Operations Research Perspectives 14 (2025) 100329

Contents lists available at ScienceDirect

Operations Research Perspectives

journal homepage: www.elsevier.com/locate/orp

Physical question, virtual answer: Optimized real-time physical simulations and physics-informed learning approaches for cargo loading stability

Philipp Gabriel Mazur *, Johannes Werner Melsbach, Detlef Schoder

Cologne Institute for Information Systems, Pohligstr. 1, Cologne, 50969, Germany

ARTICLE INFO

Keywords:
Static stability
Loading stability
Physical simulation
Physics-informed learning
Pallet loading problem

ABSTRACT

Cargo stability is a crucial requirement for safe cargo loading and transport. Current state-of-the-art approaches simplify cargo loading to an idealized static problem and employ geometric- and force-based approaches. In this research, we model cargo loading stability as a dynamic problem and propose two approaches. We use (a) a physical simulation using a real-time physics engine fitted for cargo loading and (b) a physics-informed learning model trained on cargo loading data. Both approaches are capable of handling dynamic physical behavior, either explicitly through simulation, or implicitly through training a recurrent neural network on physically-biased sequential cargo loading data. Given our two objectives of maximal accuracy and minimal runtime, our benchmarking results show that our approaches can outperform current state-of-the-art static stability methods in terms of accuracy depending on the complexity scenario, but consume more runtime.

1. Introduction

Transportation – the physical movement of goods and people – often requires consideration of physical dynamics. Examples of physical dynamics include oil flows (fluids), container stacking for port logistics (rigid bodies), and bicycle flows (particles). One physical dynamics problem that affects transportation safety is the cargo stability when loading, unloading, and transporting cargo on pallets or in containers [1]. Unstable cargo in loading, unloading, and transport situations is a safety issue and may lead to damaged goods, injuries of personnel [2–4], and -in the case of stability problems during transport- even environmental pollution [5]. In this study, we focus on cargo stability in loading situations as part of the pallet loading problem (PLP). This PLP can be categorized as a distributor's PLP [6], involving the loading of three-dimensional cargo items (boxes) onto a single pallet without lateral support. The assortment of items is highly heterogeneous and the pallets have a cuboid contour that defines the entire permissible loading space. We assume that the solutions (or: layouts) to the PLP are given and are the object of a stability evaluation. Every layout consists of both three-dimensional item loading positions and sequences.

Past work has widely modeled the stability during cargo loading as a static problem (static or vertical stability), whereas the transport of cargo is modeled as a dynamic problem (dynamic or horizontal stability) [1]. In literature, there are several methods to evaluate static stability, for instance, geometric-based methods, such as full base or partial base support [1], or idealized static force-based approaches, such as static mechanical equilibrium based on known physical laws [7,8]. However, these static stability methods, although they provide good approximations of cargo loading stability and consume little runtime, focus only on snapshots of the cargo loading process and neglect cargo loading dynamics. Consequently, when testing these static stability methods against dynamic digital simulations of cargo loading, they tend to overestimate or underestimate cargo loading stability [9]. As a novel concept, cargo loading stability is defined as modeling the dynamic process of safely loading items on a pallet. It distinguishes from static stability by enforcing that item placements lead to no *considerable* spatial position and orientation changes [9], so it makes it necessary to capture and calculate physical dynamics of cargo during loading situations.

Calculating physical dynamics is a challenging and time-consuming operation, and it must be fast (i.e., low runtime) and accurate to be utilized within PLP optimization algorithms [10]. There are multiple computational physics methods with high potential to accurately model physical dynamics, such as real-time physics engines or hybrid physics-informed learning. Both methods are dedicated to model physical dynamics, either through general-purpose simulation or by training a machine learning algorithm, preferably a Long-Short Term Memory (LSTM) recurrent neural network (RNN) capable of handling sequential data, on massive amounts of data. Physics engines have been successfully applied within games and animations [11], but increasingly also for research problems such as design and structural analysis [12],

* Corresponding author.
E-mail addresses: mazur@wim.uni-koeln.de (P.G. Mazur), melsbach@wim.uni-koeln.de (J.W. Melsbach), schoder@wim.uni-koeln.de (D. Schoder).

https://doi.org/10.1016/j.orp.2025.100329
Received 13 August 2024; Received in revised form 8 February 2025; Accepted 9 February 2025
Available online 16 February 2025

mechanical product design [13], digital twins, and material flows [14]. LSTMs have been successfully applied to intuitive physics tasks [15].

Several studies treat cargo loading as a dynamic problem and propose simulations [16,17]. However, they lack a comprehensive problem statement, a description of the algorithm, and optimization of its hyperparameters towards prediction accuracy and runtime. In this study, we fill this gap and propose two approaches that build on physical simulations using real-time physics engines and physics-informed learning to solve the cargo loading stability problem. We optimize our approaches for high prediction accuracy and low runtime and evaluate our approaches on a publicly available PLP loading stability benchmark dataset.

The remainder of this paper is structured as follows. The following section introduces the PLP, cargo loading stability, and current computational physics modeling approaches that are able to model physical dynamics. We then formulate loading stability as our specific physical dynamics problem. The following section explains our solution approaches by specifying two novel physical simulation-based and physics-informed learning methods. We report on the experimental setup, present our results, and end with a discussion of the implications of this study and some conclusions.

2. Literature review

2.1. Pallet loading problems and static stability

Pallet loading problems are combinatorial optimization problems that belong to the Cutting & Packing problem class [1,18]. The goal is to arrange orthogonally a set of non-overlapping smaller items (cargo) on or within larger objects that are being loaded for transport in a truck, vessel, or aircraft. Loading efficiency typically means the cargo arrangement (layout) utilizes as much space as possible [1]. These large objects can be containers or pallets used to simplify cargo handling and streamline operations [19,20]. Although our focus is on pallet loading, given the association between pallet and container loading, we also consider research findings from the field of container loading. In this study, the PLP can be classified as a (single) distributor's PLP [6] that entails the loading of a three-dimensional, highly heterogeneous assortment of cargo items (boxes) onto a single pallet with no lateral support.

Solutions to the PLP must meet a set of hard and soft constraints to be practically relevant. Constraints are restrictions of the solution space and model economic and physical requirements, such as priority cargo, maximum weight restrictions, balancing, and load-bearing capacities [1,21]. Among these, stability is very important, particularly relevant for transportation safety [2,3], and highly significant for practical usage [22], since instability can cause damage to cargo, workers, and pallets during loading operations [8,23]. A layout is deemed statically stable when all items can endure the force of gravity acting upon them, ensuring that they maintain their initial positions without slipping, rotating, tipping, or falling [24].

At present, three physics-based approaches to measuring static stability can be categorized: geometry-based approaches; force-based approaches; and dynamic simulations. Geometry-based approaches – such as full base support (FBS) or partial base support (PBS) – aim to ensure that a minimum portion of the lower surface of each cargo item is adequately supported, whether by the top faces of underlying items or the pallet. We call these approaches geometry-based because their decision criterion is limited to the geometries of the lower and upper sides of the items. The support factor value α indicates the percentage of an item's lower face that is supported. Multiple recent studies employ PBS with α beginning at different levels: 0.7 [25]; 0.75 [7,26,27]; and 0.8 [28,29]. In a recent comparison study of two-dimensional packing problems, an α-value of 42% is found to be sufficient to achieve static stability with a probability of over 90% [4].

Force-based approaches – such as static mechanical equilibrium (SME) – opt to validate whether there is a static mechanical equilibrium of cargo. Achieving equilibrium in this context ensures that the sum of all forces acting on the cargo item is zero and that the sum of the moments of all forces acting on each part of each item also adds up to zero [8]. These approaches are closer approximations to the real world because they make use of more information (geometry and forces), but they represent only snapshots of cargo arrangements. Both geometry- and force-based approaches assume static problems and neglect the dynamics that occur during loading operations, such as collision of items or side support.

Dynamic physical simulations (PS) fill this gap by considering dynamic events such as collisions. Dynamic simulations have been predominantly applied to dynamic stability problems [16,30–33], e.g., to simulate the impact of vehicle braking and acceleration [30]. For cargo loading stability, only a few studies exist that employ dynamic simulations, mostly using real-time physics engines [16,17,34]. With these simulations, researchers approximate the real-world, step-by-step cargo loading process by simulating some amount of time and measuring the spatial displacements of the rigid-body-modeled cargo during a time period. In terms of workflow and algorithm, different simulation loading algorithms are possible, which simulate the entire (fully built-up) layout or consider cargo loading item-by-item [17]. To speed up calculations, the loading algorithm can be extended for parallelization using general-purpose compute capabilities of modern graphics processing units [34]. As measured simulation outcome, usually the amount of spatial displacement of an item is used [16,17,34]. The simulations are carried out using real-time physics engines that can achieve result accuracy comparable to specialized, dedicated physical multibody simulators [32], which are the "gold standard" for multibody simulations. Among the major benefits of real-time physics engines are that they allow for modeling more types of cargo (e.g., boxes, convex shapes defined by a list of points, and compound shapes), include physical meta-information (e.g., uneven mass distributions, restitution, and friction), and make it possible to work in a way that is similar to how cargo is loaded in the real world (step-by-step buildup). However, they require extensive amounts of runtime compared to other approaches. This is a common problem for physics-based approaches with high resolutions, which creates disadvantages when integrated with optimization algorithms [35,36]. In a recent loading stability benchmark, simulations performed well when compared with geometry- and force-based approaches, but required considerably more runtime than geometry-based approaches [9]. All geometry-, force-, and simulation-based approaches are prone to errors because they systematically over- or underestimate stability. Overestimations falsely predict that unstable cargo is actually stable; underestimations conservatively predict a stable layout to be unstable. To control for these errors, simulations depend on a set of adjustable hyperparameters, which need to be fitted for the specific simulation setting.

2.2. Approaches to computational physics

In the following, we outline the modeling characteristics of the three current modeling strategies in computational physics: physics-based models, data-driven models, and physics-informed learning. Table 1 summarizes their strengths and weaknesses.

2.2.1. Physics-based models and real-time physics engines

In computer graphics, a branch of computer science, physical phenomena are traditionally modeled within a set of known equations that reflect understood physical phenomena. This physically based modeling focuses on the computations of object behavior (such as shape and motion), which is determined by objects' physical properties [40]. Depending on the object of interest and its complexity, we distinguish between particles, rigid bodies, soft rigid bodies, articulated rigid body, mass–spring systems, skinned skeletons, and fluid flows [41].

Table 1
Overview of qualities of physics-based, data-driven and physics-informed learning approaches based on Kadambi et al. [37], Karniadakis et al. [38], and Blakseth et al. [39].

Approach	Strengths	Weaknesses
Physics-based modeling	+ generalizable + only small amounts of data required + interpretable + falsifiable	- computationally extensive - idealized, rather simplified situations
Data-driven modeling	+ low inference runtime + continuous updates after deployment	- prediction quality depends on training data quality and conditions - large amounts of training data necessary - black box, little interpretability
Physics-informed learning	+ stronger link to physical scenarios + generalizable for small amounts of data + robust for imperfect data	- depends on physical-grounded data - integration strategies needed (observational bias, inductive bias, learning bias)

In the case of dynamics (also known as physically based animation, simulation), we consider physical laws that govern kinematics (through Newton's laws of motion for large bodies) and represent objects by their positions and velocities. Physical laws are applied to advance the state between frames [41]. These laws of mechanics are well understood and can be modeled in dynamic constraints [40,41]. These dynamic simulations represent the state of a physical system given a time, inputs, and a beginning state. The state of the system is discretized and algebraically represented in non-linear ordinary differential equations, which are numerically solved [42]. Solving these equations is computationally intensive and challenging to accomplish in realistic timespans, so researchers have employed simplifications, optimizations, and numerical methods [40].

Physics-based models, because they use explicit physical knowledge, offer interpretability, falsifiability, and generalizability [37]. They require little data and are able to create realistic results under novel, unseen conditions, even when interacting with external objects [43]. Although these models are versatile, they assume idealized physical conditions, which is rarely the case in reality [37]. Further, they offer limited accuracy due to numerical imprecision. High-quality results require high resolutions, which, together with their general-purpose characteristic, increases the computational effort [39].

Among the physics-based dynamic simulation techniques that achieve the highest computational performances, real-time physics engines are the most important techniques that return plausible results instantaneously. Many different tasks and modules are coordinated and managed by them to simulate the basic parts of Newtonian mechanics for fluids, rigid bodies, or soft bodies [32]. Physics engines repeat three process steps in a simulation loop: collision detection, contact handling & collision resolution, and time integration. These three steps are usually split into two architectural components: collision detection and simulation [44]. The step-function forwards the time by a given interval:

$$stepSimulation(timeStep, maxSubSteps, fixedTimeStep) \quad (1)$$

where *timeStep* represents the forwarded time (in seconds), *maxSubSteps* quantifies the maximum number of sub-steps the physics engine is allowed to take within a single frame, and *fixedTimeStep* is the length of the internal fixed timestep and corresponds to the inverse of the desired internal simulation frequency [45,46]. Substeps are internally necessary to forward the time by a given small amount rather than jumping directly to the finish time [46]. The *fixedTimeStep* should be small enough to accurately capture the dynamics of the simulated objects. A high internal simulation frequency typically produces more accurate and stable results since it helps to prevent jumps and missed collisions [46]. In general, optimal values for *fixedTimeStep* and *maxSubStep* depend on the specific requirements of the simulation, which makes it necessary to experiment and tune these parameters to find the proper balance between accuracy and performance.

2.2.2. Data-driven physics and physics-informed learning

In contrast to physics-based models, data-driven models extract their knowledge from data rather than well-known physical equations. These approaches train machine learning models that learn the transition function between physical states from data of actual transitions. Successful applications include cloths [47], computer vision [37], particle physics [48], and human soft tissue animation [43]. In the tradeoff between computation speed and resolution of a physical model, these data-driven methods help speed up calculations significantly [49]. They perform well for real-life physical problems that have missing, gappy, or noisy boundary conditions and where traditional approaches reach their modeling limits [38]. They can reflect the entirety of a physical system [39] and, because they learn offline using pre-computations, they reduce computational effort extensively, as inference is very fast [47].

These models do have drawbacks. Knowledge manifested in neural network (NN) weights can make them difficult to interpret [39], and they require lots of real-world training data. They offer limited generalizability [39] and weak adaptability to changing or unseen conditions [43]. Their quality can be bounded only if the finite training sample and unseen test data are drawn from the same underlying distribution [37]. As this inductive hypothesis is unlikely under realistic conditions, they will be biased towards the training sample.

There are multiple strategies for a synergistic combination of physics- and data-driven physics in a hybrid physics-informed learning approach. These hybrid, physics-informed learning models seamlessly integrate noisy data and mathematical models within a network [38] at different components, such as inputs, outputs, loss functions, weights, and so on [37]. The objective is to integrate fundamental physical laws to achieve inductive biases [38]. Different combination strategies include incorporating physics into training datasets (observational bias); into network architecture (inductive bias), thus explicitly modeling physical laws in the form of mathematical constraints in the network; and into network loss functions (learning bias), thus explicitly modeling loss functions such that physically implausible results are penalized [37,38]. Other combination strategies are discussed in Blakseth et al. [39].

The observational bias strategy produces a physics-informed NN through a strong link of the training dataset to physics [37]. It purposefully introduces an observational bias into the NN, as the observational data reflect the governing physical structure [38]. The dataset can be either synthetic or real. In the synthetic case, the data can be generated by a physics engine that is constrained by the physical laws, which produces a data distribution strongly attached to some (physically) feasible set of data [37]. However, data generation's strong dependence on the underlying synthetic engine implicitly introduces idealized physical situations and oversimplifications [37]. Large volumes of data are necessary to extract and reinforce the observational biases [38].

Recent studies have shown that hybrid approaches combining physics-based and data-driven modeling outperform both single approaches [39] and offer symbiotic potential [43]. In a recent example,

Piloto et al. [15] developed a machine learning model, which accurately predicts relevant intuitive physical characteristics such as continuity, object persistence, solidity, unchangeableness, and directional inertia. The authors focused on visual training, using video sequences of physical object interactions procedurally generated using a physics engine. They conceptually applied a violation-of-expectation paradigm derived from developmental psychology. The machine learning model consists of two modules: a perception module responsible for latent object code prediction using an autoencoder and a recurrent dynamics module, based on an LSTM [50], which predicts object dynamics on the basis of the latent object code for the next frame.

3. Problem statement

We follow a loading stability approach to model cargo loading dynamics [9]. The input for the loading stability problem is a set of N item layouts. Each layout $A_i, i = 1, \dots, N$ consists of a sequence of J items $b_{i,j}, j = 1, \dots, J$, which determine item loading order. The layouts and loading sequences might be obtained through, for example, optimization. Every single item $b_{i,j}$ has its own three-dimensional loading position $x_{i,j}, y_{i,j}, z_{i,j}$; cuboid shape with width $w_{i,j}$, height $h_{i,j}$, and depth $d_{i,j}$; and weight $m_{i,j}$. Item positions, widths, heights, and depths are discrete values and organized in a grid of points with sets X, Y, and Z that include all possible coordinates along the width, height, and depth of the pallet. Thereby, $(0,0,0)'$ marks the left-bottom-back coordinate of the grid. Items are positioned such that their left-bottom-back coordinate lies on one of the points in the grid. An item's three-dimensional geometric center $CG_{i,j} = (CG_{i,j,x}, CG_{i,j,y}, CG_{i,j,z})'$ is obtained as $CG_{i,j,x} = x_{i,j} + w_{i,j}/2$, $CG_{i,j,y} = y_{i,j} + h_{i,j}/2$, and $CG_{i,j,z} = z_{i,j} + d_{i,j}/2$. The item's input, fixed three-dimensional floating center of mass $CM_{i,j} = (CM_{i,j,x}, CM_{i,j,y}, CM_{i,j,z})'$ can be obtained as $CM_{i,j,x} = CG_{i,j,x} + r_{i,j,x}$, $CM_{i,j,y} = CG_{i,j,y} + r_{i,j,y}$, and $CM_{i,j,z} = CG_{i,j,z} + r_{i,j,z}$, where $r_{i,j} = (r_{i,j,x}, r_{i,j,y}, r_{i,j,z})'$ shifts the center of mass relative to the geometric center. Note that the center of mass is a fixed property of the input item and is not updated during loading. Items must be loaded entirely within the pallet's dimensions and are placed using 90° rotations on the X-, Y-, and Z-axes. We assume all items to be rigid bodies. During loading, there is no lateral side support. We assume the loadable pallet contour to be cuboidal with dimensions W, H, and D and the total available space as $W \times H \times D$.

We define loading stability as a continuous property of an entire item layout. The amount of stability of a layout can be measured by the number of stable, safely loadable items in relation to the total number of items in the layout A_i. An item layout is stable if all items can be loaded safely without substantial changes in position and rotation. A layout is unstable if only a portion of the items can be loaded safely. This definition approximates a realistic, cumulative, and consecutive loading process as it would be done in practice. In practice, the items are loaded one after the other until the loading of the items leads to instabilities, either in the current item or in already loaded items. If an item layout becomes unstable, the loading process is stopped and no further estimation of hypothetical subsequent stability can be made.

Following Ramos et al. [51], we distinguish between two item sequences: the sequence A_i contains all items in the layout i and the subsequence $A_{i,j} \subseteq A_i$ includes the subsequence at sequence step j of all loaded items until item $b_{i,j}$. This means, for every newly placed item $b_{i,j}$ in step j, the entire subsequence $A_{i,j}$ is evaluated. The loading stability algorithm is finished if $j = J$. To quantify the amount of stability in relation to the number of items in the layout, we propose the following metric:

$$ls(A_i) = \frac{j_{max}}{J},\ j_{max} = j \text{ such that } ls(A_{i,j}) < 1, j = 1, \dots, J \tag{2}$$

The input to the loading stability algorithm ls is the complete set of items A_i for layout i, and the output is a stability score $ls(A_i)$ that derived from dividing the number of the highest stable loading sequence j_{max} by the total number of items J in the layout (Eq. (2)). This metric assumes that a layout can be safely and consecutively loaded up to the item in sequence step j. To calculate j_{max}, we consecutively consider all loading sequences $A_{i,j}$ until we find the first unstable sequence ($ls(A_{i,j}) = 0$) or all items are loaded and j reaches J. A subsequence $A_{i,j}$ is stable, if all items $b_{i,j}$ in $A_{i,j}$ are stable. A subsequence is unstable, if there is one item $b_{i,j}$ in $A_{i,j}$ that is unstable. The loading stability for a single item $b_{i,j}$ can be obtained by:

$$ls(b_{i,j}) = \begin{cases} 0 & \text{item } b_{i,j} \text{ is unstable}, i = 1, \dots, N, j = 1, \dots, J \\ 1 & \text{else} \end{cases} \tag{3}$$

Consequently, each item $b_{i,j}$ in $A_{i,j}$ is either stable ($ls(b_{i,j}) = 1$) or unstable ($ls(b_{i,j}) = 0$) (Eq. (3)).

4. Solution approaches

4.1. Goals & requirements

In this study, we propose two approaches, an optimized real-time physical simulation approach and a physics-informed learning approach, which we benchmark against other state-of-the-art approaches on a published multibody simulation dataset. We opt to develop loading stability approaches that achieve high accuracy while keeping the runtime low. Accuracy refers to the correct number of loading stability predictions, given some benchmark. Runtime performance refers to the average wall-clock time per loading stability approach. Often, there is an inherent tradeoff between these two, and it is up to the system designer to prioritize one or the other. We operationalize our goals by the mean accuracy f_a and runtime performance f_p, and obtain as

$$f_p = \frac{1}{N} \sum_{i=1}^{N} T_{end}(ls(A_i)) - T_{start}(ls(A_i)) \tag{4}$$

$$f_a = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{ls(A_i)==ls_b(A_i)} \qquad i = 1, \dots, N \tag{5}$$

where f_p measures the wall-clock difference between start timestamp $T_{start}(ls(A_i))$ and end timestamp $T_{end}(ls(A_i))$ for calculating the loading stability $ls(A_i)$ for layout A_i, $ls_b(A_i)$ represents the benchmark result of layout A_i, $\mathbb{1}_{ls(A_i)==ls_b(A_i)}$ compares the approach score to the loading stability benchmark result by comparing the respective ls scores, and N represents the number of layouts in our dataset.

4.2. Approach 1: Optimized real-time physical simulation

Our first approach PS_{opt} employs physics-based modeling using an optimized and problem-fitted real-time physics engine. Real-time physics engines are not as accurate as dedicated multibody simulators. They are a good approximation of dynamic multibody simulators for the PLP [32] because they are designed to deliver plausible results in shorter runtimes. These engines can be adjusted to approximate the results of a dynamic multibody simulator as closely as possible while consuming as little runtime as possible. Physics engines depend on a set of simulation hyperparameters that can be adjusted to control and balance accuracy and runtime. In this study, our first approach makes use of optimized physics engine hyperparameters.

To account for the dynamics of loading stability, every loading step j consists of a set of time steps $t, t = t_0, \dots, t_{max}$ that represent the current item state in a given time frame, during which the items move, collide, and rotate. These physical interactions lead to changes in spatial positions (translations) and orientations (rotations). To derive item translation, we calculate the Euclidean item movement Δ on the X, Y, and Z axes. For every time frame t, translation can be obtained as the absolute distance between end position and origin:

$$\Delta_{i,j,t} = \sqrt{(x_{i,j,t} - x_{i,j,t_0})^2 + (y_{i,j,t} - y_{i,j,t_0})^2 + (z_{i,j,t} - z_{i,j,t_0})^2} \tag{6}$$

The rotation is given by Euler roll, pitch, and yaw angles around the X-, Y-, and Z-axes $\Theta_{i,j,t} = (\varphi_{i,j,t}, \theta_{i,j,t}, \psi_{i,j,t})'$. We assume the rotation

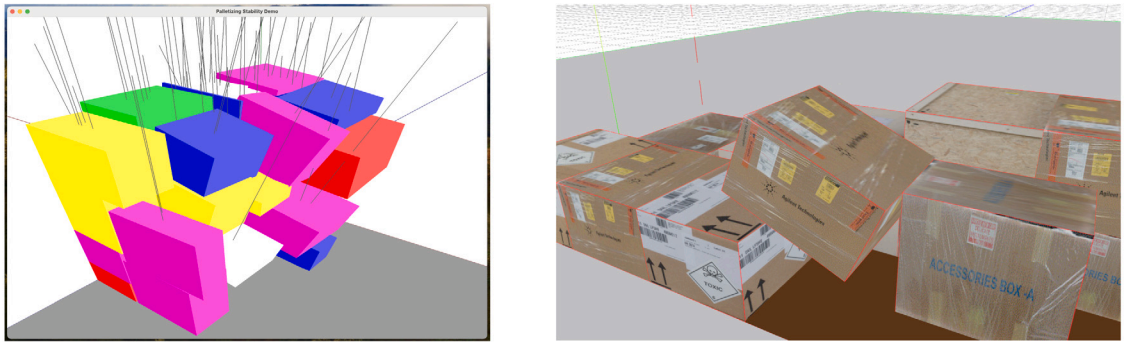

Fig. 1. Visualizations of the simulation-based approach using OpenGL and three.js.

angles at t_0 to be $\varphi_{i,j,t_0} = \theta_{i,j,t_0} = \psi_{i,j,t_0} = 0$. The final loading stability can be obtained as:

$$ls(b_{i,j}) = \begin{cases} 0, & \text{if } \Delta_{i,j,t} > \epsilon_T \vee max(\varphi_{i,j,t}, \theta_{i,j,t}, \psi_{i,j,t}) > \epsilon_R \\ 1, & \text{else} \end{cases} \tag{7}$$

The item $b_{i,j}$ at the loading step j is unstable if, for any t, the observed spatial displacement exceeds a translational threshold ϵ_T (e.g., 10 cm) or the observed rotation angles exceed a rotational threshold ϵ_R (e.g., 10°) (Eq. (7)). These tolerance values ϵ_T and ϵ_R depend on the practical scenario. For some PLP applications, small shifts of cargo during loading are unproblematic (e.g., if there is not much risk of damaged goods due to fragile items or the items have a high crumple tolerance). In other applications, the slightest movement of cargo items is critical. The tolerance values are also likely to depend on transportation mode and item sizes.

From the definition, we can derive the simulation algorithm. We illustrate the algorithm in pseudocode (Algorithm 1) and Fig. 1 provides visualizations. For a given cargo *layout*, our proposed approach adds items consecutively to the *physical_world* according to the loading step j. In each loading step, we simulate a number of *timesteps* (t_{max}) at a given internal simulation *frequency* (r) without constraining the number of simulation *substeps* (i.e., we fixed $substeps = t_{max} \times r$) and compare item 3D Euclidean translational (*b_translation*) and rotational displacements (*b_pitch, b_roll, b_yaw*) against threshold values (ϵ_T, ϵ_R). If an item that exceeds these thresholds is unstable, and so we return the loading stability ls as the current item sequence divided by the total number of items in the layout. If no unstable items are found, the entire layout is stable ($ls = 1$).

Algorithm 1 Simulation-based Loading Stability

```
procedure SIMULATE(layout, time, substeps, frequency, epsilon_translation,
epsilon_rotation)
    ls ← 0
    for each item j in layout do
        physics_world ← add(j.body)
        for each item b in physics_world.items do
            b_translation ← calculate_translation(b.position, b.origin)
            b_pitch, b_roll, b_yaw ← calculate_rotations(b.orientation)
            if b_translation > epsilon_translation then
                return ls
            else if b_pitch, b_roll, b_yaw > epsilon_rotation then
                return ls
            end if
        end for
        ls ← j/J
        physics_world ← step_simulation(time, substeps, frequency)
    end for
    return ls
end procedure
```

Our simulation-based algorithm depends on these hyperparameters: simulation length t_{max} (s); internal simulation frequency r (Hz); translational threshold ϵ_T (cm); rotational threshold ϵ_R (°); the material-based coefficient of restitution $\in$; coefficient of friction μ; linear and angular damping; collision margin (in m); and broad phase algorithm. Both threshold parameters ϵ_T and ϵ_R represent bounds for item stability. Items in rigid body simulation will always slightly move, as normal force development is complex to calculate for stacked objects, and the energy cannot be transferred into item deformation. Thresholds that are too restrictive lead to falsely declared instabilities (underestimations). If set too large, instabilities might be missed (over-estimations) [9]. The hyperparameter t_{max} determines the maximum number of simulation timesteps and steers simulation length. When moving, items displace stronger for higher values of t_{max}. However, longer simulations negatively influence runtime. The internal simulation frequency r refers to the inverse of the simulation's internal fixed timestep. Games and other applications of physics engines usually employ an internal simulation frequency of 60 Hz (corresponds to an internal fixed timestep of 0.01666 s), at which they work best and deliver real-time results [45]. A higher internal simulation frequency leads to more accurate results but increases runtime. The two properties of the coefficient of restitution $\in$ and the coefficient of friction μ are material-related. $\in$ refers to the elasticity of material; it defines the bounciness of a collision, which models the kinematic energy loss [42]. The friction μ is a force that opposes the normal force of two objects in contact and depends on the assortment of contact objects' materials. Linear and angular damping refer to linear and angular velocity loss over time, which also simulates frictional effects of translation and rotation such as air resistance. The collision margin is used in the physics engine for performance and reliability reasons [45]. The broad phase collision detection algorithm rejects object pairs based on their bounding box and accelerates calculations. There are several broad-phase algorithms, such as Simple, DBVT, and Sweep. Simple uses a brute force axis-aligned bounding box culling check algorithm; DBVT uses a dynamic bounding volume hierarchy; and Sweep uses an incremental sweep and prune algorithm [45].

4.3. Approach 2: Hybrid physics-informed learning

We base our second approach (LSTM) on the hybrid physics-informed learning. If real-world data are available, it may be possible to employ a data-driven approach and use the data as training input. For PLP, there is currently a substantial lack of real-world pallet loading datasets [52]. Combining both approaches, physics-informed learning makes use of data-driven training on physics-based, generated data produced by a dynamic simulator or real-time physics engine. We follow an observational bias strategy to train our machine learning model on physics-based generated data using a multibody simulation

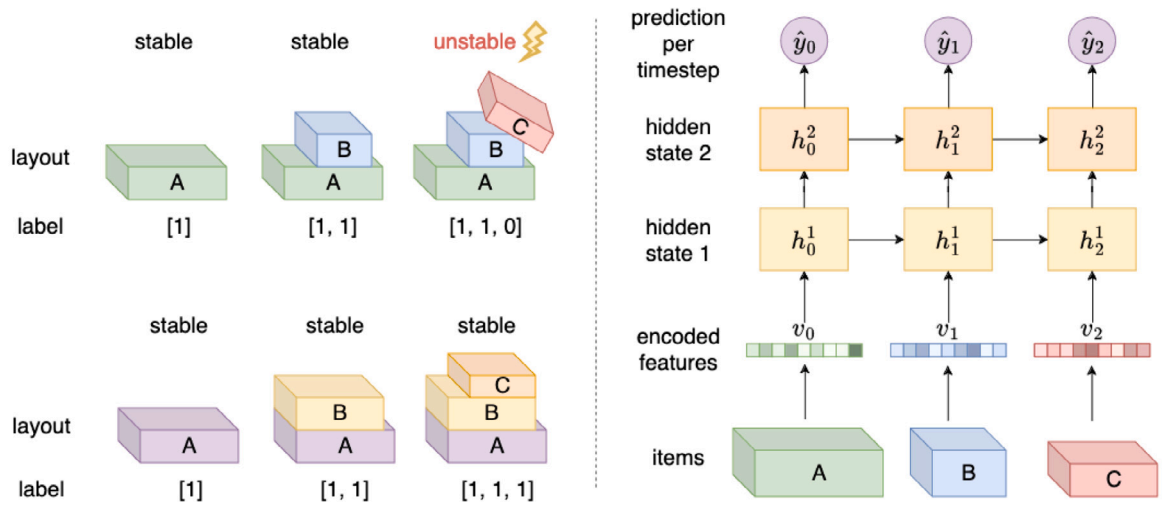

Fig. 2. Physics-informed learning approach with two example layout sequences and stability labels (left side) and LSTM-network architecture (right side) for one fixed layout i. The feature vector $v_{i,j}$ consists of item placement coordinates $x_{i,j}, y_{i,j}$, and $z_{i,j}$, item dimensions $w_{i,j}, h_{i,j}$, and $d_{i,j}$, weight $m_{i,j}$, and center of mass coordinates $CM_{i,j}$. The hidden states h^1 and h^2 opt to learn dynamics transition knowledge influencing the prediction per item step $\hat{y}_{i,j}$ given some novel item feature vector $v_{i,j}$ from subsequence $A_{i,j}$.

dataset with loading stability benchmark labels $ls_b(A_i)$. As the dataset contains stability labels and is based on a physical simulation, we assume the underlying physical knowledge is implicitly present. We use the publicly accessible loading stability benchmark for pallet loading problems [53] as our training basis. The dataset contains a total of approximately 29,000 cargo layouts consisting of items, item loading positions, and item loading sequences. Fig. 2 illustrates two examples of stable and unstable configurations. Further dataset information is provided in Section 5.1. For the architecture of our physics-informed learning approach, we chose an LSTM model, which is well suited for the sequential learning task of dynamics prediction. The usage of LSTMs is recommended for physics-informed learning [39] and has been used for object dynamics prediction [15]. For the LSTM, we assume one LSTM-timestep equals one sequence step j and consider each subsequence $A_{i,j}$ of layout A_i. The goal of the model is to predict whether a subsequence is stable. From each $A_{i,j}$ at step j, we create a feature vector $v_{i,j}$ that consists of the normalized values for the weight $w_{i,j}$, item placement coordinates $x_{i,j}, y_{i,j}, z_{i,j}$, dimensions $w_{i,j}, h_{i,j}, d_{i,j}$, and center of mass coordinates $CM_{i,j}$ (Fig. 2). At each step j, we feed the item feature vector $v_{i,j}$ (Fig. 2, right). For every step, a binary classification predicts whether the configuration is stable or unstable. We propose a classic single model and an ensemble model strategy. The single model LSTM_1 deploys one LSTM model for prediction. The ensemble model LSTM_e combines the predictions of multiple individually trained LSTM models. For this ensemble prediction, we average the probabilities of the individual models

$$\hat{y}_{ensemble} = \frac{1}{N}\sum_{k=1}^{N} \hat{y}_k \tag{8}$$

where $\hat{y}_{ensemble}$ represents the ensemble prediction, $\hat{y}_k$ denotes the prediction from the kth LSTM model and N is the number of individually trained LSTM models in the ensemble.

5. Experimental setup

In this section, we present the experimental setup, which consists of the test instances used and our hyperparameter optimization. We coded the approach PS_{opt} in Java and used the JBullet Java implementation of the Bullet Physics Library. The approach LSTM is based on Python, using the library PyTorch. We benchmarked the approaches on a workstation with an AMD Ryzen ThreadRipper 3960X 3.8 GHz CPU, 256 GB of RAM, and an NVIDIA GeForce RTX 3090 Ti GPU.

5.1. Test instances

As a basis for our evaluation, we used the publicly available PLP load stability dataset [9].[1] As realistic PLP datasets are missing [52], simulations are the best approximations of real-world pallet loading. This load stability dataset contains loading stability labels obtained from a simulation using the multibody simulator MSC ADAMS,[2] which allows us to directly compare stability approaches against each other.

The initial dataset contains two sub-datasets (1, 2) and three scenarios (S_1, S_{2a}, S_{2b}) with varying degrees of complexity modeled through CM positioning within the item. Scenario S_1 contains items with CM centered at the item's geometric center. Scenario S_{2a} introduces CM displacement using a normal distribution around the geometric center with $\mu = CG$ and standard deviation for each item given by width $\sigma_{w,i,j} = (w_{i,j} * 0.8)/6$, height $\sigma_{h,i,j} = (h_{i,j} * 0.8)/6$, and depth $\sigma_{d,i,j} = (d_{i,j} * 0.8)/6$, so approximately 99.7% of center of mass positions are within 80% of the items' bounds. When exceeded, this value was clipped. Thus, the majority of items have centers of mass close to their geometric centers. The scenario S_{2b} is the same as the scenario S_{2a} but follows a uniform distribution along the item's dimensions.

For this study, we selected sub-dataset 2, which is generated without any stability and thus removes creation algorithm bias and increases evaluation robustness. The dataset contains approx. 29,000 layouts with item dimensions, placement coordinates, loading sequences, and weights that are proportional to item volumes. The layouts were generated using 160 packing heuristics developed by combining four bin selection strategies, eight space selection strategies, and five placement rules [54]. The item dimensions were randomly sampled from a uniform distribution of intervals [1, 10], [1, 35], and [1, 100], respectively. The size of the loadable pallet contour is either $30 \times 30 \times 30$ or $100 \times 100 \times 100$. To test more corner cases of stability, we do not filter out items with unrealistic dimensions (e.g., very thin objects). This dataset is rather unrealistic with respect to the item characteristics; however, it does not include a static stability generation bias.

We performed a train–validation–test split to overcome overfitting of the hyperparameter optimization on the training sample. The training dataset is balanced towards stable layouts (approx. 76.32% of layouts are stable). Balancing the validation and test set in terms of stable to unstable layouts helps to get a realistic estimate of the algorithm's

[1] https://zenodo.org/records/13925610.

[2] https://hexagon.com/de/products/product-groups/computer-aided-engineering-software/adams.

Table 2
Overview of data splits, scenarios, and labels.

Scenario	Training			Validation			Test		
	Stable	Unstable	Total	Stable	Unstable	Total	Stable	Unstable	Total
S_1	7458	1843	9301	44	44	88	150	150	300
S_{2a}	7163	2127	9290	50	49	99	150	150	300
S_{2b}	6649	2627	9276	56	57	113	150	150	300
$S_1 - S_{2b}$	21 270	6597	27 867	150	150	300	450	450	900

Table 3
Hyperparameter assortment, search range, and optimized values for the real-time physical simulation approach PS_{opt}.

Parameter	Unit	Search range			Optimized value PS_{opt}
		Min	Max	Step	
ϵ_T	m	0.01	1	0.01	0.22
ϵ_R	°	5	15	1	7
t_{max}	s	0.1	3	0.1	0.3
r	Hz	5	100	5	90
$\in$	–	0.05	0.2	0.01	0.11
μ	–	0.4	0.6	0.01	0.5
Linear damping	–	0.01	0.2	0.05	0.06
Angular damping	–	0.01	0.2	0.05	0.01
Collision margin	m	0.001	0.02	0.001	0.006
Broad phase	–	[Simple, DBVT, Sweep]			DBVT

Table 4
Hyperparameter assortment, search range, and optimized values for the individual LSTM approaches.

Parameter	Search range	Optimized value LSTM				
		$LSTM_1$	$LSTM_2$	$LSTM_3$	$LSTM_4$	$LSTM_5$
Learning rate ($\times 10^{-3}$)	[0.1; 1]	0.35	0.15	0.67	0.19	0.71
Hidden size	[100, 125, 150, 175, 200]	125	175	150	150	200
Optimizer	[Adam, Adamw]	Adam	Adam	Adam	Adam	Adam

performance and overcome an algorithm bias towards stable layouts. Therefore, for validation and test samples, we selected 50% instances that are stable ($ls_b = 1$) and 50% instances that are unstable ($ls_b < 1$). This selection contributes to overcoming algorithm bias towards stable layouts. Table 2 shows the final dataset, splits, scenarios, and labels.

5.2. Hyperparameter optimization

Both of our new loading stability approaches depend on a set of hyperparameters that can be adjusted. For the real-time physical simulation approach, we first defined the hyperparameter search range by the recommended values of the physics engine and the simulation values of the benchmark simulation [9]. Then, we refined the search range by iterative testing and by narrowing down the intervals to an acceptable range. Finally, we employ a genetic algorithm (GA) to solve the hyperparameter optimization problem. Our genotype encodes the hyperparameters ϵ_T, ϵ_R, t_{max}, r, $\in$, μ, linear damping, angular damping, collision margin, and broad phase algorithm as genes. Each gene is allowed to take an allele according to the allowed value space in our search range, which we display in Table 3. Derived from our goals, our GA fitness function f is composed of a weighted average accuracy f_a and weighted average performance fitness f_p and obtains as $f = w_a * f_a + w_p * f_p$. where w_a and w_p are the fitness weights. For every genotype, we execute the simulation N times. We employ a random mutation strategy, a mutation number of genes of two, rotated wheel selection, a uniform crossover strategy, a generation size of 50, and a population size of 1000. We use the validation split as the training basis for the GA hyperparameter optimization since it contains a balanced stability distribution. We set the fitness function weights to $w_a = 0.85$ and $w_p = 0.15$.

For the LSTM approach optimization, we combine an ensemble strategy with a Bayesian hyperparameter optimization to optimize the key model parameters: learning rate, hidden size, optimizer, number of RNN layers, and number of epochs. To find out relevant hyperparameters and specify a reasonable hyperparameter value search range, we conduct an initial hyperparameter search. The initial search results in a reasonable hyperparameter value search range displayed in Table 4 with dominant values for the Adam optimizer [55], fixed number of RNN layers (2), and a fixed epoch size (700). We set the number of epochs to 700, as the validation accuracy in the training did not improve significantly after this threshold value. We employed cross entropy loss, which is well-suited for our classification task. For the remaining two hyperparameters, learning rate and hidden size, we sample five different train–validation splits from the data, on which we conduct a refined hyperparameter search with the search range obtained in the previous step. The outcome of this process consists of five optimized models, $LSTM_1$-$LSTM_5$, from which we used the first for single model prediction ($LSTM_1$) and all five models for ensemble prediction ($LSTM_e$).

6. Results

We benchmark our novel approaches, PS_{opt}, $LSTM_1$, and $LSTM_e$ against two non-optimized baseline models, PS_3 and PS_5 with translational thresholds of $\epsilon_T = 0.03$ and $\epsilon_T = 0.05$. Further, we include in our benchmark the other common static stability approaches $PBS_{0.5}$, $PBS_{0.7}$, and $PBS_{0.9}$, which represent PBS with $\alpha = 0.5$, $\alpha = 0.7$, $\alpha = 0.9$, FBS, and SME. We depict the mean approach accuracy and runtimes in Table 5 and illustrate the accuracy-runtime tradeoff in Fig. 3.

Mean accuracy represents f_a and refers to the mean number of correct stability scores $ls(A_i)$ compared with the loading stability benchmark result $ls_b(A_i)$ for layout i in the test set (Eq. (5)). The stability score must exactly match, so we treat overestimations (a stability approach predicts a higher stability score compared to the benchmark) and underestimations (a stability approach predicts a lower stability score compared to the benchmark) of a layout's stability approach as wrong predictions. Both outcomes are incorrect and might be associated with different costs. Underestimation represents an overly

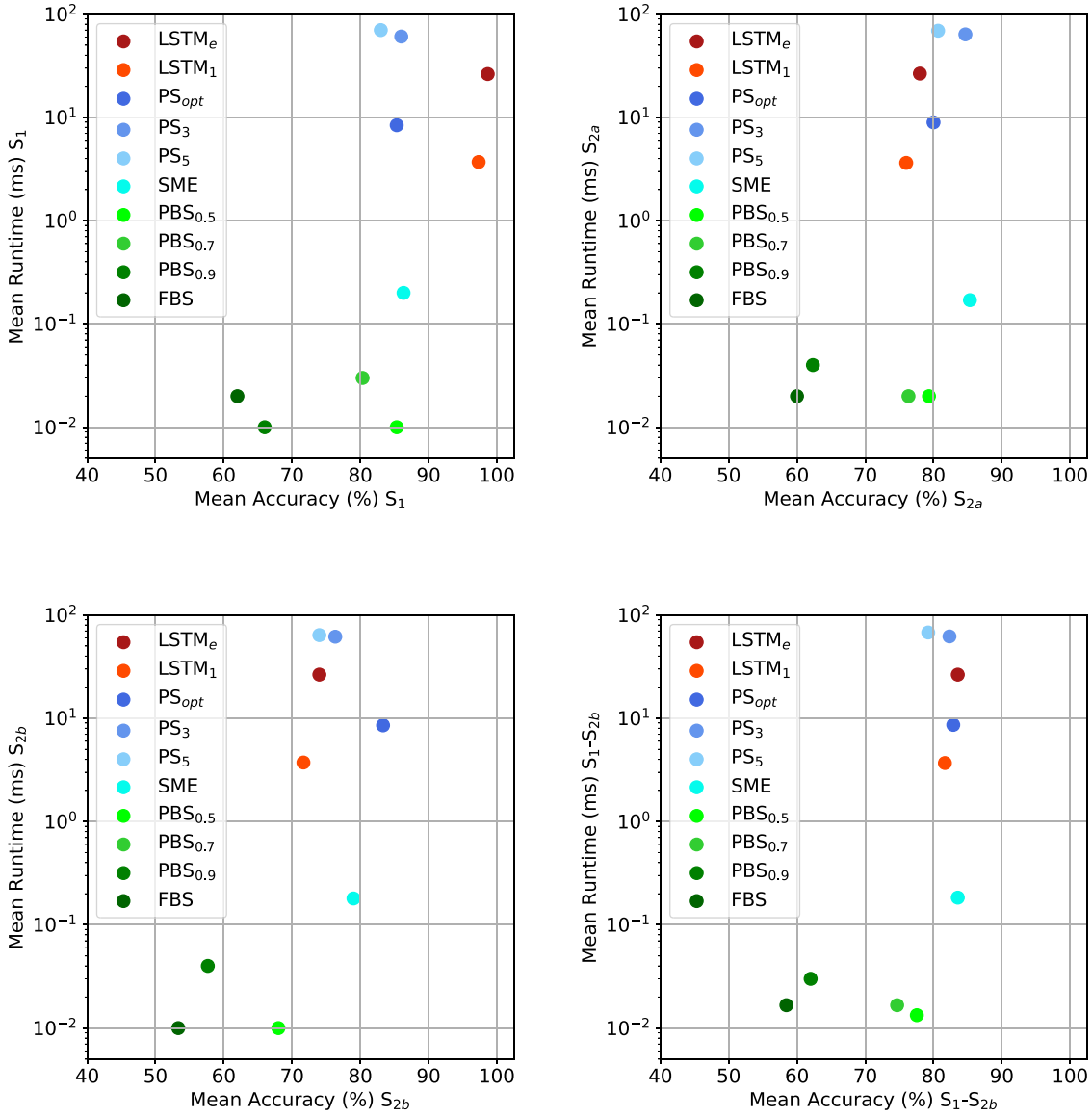

Fig. 3. Mean Runtime/Accuracy comparison for scenarios S_1 (top-left), S_{2a} (top-right), S_{2b} (bottom-left), and S_1-S_{2b} (bottom-right).

Table 5
Computational experiments results mean accuracy and runtime (our novel PS_{opt}, $LSTM_1$, and $LSTM_e$ approaches are displayed on the right).

Objective	Scenario	FBS	$PBS_{0.9}$	$PBS_{0.7}$	$PBS_{0.5}$	SME	PS_5	PS_3	PS_{opt}	$LSTM_1$	$LSTM_e$
Accuracy (%)	S_1	62.00	66.00	80.33	85.33	86.33	83.00	86.00	85.33	97.33	**98.67**
	S_{2a}	60.00	62.33	76.33	79.33	**85.33**	80.67	84.67	80.00	76.00	78.00
	S_{2b}	53.33	57.67	67.33	68.00	79.00	74.00	76.33	**83.33**	71.67	74.00
Avg. (S_1-S_{2b})	–	58.44	62.00	74.67	77.56	**83.56**	79.22	82.33	82.89	81.67	**83.56**
Runtime (ms)	S_1	0.02	**0.01**	0.03	**0.01**	0.20	70.12	60.72	8.40	3.70	26.30
	S_{2a}	**0.02**	0.04	**0.02**	**0.02**	0.17	69.13	63.70	8.95	3.63	26.57
	S_{2b}	**0.01**	0.04	0.00	**0.01**	0.18	63.85	61.63	8.54	3.73	26.47
Avg. (S_1-S_{2b})	–	0.02	0.03	0.03	**0.01**	0.18	69.62	62.21	8.68	3.67	26.44

*The best values appear in bold.

conservative estimate and can lead to suboptimal space utilization. Overestimations can produce damaged cargo [9]. Mean runtime represents f_p and refers to the average wall-clock time for the calculation of a layout using a stability approach.

We observe considerable accuracy differences throughout the approaches. FBS and PBS with higher α levels perform worse compared to the other approaches. Both LSTM approaches achieve the highest accuracy for scenario S_1. Notably, both LSTM approaches perform worse for more complex scenarios S_{2a} (78.00–76.00) and S_{2b} (74.00–71.67) than the simpler scenario S_1 (98.67–97.33), while the ensemble model $LSTM_e$ consistently performs better than the single model $LSTM_1$. The same pattern applies to SME, PS_3, and PS_5 with the only exception

of PS_{opt}. PS_{opt} performs better for the more complex scenario S_{2b} than S_{2a}. What is striking in these results is the rapid decrease in accuracy for $PBS_{0.5}$, when moving from S_1 (85.33) to shifted CM scenarios S_{2a} (79.33) and S_{2b} (68.00). The mean accuracy among all scenarios is not clearly dominated by a single approach; rather, we observe SME and $LSTM_e$ being the most accurate (83.56), followed by the simulation-based approaches PS_{opt} (82.89), PS_3 (82.33), and PS_5 (79.22), and the geometry-related approaches (77.56–58.44). What stands out is the general pattern of higher α levels being attached to lower prediction accuracy.

When we consider runtime, the geometry-related approaches clearly consume the least runtimes. We identify four clusters: geometry-related approaches; SME; $LSTM_1$, PS_{opt}; and the baseline simulations and the ensemble model $LSTM_e$. Notably, the slowest approach (PS_5: 69.62 ms) consumes on average approximately 7000 times the runtime of the fastest approach ($PBS_{0.5}$: 0.01 ms). Taking into account the scenarios, we observe little differences apart from PS_3. For this approach, we note a runtime decrease when moving from S_1 (70.12 ms) to S_{2a} (69.13 ms) and S_{2b} (63.85 ms). Among the baseline models, PS_5 is slower (69.62 ms) compared to PS_3 (62.21 ms). What is striking is the large difference between optimized simulation PS_{opt} (8.68 ms) and the baseline simulations PS_3 and PS_5.

To emphasize the tension between runtime and accuracy, we illustrate in Fig. 3 the results with respect to our objectives of prediction accuracy and runtime. The runtime axis is log-scaled. The optimal quadrant is the lower right quadrant, with a low runtime paired with high accuracy. We observe the three clusters: geometry-related approaches; SME; and $LSTM_1$, $LSTM_e$, PS_{opt} and the baseline models. For most of the approaches, we observe a tradeoff between accuracy and runtime. We overlook the slight pattern of increased runtime, especially when considering moving to the right and increasing accuracy.

7. Discussion and conclusion

The objective of this study is to develop approaches that provide decision support for the real-world physical dynamics problem of evaluating cargo loading stability for PLP. Derived from computational physics, we came up with two new approaches to solve the loading stability problem for PLP: an optimized physical simulation and a physics-informed learning approach. In this study, we formulate two objectives for a good algorithm to solve the cargo loading stability problem: accuracy and runtime. Accuracy refers to the number of correctly predicted stable items when loading an arrangement on a pallet. The accuracy is the most important objective since inaccurate predictions can cause damaged goods or personnel. Runtime is the second important objective, since algorithms need to be fast when integrated into algorithms that solve the PLP. Often, strict time constraints exist due to transportation schedules in combination with limited storage space, as in the example of cross-docking [56].

The results of our computational experiments show that our novel modeling approaches can make more accurate predictions than current state-of-the-art methods, depending on the scenario. Our key finding is that the modeling using optimized physical simulation and physics-informed learning can be a useful modeling choice. Our assumption is that both approaches can predict loading stability more accurately since they include more physical knowledge, either implicit through data-driven training on physically-based data or explicit through modeling and simulation. Our results show that there is no perfect one-size-fits-all approach. For easier scenarios, $LSTM_e$ can be a useful choice; SME excels for the intermediate scenario S_{2a}, and PS_{opt} performs best in the most complex scenario S_{2b}. We further observe that PS_{opt} is the most robust approach, as it was subject to the least variation in accuracy of the scenarios. For a good trade-off between accuracy and runtime, SME is a good fit.

These results, however, are difficult to generalize beyond the case of loading stability in PLP because other transportation modes may require different modelings. In this study, we assumed items to be rigid bodies, which allowed us to employ rigid body dynamics. Rigid body dynamics is one of the easiest modeling forms and computationally inexpensive [32]. Our assumption of rigid bodies is unrealistic but necessary to employ the computationally simpler rigid body physics compared to, e.g., soft body physics.

Physics-informed learning depends highly on the underlying data distribution. At present, there are no real-world data available from PLP in general or loading stability in particular; therefore, simulations are the best approximations of real-world physical dynamics. The absence of datasets is not surprising, as every fallen cargo item can cause damage and costs. However, it remains unclear if our training data distribution represents real-life distributions of cargo arrangements. If there is a mismatch between training data distribution and real-life distribution, the algorithms might be of limited accuracy in realistic scenarios.

As the benchmark dataset is generated using a multibody simulator and this simulation approximates real-world physical dynamics, our results also rely on the correctness of the benchmark simulation model. However, the generated data may not perfectly reflect real-world physical conditions, which renders the physics-informed learning approach less applicable. The availability of data is key for the further development of both physics-informed learning and initial application of data-driven approaches. The observational bias should come from a data distribution with realistic implicit physical knowledge, foremost from a real-world field study. Further, we assumed one item loading sequence equals one timestep with a concrete stability label during LSTM training and prediction. This abstraction simplifies the problem.

Although our hyperparameter optimization has a positive impact on the prediction accuracy and contributes to substantially lower runtimes of PS_{opt}, it may introduce unrealistic physical hyperparameter values. As our hyperparameter optimization is output-driven by prediction accuracy and runtime performance, we only control hyperparameter plausibility through parameter search range. In the case of simulation runtime t_{max}, the optimized approach coherently obtains as 0.3 s, exactly the same amount of time in the benchmark simulation [9]. When we compare PS_{opt} to the benchmark, we find no clear correspondence but, rather, a tendency towards matching parameter values: ϵ_T: 0.22 (PS_{opt}) vs. 0.1 (benchmark), ϵ_R: 7 (PS_{opt}) vs. 10 (benchmark). The difference between a real-time physics engine and a multibody simulator may explain this deviation. Real-time physics engines are faster because they use fixed integrator timesteps. Multibody simulators, in contrast, can become numerically accurate (the benchmark integrator error is 1.0e−7) and may obtain slightly different simulation results.

Our runtime analysis focuses on the overall wall-clock execution runtime, which depends on the number of items and underlying hardware. Although optimized simulations and machine learning approaches are not yet as fast as the other geometry-related and force-based approaches, we recognize a substantial runtime difference compared to the simulation baseline models PS_3 and PS_5.

In general, the LSTM approach is promising, as little explicit knowledge must be taken into account. The problem of determining physical knowledge is complex for a neural network, and the first approaches are developed that tackle these problems. The key advantage of physics-informed learning is the option to self-generate massive amounts of training data and use them as input for the training of the machine learning algorithm. For the simplest scenario S_1, its accuracy is very high compared to the other approaches. For the more complex scenarios S_{2a} and S_{2b} in particular, where the center of mass is shifted, the accuracy drops. This could indicate that the LSTM model was not able to correctly internalize the meaning of a shifted CM. Our ensemble hyperparameter optimization strategy for $LSTM_e$ contributes to increased accuracy compared to the single model $LSTM_1$ but also

negatively affects runtime. Concerning LSTM runtime, another remark is that we conducted the runtime analysis using a batch size of 1. This is a conservative estimate of the runtime, since usually inference would be batched to concurrently predict the stability of multiple layouts at the same time. Therefore, batching likely decreases the average runtime per layout.

A different finding concerns the decreasing accuracy given FBS and PBS with a higher level of α-values. This is in line with previous research [9] and is likely due to the differentiation between over- and underestimation of stability by the various approaches. In this study, our accuracy metric is constructed such that it penalizes overly conservative or deliberate stability estimations likewise. Since FBS and PBS with higher α-values tend to underestimate stability, the accuracy decreases [9].

Future research could build on the findings and test different network architectures, such as transformer learning. In the past, transformers have been successfully deployed for sequential data, for example, natural language processing. However, they depend on massive amounts of data.

From a managerial perspective, closer problem modeling is beneficial because it enables practitioners to predict and assess loading stability prior to the actual loading operations and recognize potential dangerous cargo loadings. Avoiding dangerous cargo loadings helps to mitigate costs and injuries by fallen and damaged cargo. To meet logistic deadlines, our focus on accuracy and runtime can contribute to the development of faster systems for PLP that would deliver solutions quickly to cope with tight transportation schedules. Therefore, our approach provides a good step towards accurate but also fast results.

A possible future research path emerges when applying the proposed method for other PLP constraints, such as dynamic stability or load bearings. Both use physical properties and predict physical behavior; both need realistic physical feedback. However, the prediction target, item sequence characteristics, and placement procedures are different. Since dynamic stability focuses on moving vehicles, no loading operations are performed, which reduces the number of simulations considerably, but also includes the modeling of different forces that approximate accelerations or braking dynamics.

So far, there has been research in the fields of computer science and physics to develop physics-informed learning techniques and physics engines. However, little research has investigated applications of computational physics for practical transportation problems. Many transportation and logistics applications optimize for an economic goal while ensuring practical relevance and applicability of approaches. The closer modeling of physical conditions and behavior might be an underresearched but promising future direction. Novel insights from computational physics, especially in combination with machine learning, can lead the way.

CRediT authorship contribution statement

Philipp Gabriel Mazur: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Johannes Werner Melsbach:** Writing – review & editing, Writing – original draft, Validation, Methodology, Conceptualization. **Detlef Schoder:** Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

We have no conflict of interest to declare.

Data availability

Data will be made available on request.

References

[1] Bortfeldt A, Wäscher G. Constraints in container loading – a state-of-the-art review. European J Oper Res 2013;229(1):1–20. http://dx.doi.org/10.1016/j.ejor.2012.12.006.

[2] Bischoff EE. Stability aspects of pallet loading. OR Spektrum 1991;13(4):189–97. http://dx.doi.org/10.1007/BF01719394.

[3] Parreño F, Alvarez-Valdes R, Tamarit JM, Oliveira JF. A maximal-space algorithm for the container loading problem. INFORMS J Comput 2008;20(3):412–22. http://dx.doi.org/10.1287/ijoc.1070.0254, URL http://pubsonline.informs.org/doi/10.1287/ijoc.1070.0254.

[4] Junqueira L, de Queiroz TA. The static stability of support factor–based rectangular packings: An assessment by regression analysis. Int Trans Oper Res 2022;27(2):311. http://dx.doi.org/10.1111/itor.12750.

[5] Jones S. Northern Spain on alert as plastic pellets from cargo spill wash up on beaches. Guardian 2024. URL https://www.theguardian.com/world/2024/jan/09/northern-spain-plastic-pellets-cargo-spill-beaches.

[6] Hodgson TJ. A combined approach to the pallet loading problem. A I I E Trans 1982;14(3):175–82. http://dx.doi.org/10.1080/05695558208975057, URL http://www.tandfonline.com/doi/abs/10.1080/05695558208975057.

[7] Krebs C, Ehmke JF. Vertical stability constraints in combined vehicle routing and 3D container loading problems. In: Mes M, Lalla-Ruiz E, Voß S, editors. Computational logistics. Lecture notes in computer science, Cham: Springer International Publishing; 2021, p. 442–55. http://dx.doi.org/10.1007/978-3-030-87672-2_29.

[8] Ramos AG, Oliveira JF, Gonçalves JF, Lopes MP. A container loading algorithm with static mechanical equilibrium stability constraints. Transp Res Part B: Methodol 2016;91:565–81. http://dx.doi.org/10.1016/j.trb.2016.06.003, URL https://linkinghub.elsevier.com/retrieve/pii/S0191261515302022.

[9] Mazur PG, Gamer F, Ramos AG, Schoder D. Standing on a common ground: A comparison of static stability approaches for pallet loading. Preprint. 2024, http://dx.doi.org/10.2139/ssrn.4778113, URL https://papers.ssrn.com/abstract=4778113.

[10] Zhu W, Fu Y, Zhou Y. 3D dynamic heterogeneous robotic palletization problem. European J Oper Res 2024;316(2):584–96. http://dx.doi.org/10.1016/j.ejor.2024.02.007, URL https://www.sciencedirect.com/science/article/pii/S0377221724000985.

[11] Hummel J, Wolff R, Stein T, Gerndt A, Kuhlen T. An evaluation of open source physics engines for use in virtual reality assembly simulations. In: Hutchison D, Kanade T, Kittler J, editors. Advances in visual computing. Lecture notes in computer science, vol. 7432, Berlin, Heidelberg: Springer Berlin Heidelberg; 2012, p. 346–57. http://dx.doi.org/10.1007/978-3-642-33191-634.

[12] Senatore G, Piker D. Interactive real-time physics: An intuitive approach to form-finding and structural analysis for design and education. Computer- Aided Des 2015;61:32–41. http://dx.doi.org/10.1016/j.cad.2014.02.007, URL https://www.sciencedirect.com/science/article/pii/S0010448514000311.

[13] Zhu W, Fan X, Tian L, He Q. An integrated simulation method for product design based on part semantic model. Int J Adv Manuf Technol 2018;96(9):3821–41. http://dx.doi.org/10.1007/s00170-018-1808-1, URL https://doi.org/10.1007/s00170-018-1808-1.

[14] Glatt M, Sinnwell C, Yi L, Donohoe S, Ravani B, Aurich JC. Modeling and implementation of a digital twin of material flows based on physics simulation. J Manuf Syst 2021;58:231–45. http://dx.doi.org/10.1016/j.jmsy.2020.04.015, URL https://www.sciencedirect.com/science/article/pii/S0278612520300595.

[15] Piloto LS, Weinstein A, Battaglia P, Botvinick M. Intuitive physics learning in a deep-learning model inspired by developmental psychology. Nat Hum Behav 2022;6(9):1257–67. http://dx.doi.org/10.1038/s41562-022-01394-8, Publisher: Nature Publishing Group. URL https://www.nature.com/articles/s41562-022-01394-8.

[16] Bracht EC, de Queiroz TA, Schouery RCS, Miyazawa FK. Dynamic cargo stability in loading and transportation of containers. In: 2016 IEEE international conference on automation science and engineering. CASE, IEEE; 2016, p. 227–32. http://dx.doi.org/10.1109/COASE.2016.7743385, URL http://ieeexplore.ieee.org/document/7743385/.

[17] Mazur PG, Lee N-S, Schoder D. Integration of physical simulations in static stability assessments for pallet loading in air Cargo. In: Bae K-H, Feng B, Kim S, Lazarova-Molnar S, Zheng Z, Roeder T, Thiesing R, editors. Proceedings of the 2020 winter simulation conference, December 13-16, Orlando, Florida, USA. 2020, p. 1312–23. http://dx.doi.org/10.1109/WSC48552.2020.9383878.

[18] Wäscher G, Haußner H, Schumann H. An improved typology of cutting and packing problems. European J Oper Res 2007;183(3):1109–30. http://dx.doi.org/10.1016/j.ejor.2005.12.047, Publisher: North-Holland. URL https://www.sciencedirect.com/science/article/pii/S037722170600292X.

[19] Limbourg S, Schyns M, Laporte G. Automatic aircraft cargo load planning. J Oper Res Soc 2012;63(9):1271–83. http://dx.doi.org/10.1057/jors.2011.134, Publisher: Nature Publishing Group. https://www.tandfonline.com/doi/full/10.1057/jors.2011.134.

[20] Brandt F, Nickel S. The air cargo load planning problem - a consolidated problem definition and literature review on related problems. European J Oper Res 2019;275(2):399–410. http://dx.doi.org/10.1016/j.ejor.2018.07.013, Publisher: Elsevier B.V..

[21] Pollaris H, Braekers K, Caris A, Janssens GK, Limbourg S. Vehicle routing problems with loading constraints: state-of-the-art and future directions. OR Spectrum 2015;37(2):297–330. http://dx.doi.org/10.1007/s00291-014-0386-3, URL http://link.springer.com/10.1007/s00291-014-0386-3.

[22] Ramos AG, Oliveira JF. Cargo stability in the container loading problem - State-of-the-art and future research directions. Springer Proc Math Stat 2018;223:339–50. http://dx.doi.org/10.1007/978-3-319-71583-4_23.

[23] Bischoff EE, Ratcliff MSW. Issues in the development of approaches to container loading. Omega 1995;23(4):377–90. http://dx.doi.org/10.1016/0305-0483(95)00015-G.

[24] Junqueira L, Morabito R, Sato Yamashita D. Three-dimensional container loading models with cargo stability and load bearing constraints. Comput Oper Res 2012;39(1):74–85. http://dx.doi.org/10.1016/j.cor.2010.07.017, Publisher: Elsevier.

[25] Elhedhli S, Gzara F, Yildiz B. Three-dimensional bin packing and mixed-case palletization. Informs J Optim 2019;1(4):323–52. http://dx.doi.org/10.1287/ijoo.2019.0013, URL http://pubsonline.informs.org/doi/10.1287/ijoo.2019.0013.

[26] Krebs C, Ehmke JF. Axle weights in combined vehicle routing and container loading problems. EURO J Transp Logist 2021;10:100043. http://dx.doi.org/10.1016/j.ejtl.2021.100043, URL https://www.sciencedirect.com/science/article/pii/S2192437621000157.

[27] Brandt F. The air cargo load planning problem [Ph.D. thesis], Karlsruher Instituts für Technologie (KIT); 2017.

[28] Olsson J, Larsson T, Quttineh N-H. Automating the planning of container loading for Atlas Copco: Coping with real-life stacking and stability constraints. European J Oper Res 2020;280(3):1018–34. http://dx.doi.org/10.1016/j.ejor.2019.07.057, Publisher: Elsevier B.V. URL https://linkinghub.elsevier.com/retrieve/pii/S0377221719306368.

[29] Gajda M, Trivella A, Mansini R, Pisinger D. An optimization approach for a complex real-life container loading problem. Omega 2022;107(C). Publisher: Elsevier. URL https://ideas.repec.org/a/eee/jomega/v107y2022ics0305048321001687.html.

[30] Ramos AG, Jacob J, Justo JF, Oliveira JF, Rodrigues R, Gomes AM. Cargo dynamic stability in the container loading problem - a physics simulation tool approach. Int J Simul Process Model 2017;12(1):29–41. http://dx.doi.org/10.1504/IJSPM.2017.10003690.

[31] Martínez-Franco J, Céspedes-Sabogal E, Álvarez-Martínez D. PackageCargo: A decision support tool for the container loading problem with stability. SoftwareX 2020;12:100601. http://dx.doi.org/10.1016/j.softx.2020.100601, URL https://www.sciencedirect.com/science/article/pii/S2352711020303149.

[32] Martinez-Franco JC, Alvarez-Martinez D. Physx as a middleware for dynamic simulations in the container loading problem. In: 2018 winter simulation conference. WSC, IEEE; 2018, p. 2933–40. http://dx.doi.org/10.1109/WSC.2018.8632469, Issue: 2014. URL https://ieeexplore.ieee.org/document/8632469/.

[33] Nishiyama S, Lee C, Mashita T. Designing a flexible evaluation of container loading using physics simulation. In: Optimization and learning: third international conference, OLA. 1, Springer International Publishing; 2020, p. 255–68. http://dx.doi.org/10.1007/978-3-030-41913-4_21, URL http://dx.doi.org/10.1007/978-3-030-41913-4_21.

[34] Mazur PG, Lee N-S, Schoder D. A GPU-accelerated approach of static stability assessments for pallet loading in air cargo. In: Proceedings of the 55th Hawaii international conference on system sciences (2022). 2022, p. 1177–85.

[35] Oliveira Ld, de Lima VL, de Queiroz TA, Miyazawa FK. The container loading problem with cargo stability: A study on support factors, mechanical equilibrium and grids. Eng Optim 2021;53(7):1192–211. http://dx.doi.org/10.1080/0305215X.2020.1779250, Publisher: Taylor & Francis _eprint.

[36] Ramos AG, Oliveira JF, Gonçalves JF, Lopes MP. Dynamic stability metrics for the container loading problem. Transp Res Part C: Emerg Technol 2015;60:480–97. http://dx.doi.org/10.1016/j.trc.2015.09.012, URL https://linkinghub.elsevier.com/retrieve/pii/S0968090X15003459.

[37] Kadambi A, de Melo C, Hsieh C-J, Srivastava M, Soatto S. Incorporating physics into data-driven computer vision. Nat Mach Intell 2023;5(6):572–80. http://dx.doi.org/10.1038/s42256-023-00662-0, Number: 6 Publisher: Nature Publishing Group. URL https://www.nature.com/articles/s42256-023-00662-0.

[38] Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S, Yang L. Physics-informed machine learning. Nat Rev Phys 2021;3(6):422–40. http://dx.doi.org/10.1038/s42254-021-00314-5, Number: 6 Publisher: Nature Publishing Group. URL https://www.nature.com/articles/s42254-021-00314-5.

[39] Blakseth SS, Rasheed A, Kvamsdal T, San O. Combining physics-based and data-driven techniques for reliable hybrid analysis and modeling using the corrective source term approach. Appl Soft Comput 2022;128:109533. http://dx.doi.org/10.1016/j.asoc.2022.109533, URL https://www.sciencedirect.com/science/article/pii/S156849462200607X.

[40] Foley JD, Dam AV, Feiner SK, Hughes JF. Computer graphics: Principles and practice. 2nd ed.. Boston: Addison-Wesley Professional; 1996.

[41] Hughes JF, Dam Av, McGuire M, Sklar DF, Foley JD, Feiner SK, Akeley K. Computer graphics: Principles and practice. 3rd ed.. Upper Saddle River (NJ): Addison-Wesley; 2013.

[42] Baraff D. Rigid body simulation. SIGGRAPH course notes 1997, vol. 24, 1997.

[43] Kim M, Pons-Moll G, Pujades S, Bang S, Kim J, Black MJ, Lee S-H. Data-driven physics for human soft tissue animation. ACM Trans Graph 2017;36(4):54:1–54:12. http://dx.doi.org/10.1145/3072959.3073685, URL https://dl.acm.org/doi/10.1145/3072959.3073685.

[44] Bender J, Erleben K, Trinkle J. Interactive simulation of rigid body dynamics in computer graphics. Comput Graph Forum 2014;33(1):246–70. http://dx.doi.org/10.1111/cgf.12272.

[45] Bullet Physics Library. Bullet physics library. 2019, URL https://pybullet.org/wordpress/.

[46] Dickinson C. Learning game physics with bullet physics and OpenGL. Birmingham - Mumbai: Packt Publishing Ltd.; 2013, URL https://ebin.pub/learning-game-physics-with-bullet-physics-and-opengl-1783281871-9781783281879.html.

[47] Holden D, Duong BC, Datta S, Nowrouzezahrai D. Subspace neural physics: fast data-driven interactive simulation. In: Proceedings of the 18th annual ACM SIGGRAPH/eurographics symposium on computer animation. SCA '19, New York, NY, USA: Association for Computing Machinery; 2019, p. 1–12. http://dx.doi.org/10.1145/3309486.3340245.

[48] Radovic A, Williams M, Rousseau D, Kagan M, Bonacorsi D, Himmel A, Aurisano A, Terao K, Wongjirad T. Machine learning at the energy and intensity frontiers of particle physics. Nature 2018;560(7716):41–8. http://dx.doi.org/10.1038/s41586-018-0361-2, Number: 7716 Publisher: Nature Publishing Group. URL https://www.nature.com/articles/s41586-018-0361-2.

[49] Nasrollahi S. Real-time physics simulations and machine learning. 2021, URL https://seyedhn.medium.com/real-time-physics-simulations-and-machine-learning-90e6e1ef1b3d.

[50] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8):1735–80. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[51] Ramos AG, Oliveira JF, Lopes MP. A physical packing sequence algorithm for the container loading problem with static mechanical equilibrium conditions. Int Trans Oper Res 2016;23(1–2):215–38. http://dx.doi.org/10.1111/itor.12124.

[52] Daios A, Kladovasilakis N, Kostavelis I. Mixed palletizing for smart warehouse environments: Sustainability review of existing methods. Sustainability 2024;16(3):1278. http://dx.doi.org/10.3390/su16031278, Number: 3 Publisher: Multidisciplinary Digital Publishing Institute. URL https://www.mdpi.com/2071-1050/16/3/1278.

[53] Mazur PG. [Dataset] loading stability benchmark for pallet loading problems. 2024, http://dx.doi.org/10.5281/zenodo.11281305, URL https://zenodo.org/records/11281305.

[54] Ali S, Ramos AG, Carravilla MA, Oliveira JF. Heuristics for online three-dimensional packing problems and algorithm selection framework for semi-online with full look-ahead. Appl Soft Comput 2024;151:111168. http://dx.doi.org/10.1016/j.asoc.2023.111168, URL https://www.sciencedirect.com/science/article/pii/S1568494623011869.

[55] Kingma DP, Ba J. Adam: A method for stochastic optimization. 2014, CoRR URL https://www.semanticscholar.org/paper/Adam%3A-A-Method-for-Stochastic-Optimization-Kingma-Ba/a6cb366736791bcccc5c8639de5a8f9636bf87e8.

[56] Castellucci PB, Toledo FMB, Costa AM. Output maximization container loading problem with time availability constraints. Oper Res Perspect 2019;6:100126. http://dx.doi.org/10.1016/j.orp.2019.100126, URL https://www.sciencedirect.com/science/article/pii/S2214716019300910.