Type 1 errors in relation to the number of non-zero labeled samples. Table 4 presents the Type 1 errors on the test set, using the ML methods trained on the unbalanced set.

Table 3 ML Performance of Methods, on the test set, for t Percentages with Balanced training

	20%	30%	40%	50%	60%	70%
RF	41 (1.95 %)	84 (3.99 %)	129 (6.13 %)	192 (9.12 %)	264 (12.54 %)	386 (18.33 %)
SVR	69 (3.28 %)	151 (7.17 %)	226 (10.73 %)	373 (17.71 %)	515 (24.45 %)	695 (33.00 %)
HYB	60 (2.85 %)	95 (4.51 %)	144 (6.84 %)	192 (9.12 %)	262 (12.44 %)	364 (17.28 %)

Table 4 ML Performance of Methods, on the test set, for Different Reduction Percentages with Unbalanced training

	20%	30%	40%	50%	60%	70%
RF	42 (1.99 %)	71 (3.37 %)	120 (5.7 %)	170 (8.07 %)	241 (11.44 %)	343 (16.29 %)
SVR	135 (6.41 %)	141 (6.7 %)	227 (10.78 %)	370 (17.57 %)	500 (13.74 %)	709 (33.67 %)
HYB	88 (4.18 %)	111 (5.27 %)	142 (6.74 %)	189 (8.97 %)	244 (11.59 %)	324 (15.38 %)

For the unbalanced case, RF was trained once with all the data, whereas in the case of SVR we created 5 models, each of which sees a portion of the data.

The increasing error counts for all methods indicate that as the reduction percentage rises, the number of excluded layers with non-zero label also grows, reflecting a decrease in accuracy. The Random Forest method generally has the lowest number of errors, suggesting it performs the best among the three methods for the tested reduction percentages.

5.1.2 ML-PL performances

In this section we present the results of experiments on the test set. We selected 20 instances for each dimension, ranging from 100 to 600 boxes, which have not been included in the training set.

Tables 5 and 6 present a comparative analysis of the RF, SVR, and HYB methods across varying total box quantities and layers reduction percentage, excluding all unfeasible instances and all instances where the time limit was reached. The metrics considered are the average percentage solution time %AT and the standard deviation of the percentage time saved %SD for solving the ILP model. The %OPT indicates the percentage of instances for which MLrILP achieved the same objective value as the standard ILP method, without reaching the time limit. Each table is organized into columns representing different reduction percentages, ranging from 20% to 70%, and rows corresponding to each method and box count. The tables include results for both the balanced and unbalanced training versions.

When we delve into comparing methods across different reduction percentages, it becomes apparent that there is not a straightforward correlation between reduction rates, performance metrics, and the efficacy of various methods.

In instances where reductions are relatively low, specifically at 20% and 30%, the SVR method consistently emerges as the frontrunner, showcasing superior perfor-

Table 5 Comparison of Methods for Different Reduction Percentages without unfeasible and time limit instances on Balanced training

Red(%)		20			30			40		
Met	Box	%AT	%SD	%OPT	%AT	%SD	%OPT	%AT	%SD	%OPT
RF	100	33.37	1.12	100.00	51.27	1.09	100.00	70.95	0.71	100.00
	200	17.00	2.55	100.00	40.59	1.94	100.00	60.42	2.79	100.00
	300	20.40	3.51	100.00	23.65	3.67	100.00	57.37	2.60	100.00
	400	32.70	1.46	100.00	47.80	2.35	100.00	56.39	5.50	100.00
	500	24.48	10.98	100.00	38.63	8.44	100.00	62.33	6.11	100.00
	600	27.79	10.02	100.00	52.73	4.75	100.00	57.44	8.40	100.00
Mean		**25.96**	**4.94**	**100.00**	**42.44**	**3.71**	**100.00**	**60.48**	**4.69**	**100.00**
SVR	100	40.24	1.09	100.00	55.25	1.06	100.00	73.39	0.96	100.00
	200	21.09	3.21	100.00	52.51	2.73	100.00	68.62	3.43	100.00
	300	28.92	3.00	100.00	38.01	3.03	100.00	29.01	4.03	100.00
	400	36.71	4.72	100.00	59.80	4.66	100.00	61.63	2.95	100.00
	500	33.61	8.28	100.00	53.68	8.28	100.00	61.24	13.19	100.00
	600	34.63	7.13	100.00	55.37	10.57	100.00	65.19	12.50	100.00
Mean		**32.53**	**4.57**	**100.00**	**52.77**	**5.06**	**100.00**	**59.18**	**6.18**	**100.00**
HYB	100	40.93	0.91	100.00	53.94	0.92	100.00	72.89	1.11	100.00
	200	24.60	2.29	100.00	17.98	1.61	100.00	61.03	1.81	100.00
	300	11.37	1.86	100.00	22.93	1.52	100.00	58.87	3.43	100.00
	400	45.49	3.08	100.00	53.16	3.08	100.00	47.99	5.21	100.00
	500	32.25	12.71	100.00	58.65	12.65	100.00	62.62	13.39	100.00
	600	26.22	10.55	100.00	52.21	10.44	100.00	62.18	12.53	100.00
Mean		**30.14**	**5.23**	**100.00**	**43.15**	**5.04**	**100.00**	**60.93**	**6.25**	**100.00**
Red(%)		50			60			70		
Met	Box	%AT	%SD	%OPT	%AT	%SD	%OPT	%AT	%SD	%OPT
RF	100	84.65	1.18	90.00	90.52	1.31	100.00	92.25	0.00	100.00
	200	76.12	4.94	93.75	79.30	3.87	92.30	95.10	0.55	100.00
	300	78.10	5.33	100.00	64.63	2.33	92.30	89.61	1.57	100.00
	400	77.27	4.68	100.00	83.03	4.48	93.33	86.64	3.28	88.00
	500	-9.62	10.71	100.00	78.72	6.18	93.75	70.87	7.91	100.00
	600	43.30	8.84	100.00	72.08	8.11	91.67	88.22	6.11	83.33
Mean		**58.30**	**5.95**	**97.29**	**78.05**	**4.71**	**95.23**	**87.11**	**3.24**	**95.72**
SVR	100	84.66	1.92	100.00	91.55	1.78	100.00	97.72	0.00	100.00
	200	80.48	5.42	100.00	84.44	2.05	85.71	94.98	0.94	100.00
	300	63.02	3.04	100.00	76.47	2.64	84.61	87.70	2.03	100.00
	400	63.92	3.08	100.00	79.96	3.31	93.75	78.95	2.24	87.50
	500	70.80	8.70	100.00	71.24	10.48	94.11	79.10	9.12	91.66
	600	64.36	9.06	100.00	67.89	10.48	91.66	75.29	10.79	100.00
Mean		**71.21**	**5.54**	**100.00**	**78.59**	**5.79**	**93.31**	**85.62**	**4.85**	**96.86**
HYB	100	85.82	2.45	90.00	91.94	1.68	100.00	90.71	0.00	100.00

Table 5 continued

Red(%)		50			60			70		
Met	Box	%AT	%SD	%OPT	%AT	%SD	%OPT	%AT	%SD	%OPT
	200	76.38	1.47	100.00	86.96	2.62	90.90	88.92	2.11	100.00
	300	66.82	3.36	100.00	71.34	2.93	92.30	85.67	2.58	100.00
	400	70.07	3.83	100.00	70.16	2.57	94.11	75.43	3.41	100.00
	500	71.11	12.69	100.00	69.08	8.81	100.00	80.91	10.71	91.66
	600	63.00	13.82	100.00	72.11	8.54	100.00	75.95	4.89	100.00
Mean		**72.20**	**6.27**	**98.33**	**76.93**	**4.86**	**96.88**	**82.93**	**3.95**	**98.61**

Table 6 Comparison of Methods for Different Reduction Percentages without unfeasible and time limit instances on Unbalanced training

Red(%)		20			30			40		
Met	Box	%AT	%SD	%OPT	%AT	%SD	%OPT	%AT	%SD	%OPT
RF	100	37.10	1.25	100.00	54.75	1.30	100.00	74.33	0.92	100.00
	200	43.05	3.10	100.00	50.06	1.94	100.00	59.42	2.79	100.00
	300	17.44	3.70	100.00	38.28	2.94	100.00	60.15	2.60	100.00
	400	34.26	2.50	100.00	61.51	2.75	100.00	52.91	5.60	100.00
	500	37.26	11.20	100.00	57.64	11.10	100.00	69.00	9.90	100.00
	600	40.45	8.61	100.00	32.32	6.75	100.00	65.42	7.40	100.00
Mean		**34.93**	**5.73**	**100.00**	**49.43**	**4.80**	**100.00**	**63.54**	**4.87**	**100.00**
SVR	100	44.63	1.30	100.00	44.85	1.15	100.00	68.96	0.96	100.00
	200	33.50	3.50	100.00	44.87	3.60	100.00	69.34	3.34	100.00
	300	37.37	3.33	100.00	55.35	3.03	100.00	55.23	4.03	88.00
	400	43.48	4.70	100.00	64.20	4.90	100.00	79.10	2.84	100.00
	500	35.46	8.50	100.00	55.67	8.60	100.00	67.24	13.50	100.00
	600	35.16	12.13	100.00	57.52	10.57	100.00	54.34	6.50	100.00
Mean		**38.27**	**5.91**	**100.00**	**53.74**	**5.64**	**100.00**	**65.70**	**5.86**	**98.00**
HYB	100	43.11	1.05	100.00	49.89	1.10	100.00	73.20	1.11	100.00
	200	22.27	2.45	100.00	43.74	2.61	100.00	66.57	1.81	100.00
	300	29.92	2.86	100.00	34.64	2.52	100.00	68.65	1.43	88.00
	400	33.51	3.68	100.00	53.29	3.15	100.00	76.35	5.80	100.00
	500	25.12	13.00	100.00	44.96	13.15	100.00	68.79	13.70	100.00
	600	39.98	10.55	100.00	59.41	10.44	100.00	52.24	10.53	100.00
Mean		**32.32**	**5.93**	**100.00**	**47.99**	**5.83**	**100.00**	**67.30**	**5.73**	**98.00**
Red(%)		50			60			70		
Met	Box	%AT	%SD	%OPT	%AT	%SD	%OPT	%AT	%SD	%OPT
RF	100	86.68	1.18	100.00	93.35	2.31	100.00	0	0	0
	200	71.43	4.94	100.00	86.51	2.87	100.00	87.41	1.55	100.00
	300	73.97	5.33	100.00	82.65	2.33	100.00	94.06	1.57	83.00

Table 6 continued

Red(%)		50			60			70		
Met	Box	%AT	%SD	%OPT	%AT	%SD	%OPT	%AT	%SD	%OPT
	400	79.06	4.68	100.00	82.11	4.48	100.00	88.65	2.28	88.00
	500	67.66	10.71	100.00	45.01	10.18	100.00	73.34	7.91	88.00
	600	34.07	9.84	100.00	63.89	9.11	100.00	63.92	4.11	85.00
Mean		**68.81**	**6.78**	**100.00**	**75.59**	**5.88**	**100.00**	**81.48**	**2.90**	**90.67**
SVR	100	84.76	1.92	100.00	91.66	2.78	100.00	96.40	1.39	100.00
	200	67.20	3.42	100.00	90.53	2.05	90.00	90.40	2.94	100.00
	300	75.78	5.04	94.11	-3.26	2.64	100.00	90.44	2.03	100.00
	400	57.59	4.08	100.00	82.76	3.31	100.00	88.31	2.24	90.90
	500	78.65	8.70	100.00	67.07	14.48	80.00	70.50	9.12	90.00
	600	75.35	10.06	100.00	77.12	10.48	41.67	11.54	10.79	80.00
Mean		**73.22**	**5.87**	**99.35**	**67.65**	**5.96**	**85.45**	**74.60**	**4.42**	**93.15**
HYB	100	88.55	2.45	100.00	93.43	1.68	100.00	0	0	0
	200	75.45	1.47	100.00	89.85	2.62	90.90	93.35	2.11	100.00
	300	79.25	3.36	100.00	86.73	2.93	100.00	90.47	2.58	85.71
	400	70.72	3.83	100.00	85.16	4.57	100.00	89.42	4.41	90.00
	500	74.82	12.03	100.00	68.71	14.81	100.00	73.94	10.71	90.00
	600	76.26	10.82	100.00	78.17	18.54	41.67	12.91	14.89	80.00
Mean		**77.50**	**5.99**	**100.00**	**83.84**	**7.69**	**88.93**	**72.02**	**5.78**	**90.62**

mance across both balanced and unbalanced data sets. This indicates that SVR tends to excel when the reduction percentages are at these lower levels.

Interestingly, the HYB method, on average, closely mirrors the performance trends exhibited by SVR and, while not always surpassing SVR, it consistently follows a similar trajectory across various reduction percentages and data set types, hinting at its stability and adaptability across different scenarios.

The connection between how much data we reduce, how well the methods work, and whether our data are balanced or not is pretty tricky. Different methods do better in different situations, therefore it is tough to say one method is always the best.

In Tables 7 and 8 we present the average time reduction percentage for the three methods considering all instances, optimal, feasible and unfeasible after the reduction on the layer sets. The percentage reduction is calculated with respect to the average time taken for the respective method with no layer reduction. The table presents average time reduction percentages for different box numbers and reduction levels, ranging from 20% to 70%. Each column represents a reduction level, while rows indicate box numbers from 100 to 600. The #unf columns correspond to the number of unfeasible instances.

It is evident from the tables that a greater reduction in the layer pool corresponds to a decrease in the number of solved instances. However, there is still a reduction in the percentage of solution times. This phenomenon occurs because the instances

Table 7 Comparison of Methods for Different Reduction Percentages with Balanced Training. Each row summarizes results obtained from solving 20 instances per dimension

Red(%)		20				30				40			
Met	Box	%AT	%SD	#unf	#TL	%AT	%SD	#unf	#TL	%AT	%SD	#unf	#TL
RF	100	33.37	1.12	0	0	46.74	1.12	1	0	52.63	0.96	4	0
	200	-3.53	2.98	1	0	-2.62	3.14	1	0	50.88	3.48	2	0
	300	20.40	3.51	0	0	23.65	3.67	0	0	51.45	4.81	2	0
	400	29.69	2.32	0	1	47.80	2.35	0	0	56.39	5.50	0	0
	500	24.48	10.98	0	0	36.01	10.80	1	0	58.90	9.61	1	0
	600	27.79	10.02	0	0	48.30	10.01	1	0	51.04	10.34	2	0
Mean		**22.03**	**5.16**	**0.17**	**0.17**	**33.31**	**5.18**	**0.67**	**0**	**53.55**	**5.78**	**1.83**	**0**
SVR	100	40.24	1.09	0	0	55.25	1.06	0	0	56.57	1.17	4	0
	200	21.09	3.21	0	0	49.80	3.34	1	0	65.11	3.92	1	0
	300	27.35	3.69	1	0	35.93	3.62	1	0	26.03	4.53	2	0
	400	36.71	4.72	0	0	59.80	4.66	0	0	61.63	2.95	0	0
	500	33.61	8.28	0	0	53.68	8.28	0	0	61.24	13.19	0	0
	600	32.75	16.02	1	0	52.50	15.96	1	0	61.82	16.93	1	0
Mean		**31.96**	**6.17**	**0.33**	**0**	**51.16**	**6.16**	**0.50**	**0**	**55.40**	**7.12**	**1.33**	**0**
HYB	100	40.93	0.91	0	0	53.94	0.92	0	0	56.37	0.91	4	0
	200	24.60	2.29	0	0	15.08	2.33	1	0	40.13	1.72	2	0
	300	10.55	2.81	1	0	21.73	2.77	1	0	55.74	4.59	1	0
	400	45.49	3.08	0	0	53.16	3.08	0	0	47.99	5.21	0	0
	500	32.25	12.71	0	0	58.65	12.72	0	0	62.62	13.39	0	0
	600	22.15	16.19	1	1	49.55	16.19	1	0	59.02	16.94	1	0

Table 7 continued

Red(%)		20				30				40			
Met	Box	%AT	%SD	#unf	#TL	%AT	%SD	#unf	#TL	%AT	%SD	#unf	#TL
Mean		**29.33**	**6.33**	**0.33**	**0.17**	**42.02**	**6.34**	**0.50**	**0**	**53.65**	**7.13**	**1.33**	**0**
Red(%)		50				60				70			
Met	Box	%AT	%SD	#unf	#TL	%AT	%SD	#unf	#TL	%AT	%SD	#unf	#TL
RF	100	34.23	1.25	10	0	19.49	1.02	15	0	3.47	0.89	19	0
	200	59.87	3.90	4	0	50.75	2.34	7	0	6.62	1.72	15	1
	300	51.04	5.10	2	1	41.62	4.23	7	0	14.51	2.67	16	0
	400	69.15	5.89	2	0	61.95	4.56	5	0	38.29	3.71	11	0
	500	-24.56	11.02	1	1	43.43	9.34	2	2	9.30	8.78	10	2
	600	32.37	9.78	5	0	43.06	8.95	8	0	19.65	7.62	13	1
Mean		**37.02**	**6.16**	**4.00**	**0.33**	**43.38**	**4.40**	**7.33**	**0.33**	**15.31**	**4.23**	**14.00**	**0.67**
SVR	100	43.62	1.21	9	0	9.22	1.01	14	0	3.78	0.98	19	0
	200	72.32	4.01	2	0	21.42	3.34	13	0	13.43	2.98	17	0
	300	44.92	4.67	3	0	17.80	4.03	5	2	23.95	3.52	14	0
	400	60.72	3.34	1	0	61.71	3.11	3	1	25.31	3.02	11	1
	500	50.11	12.01	1	1	60.25	11.34	3	0	47.14	10.78	8	0
	600	57.85	15.42	2	0	38.68	13.78	8	0	22.25	12.23	14	0

Table 7 continued

Red(%)		50				60				70			
Met	Box	%AT	%SD	#unf	#TL	%AT	%SD	#unf	#TL	%AT	%SD	#unf	#TL
Mean		**54.92**	**6.78**	**3.00**	**0.17**	**34.85**	**6.10**	**7.67**	**0.50**	**22.14**	**5.92**	**13.83**	**0.17**
HYB	100	35.95	1.01	10	0	15.95	0.89	16	0	1.67	0.81	19	0
	200	56.32	2.11	5	0	34.31	2.78	9	0	16.98	2.43	16	0
	300	40.54	4.03	3	0	16.68	3.67	5	2	23.95	3.56	14	0
	400	66.56	5.89	1	0	59.46	5.67	3	0	27.07	4.34	11	1
	500	50.38	12.43	1	1	58.36	11.78	3	0	48.25	10.89	8	0
	600	56.62	15.45	2	0	41.13	14.78	8	0	22.46	13.34	14	0
Mean		**51.06**	**6.48**	**3.67**	**0.17**	**37.65**	**6.13**	**7.33**	**0.33**	**23.40**	**5.89**	**13.67**	**0.17**

Table 8 Comparison of methods for different reduction percentages with unbalanced training. Each row summarizes results obtained from solving 20 instances per dimension

Red(%)		20				30				40			
Met	Box	%AT	%SD	#unf	#TL	%AT	%SD	#unf	#TL	%AT	%SD	#unf	#TL
RF	100	37.10	1.25	0	0	54.75	1.30	0	0	60.29	1.45	3	0
	200	43.05	3.10	0	0	29.54	3.45	1	0	54.69	3.75	1	0
	300	17.44	3.70	0	0	36.18	4.00	1	0	33.24	5.00	2	1
	400	34.26	2.50	0	0	61.51	2.75	0	0	52.91	5.60	0	0
	500	37.26	11.20	0	0	57.64	11.10	0	0	69.00	9.90	0	0
	600	37.03	10.50	0	1	28.92	10.45	1	0	53.87	10.80	2	1
Mean		**34.36**	**5.38**	**0.00**	**0.17**	**44.76**	**5.51**	**0.50**	**0.00**	**54.00**	**6.08**	**1.33**	**0.33**
SVR	100	44.63	1.30	0	0	44.85	1.15	0	0	60.83	1.25	2	0
	200	33.50	3.50	0	0	44.87	3.60	0	0	50.20	4.05	1	0
	300	33.94	4.00	0	1	52.45	3.85	1	0	-80.22	4.75	2	1
	400	39.93	5.00	0	1	64.20	4.90	0	0	73.08	3.15	0	1
	500	35.46	8.50	0	0	55.67	8.60	0	0	67.24	13.50	0	0
	600	30.09	16.25	1	1	54.60	16.10	1	0	51.57	17.15	1	0

Table 8 continued

Red(%)		20				30				40			
Met	Box	%AT	%SD	#unf	#TL	%AT	%SD	#unf	#TL	%AT	%SD	#unf	#TL
Mean		**36.26**	**6.42**	**0.17**	**0.50**	**52.77**	**6.37**	**0.33**	**0.00**	**37.12**	**7.31**	**1.00**	**0.33**
HYB	100	43.11	1.05	0	0	49.89	1.10	0	0	59.33	1.00	3	0
	200	22.27	2.45	0	0	25.79	2.55	1	0	49.07	1.95	1	0
	300	22.14	3.00	0	1	32.72	2.95	1	0	-65.29	4.75	1	1
	400	30.46	3.20	0	1	53.29	3.15	0	0	70.46	5.40	0	1
	500	25.12	13.00	0	0	44.96	13.15	0	0	68.79	13.70	0	0
	600	37.82	16.35	1	0	56.38	16.50	1	0	49.58	17.25	1	0
Mean		**30.15**	**6.51**	**0.17**	**0.33**	**43.84**	**6.56**	**0.50**	**0.00**	**38.66**	**7.34**	**1.00**	**0.33**
Red(%)		50				60				70			
Met	Box	%AT	%SD	#unf	#TL	%AT	%SD	#unf	#TL	%AT	%SD	#unf	#TL
RF	100	45.78	1.35	9	0	6.10	1.20	16	0	-0.72	1.00	20	0
	200	58.78	4.10	3	0	46.09	2.50	9	0	11.13	1.85	17	0
	300	37.74	5.35	4	1	56.22	4.50	6	0	27.12	2.75	14	0
	400	79.06	6.05	0	0	73.62	4.75	2	0	6.01	3.85	10	1
	500	57.15	11.35	1	1	40.45	9.55	2	0	32.85	8.95	11	0
	600	21.30	10.05	5	1	35.75	9.00	7	1	21.97	7.85	13	0
Mean		**49.97**	**6.04**	**3.67**	**0.50**	**43.04**	**4.42**	**7.00**	**0.17**	**16.39**	**4.23**	**14.17**	**0.17**
SVR	100	23.71	1.30	11	0	24.66	1.10	14	0	-0.96	1.05	17	1
	200	49.71	4.10	5	0	49.01	3.40	9	0	7.18	3.25	18	0
	300	-62.69	4.80	2	1	-26.94	4.10	5	1	25.63	3.75	14	0
	400	51.78	3.50	2	0	59.61	3.25	4	1	46.43	3.10	8	1
	500	59.02	12.25	0	1	37.87	11.55	3	1	34.64	11.00	10	0
	600	59.73	15.60	4	0	-157.81	14.05	7	1	-2.06	12.50	15	0

Table 8 continued

Red(%)		50				60				70			
Met	Box	%AT	%SD	#unf	#TL	%AT	%SD	#unf	#TL	%AT	%SD	#unf	#TL
Mean		**30.21**	**6.59**	**4.00**	**0.50**	**-2.27**	**6.41**	**7.00**	**0.67**	**18.14**	**5.61**	**13.67**	**0.33**
HYB	100	42.46	1.20	10	0	20.55	1.10	15	0	-2.51	0.95	20	0
	200	52.76	2.35	4	0	26.17	2.90	7	2	13.05	2.50	17	0
	300	46.01	4.30	3	1	53.53	3.85	7	0	28.73	3.75	13	0
	400	63.62	6.05	2	0	67.72	5.80	4	0	33.89	4.50	8	2
	500	55.38	12.65	0	1	39.16	12.00	3	1	36.39	11.25	10	0
	600	60.40	16.00	4	0	-157.14	15.25	7	1	-1.61	13.50	15	0
Mean		**53.44**	**7.09**	**3.83**	**0.50**	**8.33**	**6.82**	**7.17**	**0.67**	**17.99**	**6.07**	**13.83**	**0.33**

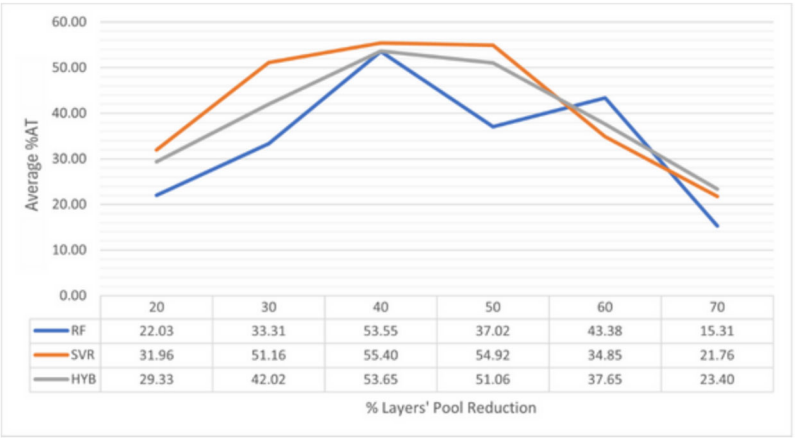

Fig. 1 Averages of %AT for Balanced data set for RF, SVR and HYB methods

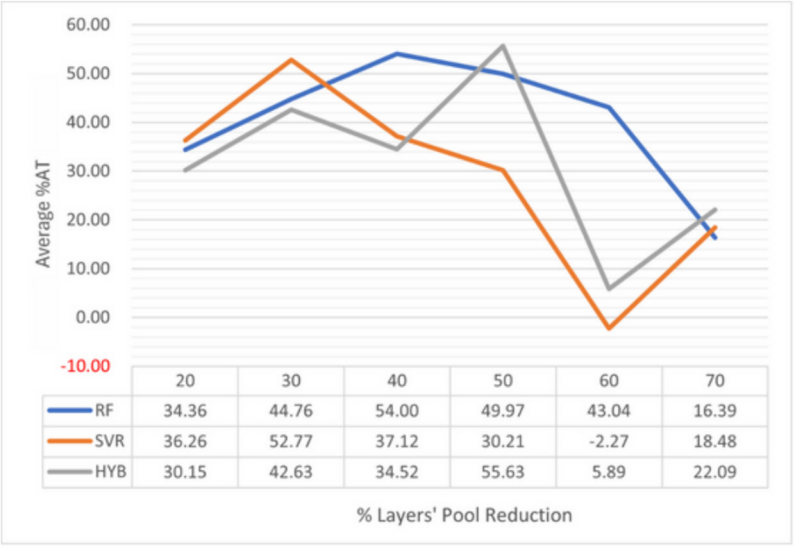

Fig. 2 Averages of %AT for Unbalanced data set for RF, SVR and HYB methods

that are solved contribute significantly to improving solution times, and the remaining unsolved instances are generally demonstrated as unfeasible within a short time.

Notably, almost all %AT mean values across the reduction percentages for each method are positive, except for the -2.27 in Table 8. This indicates that, on average, the methods exhibit improved solution times for reduced instances compared to their original counterparts.

Overall, the positive %AT values across reduction percentages imply that all methods effectively benefit from instance reduction, showcasing improved solution times.

In the Figures 1 and 2, we have depicted the averages of %AT presented in the Tables 7 and 8. Different colors correspond to different methods. On the x-axis, we have the percentage reductions of layer pools, and on the y-axis, the averages of %AT.

As can be inferred from the graphs, the method that performs the best on the balanced data set is SVR. On the other hand, in the case of the unbalanced data set, RF generally outperforms in terms of time reduction.

Additionally, from both graphs, we deduce that the method is effective for reductions up to 50% of the data set, as reducing the layer pool by 60% and 70% results in a lesser reduction in ILP solution times. This is due to the high probability of rendering instances unfeasible, a consequence of excessive reduction in the layer data set.

6 Conclusions and future works

In this paper, we have introduced a matheuristic methodology for solving the Distributor's Pallet Loading problem with constraints derived from real-world applications. The proposed approach uses ML methodologies to speed up the solution of the Integer Linear Programming model of the problem. In particular, by adding predictors trained with ML techniques to the procedure, a powerful tool can be obtained to reduce the possible combinations to be explored. We demonstrate, through extensive experiments, the effectiveness of our proposal, highlighting the benefits in terms of computational time through the use of ML algorithms.

Various issues remain open for validating the methods we have used. It is possible to experiment with different ML and Deep Learning algorithms, as well as different solutions in terms of predicting the importance score of each layer for solving instances. To shorten the data set creation time used for the training stage, it is possible to decrease the number of solved instances, also reducing the data set size and consequently the time needed for the training stage. Linked to data imbalance, a new idea that can be used is to create new data from the minority class with generative models. Finally, in order to reduce the number of unfeasible instances caused by variable reduction, different strategies can be considered, such as more conservative filtering thresholds, adaptive reduction heuristics, or post-reduction feasibility checks integrated into the learning process.

Acknowledgements L. Zanni and G. Franchini work is partly funded by "Gruppo Nazionale per il Calcolo Scientifico (GNCS INdAM)". L. Zanni work is partly funded by the - Partenariato Esteso PE00000013 - "FAIR", funded by the European Commission under the NextGeneration EU programme, PNRR - M4C2 - Investimento 1.3. - Italian MUR through the PRIN 2022 Project "Numerical Optimization with Adaptive Accuracy and Applications to Machine Learning", project code: 2022N3ZNAX (CUP E53D23007700006), under the National Recovery and Resilience Plan (PNRR), Italy, Mission 04 Component 2 Investment 1.1 funded by the European Commission - NextGeneration EU programme.

Funding Open access funding provided by Università degli Studi di Modena e Reggio Emilia within the CRUI-CARE Agreement.

Declarations

Conflict of Interest No potential conflict of interest was reported by the author(s).

References

Ali, S., Ramos, A., Carravilla, M., Oliveira, J.: On-line three-dimensional packing problems: A review of off-line and on-line solution approaches. Computers & Industrial Engineering **168**, 108122 (2022)

Alonso Mart nez, M.T., Alvarez-Valdes, R., Parre o, F., Tamarit, J.: Algorithms for pallet building and truck loading in an interdepot transportation problem. Mathematical Problems in Engineering **2016**, 1–11 (2016)

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

Ancora, G., Palli, G., Melchiorri, C.: A hybrid genetic algorithm for pallet loading in real-world applications. IFAC-PapersOnLine **53**, 10006–10010 (2020)

Aylak, B.L., İnce, M., Oral, O., Süer, G., Almasarwah, N., Singh, M., Salah, B.: Application of machine learning methods for pallet loading problem. Appl. Sci. **11**(18), 8304 (2021)

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W., Vance, P.H.: Branch-and-price: Column generation for solving huge integer programs. Oper. Res. **46**(3), 316–329 (1998)

Barrett, T., Clements, W., Foerster, J., Lvovsky, A.: Exploratory combinatorial optimization with reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 3243–3250 (2020)

Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S.: Neural combinatorial optimization with reinforcement learning. arXiv:1611.09940 (2016)

Bengio, Y., Lodi, A., Prouvost, A.: Machine learning for combinatorial optimization: a methodological tour d horizon. Eur. J. Oper. Res. **290**(2), 405–421 (2021)

Bischoff, E.E., Janetz, F., Ratcliff, M.S.W.: Loading pallets with non-identical items. Eur. J. Oper. Res. **84**, 681–692 (1995)

Bischoff, E., Ratcliff, M.: Loading multiple pallets. Journal of The Operational Research Society - J OPER RES SOC **46**, 1322–1336 (1995)

Bonettini, S., Franchini, G., Pezzi, D., Prato, M.: Explainable bilevel optimization: An application to the helsinki deblur challenge. Inverse Problems in Imaging (2023)

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

Breiman, L.: Bagging predictors. Machine Learning, 123–140 (1996)

Breiman, L.: Random forests. Machine Learning, 5–32 (2001)

Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 121–167 (1998)

Chan, P.K., Stolfo, S.: Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In: Knowledge Discovery and Data Mining (1998)

Crainic, T.G., Perboli, G., Tadei, R.: Extreme point-based heuristics for three-dimensional bin packing. INFORMS J. Comput. **20**(3), 368–384 (2008)

Dell'Amico, M., Magnani, M.: Solving a real-life distributor's pallet loading problem. Mathematical and Computational Applications **26**, 53 (2021)

Dowsland, K., Dowsland, W.: Packing problems. Eur. J. Oper. Res. **56**, 2–14 (1992)

Egeblad, J., Garavelli, A., Lisi, S., Pisinger, D.: Heuristics for container loading of furniture. Eur. J. Oper. Res. **200**, 881–892 (2010)

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

Erbayrak, S., Özkir, V., Yıldırım, U.: Multi-objective 3d bin packing problem with load balance and product family concerns. Computers & Industrial Engineering **159**, 107518 (2021)
Franchini, G., Ruggiero, V., Porta, F., Zanni, L.: Neural architecture search via standard machine learning methodologies. Mathematics in Engineering, 1–21 (2023)
Furian, N., O Sullivan, M., Walker, C., Çela, E.: A machine learning-based branch and price algorithm for a sampled vehicle routing problem. Or Spectrum **43(3)**, 693–732 (2021)
Gentili, E., Franchini, G., Zese, R., Alberti, M., Ferrara, M., Domenicano, I., Grassi, L.: Machine learning from real data: A mental health registry case study. Computer Methods and Programs in Biomedicine Update **5**, 100132 (2024)
Gerych, W., Agu, E., Rundensteiner, E.: Classifying depression in imbalanced datasets using an autoencoder-based anomaly detection approach. In: 2019 IEEE 13th International Conference on Semantic Computing (ICSC), pp. 124–127 (2019)
Gunawardena, K., Wijayanayake, A., Kavirathna, C.: Solve manufacturer's pallet loading problem (mplp) with practical warehouse constraints. In: 2021 IEEE 6th International Conference on Information Technology Research (ICITR), pp. 1–6 (2021)
Gzara, F., Elhedhli, S., Yildiz, B.: The pallet loading problem: Three-dimensional bin packing with practical constraints. European Journal of Operational Research **287** (2020)
Harrath, Y.: A three-stage layer-based heuristic to solve the 3d bin-packing problem under balancing constraint. Journal of King Saud University - Computer and Information Sciences **34** (2021)
He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. **21**(9), 1263–1284 (2009)
Iori, M., Locatelli, M., Moreira, M., Silveira, T.: Reactive GRASP-Based Algorithm for Pallet Building Problem with Visibility and Contiguity Constraints, pp. 651–665 (2020)
Khalilia, M., Chakraborty, S., Popescu, M.: Predicting disease risks from highly imbalanced data using random forest. BMC Med. Inform. Decis. Mak. **11**(1), 1–13 (2011)
Krawczyk, B.: Learning from imbalanced data: open challenges and future directions. Prog. Artif. Intell. **5**(4), 221–232 (2016)
Layeb, S.B., Omri, O.: Solving the pallet loading problem with deep reinforcement learning, pp. 807–825. Springer (2024)
Lijun, W., Zhang, D., Chen, Q.: A least wasted first heuristic algorithm for the rectangular packing problem. Computers & Operations Research **36**, 1608–1614 (2009)
Lodi, A., Martello, S., Vigo, D.: Heuristic algorithms for the three-dimensional bin packing problem. Eur. J. Oper. Res. **141**(2), 410–420 (2002)
Lodi, A., Zarpellon, G.: On learning and branching: a survey. TOP **25**, 207–236 (2017)
Magnani, M., Franchini, G., Dell'Amico, M., Zanni, L. (2025). A Machine Learning Approach to Speed up the Solution of the Distributor's Pallet Loading Problem. In: Sergeyev, Y.D., Kvasov, D.E., Astorino, A. (eds)Numerical Computations: Theory and Algorithms. NUMTA 2023. Lecture Notes in Computer Science, vol 14476. Springer, Cham
Mahvash, B., Awasthi, A., Chauhan, S.: A column generation-based heuristic for the three-dimensional bin packing problem with rotation. Journal of the Operational Research Society **69**(1), 78–90 (2018)
Majid, A., Ali, S., Iqbal, M., Kausar, N.: Prediction of human breast and colon cancers from imbalanced data using nearest neighbor and support vector machines. Comput. Methods Programs Biomed. **113**(3), 792–808 (2014)
Mamaghan, M.K., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A.M., Talbi, E.-G.: Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. Eur. J. Oper. Res. **296**, 393–422 (2021)
Martello, S., Pisinger, D., Vigo, D.: The three-dimensional bin packing problem. Oper. Res. **48**(2), 256–267 (2000)
Masola, A., Capodieci, N., Rouxel, B., Franchini, G., Cavicchioli, R.: Machine learning techniques for understanding and predicting memory interference in cpu-gpu embedded systems. 2023 IEEE 29th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 147–156 (2023)
Mazyavkina, N., Sviridov, S., Ivanov, S., Burnaev, E.: Reinforcement learning for combinatorial optimization: A survey. Computers & Operations Research **134**, 105400 (2021)
Morabit, M., Desaulniers, G., Lodi, A.: Machine-learning-based column selection for column generation. Transp. Sci. **55**(4), 815–831 (2021)

Morabito, R., Morales, S.R.: A simple and effective recursive procedure for the manufacturer's pallet loading problem. Journal of the Operational Research Society **49**, 819–828 (1998)
Moura, A., Oliveira, J.: A grasp approach to the container-loading problem. Intelligent Systems **20**, 50–57 (2005)
Paquay, C., Limbourg, S., Schyns, M., Oliveira, J.F.: Mip-based constructive heuristics for the three-dimensional bin packing problem with transportation constraints. Int. J. Prod. Res. **56**(4), 1581–1592 (2018)
Piyachayawat, T., Mungwattana, A.: A hybrid algorithm application for the multi-size pallet loading problem case study: Lamp and lighting factory. In: 2017 IEEE 4th International Conference of Industrial Engineering and Applications (ICIEA), pp. 100–105 (2017)
Ranck, R., Yanasse, H., Morabito, R., Junqueira, L.: A hybrid approach for a multi-compartment container loading problem. Expert Systems with Applications **137** (2019)
Saraiva, R., Nepomuceno, N., Pinheiro, P.: A layer-building algorithm for the three-dimensional multiple bin packing problem: a case study in an automotive company. IFAC-PapersOnLine **48**, 490–495 (2015)
Scheithauer, G., Terno, J.: A Heuristic Approach for Solving the Multi-Pallet Packing Problem (1999)
Silva, E., Oliveira, J., W scher, G.: The pallet loading problem: A review of solution methods and computational experiments. International Transactions in Operational Research **23** (2014)
Toffolo, T., Esprit, E., Wauters, T., Vanden Berghe, G.: A two-dimensional heuristic decomposition approach to a three-dimensional multiple container loading problem. European Journal of Operational Research **257** (2016)
Tresca, G., Cavone, G., Carli, R., Cerviotti, A., Dotoli, M.: Automating bin packing: A layer building matheuristics for cost effective logistics. IEEE Trans. Autom. Sci. Eng. **19**, 1–15 (2022)
Wongvorachan, T., He, S., Bulut, O.: A comparison of undersampling, oversampling, and smote methods for dealing with imbalanced classification in educational data mining. Information **14**(1) (2023)
Yan, R., Liu, Y., Jin, R., Hauptmann, A.: On predicting rare classes with svm ensembles in scene classification. Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing **3**, 21–24 (6–10 April 2003)
Zhao, X., Bennell, J., Bektaş, T., Dowsland, K.: A comparative review of 3d container loading algorithms. International Transactions in Operational Research **23** (2014)