IT Journal Research and Development (ITJRD)
Vol. x, No. x, Month 20xx, pp. 1 – 14, E-ISSN: 2528-4053 |P-ISSN: 2528-4061
DOI : 10.25299/itjrd.2018.volx(x).xxxxx

Interactive 3D Bin Packing System for Optimized Container Loading Operations

Muhamad Saladin Eka Septian[1], Dina Oktafiani[2], and Roni Andarsyah[3]
Department of Informatics Engineering, Universitas Logistik dan Bisnis Internasional[1,2,3]
ekaseptian.c@gmail.com[1], dinaoktafiani04@gmail.com[2], roniandarsyah@ulbi.ac.id[3]

Article Info

Article history:
Received xx, 20xx
Revised xx, 20xx
Accepted xx, 20xx

Keyword:
3D bin packing
container loading
logistics optimization
heuristic algorithm
web visualization

ABSTRACT

Container loading operations face persistent efficiency challenges due to reliance on operator experience without decision support tools, while existing algorithmic solutions lack practical guidance for warehouse staff. This research develops a web-based container loading planning system integrating a 3D bin packing algorithm with interactive visualization. The system employs a layered architecture separating the calculation engine from the data management backend, with WebGL-based 3D visualization providing step-by-step loading guidance accessible through standard web browsers. Evaluation across five heterogeneous item scenarios (50–300 items with varying dimensions and types) demonstrates consistent 100% placement success rate with volume utilization reaching 55.26% in the most complex scenario. Computation time exhibits quadratic growth, remaining under 40 seconds for 300 items, suitable for offline planning operations. Algorithm comparison reveals the Bigger First sorting strategy achieves 43% higher utilization than Smaller First, while stability checking adds only 8% computational overhead. Results validate the feasibility of web-based 3D bin packing systems for practical logistics operations and demonstrate that integrating step-by-step playback visualization effectively bridges the gap between algorithmic optimization and physical execution requirements.

Corresponding Author:
Muhamad Saladin Eka Septian
Department of Informatics Engineering
Universitas Logistik dan Bisnis Internasional
Jl. Sariasih No. 54, Sarijadi, Bandung, West Java, Indonesia
ekaseptian.c@gmail.com

1. INTRODUCTION

Container loading represents a critical activity in logistics operations that directly influences transportation efficiency and operational costs. The global container shortage intensified since 2021 due to the COVID-19 pandemic, coupled with disruptions such as the 2024 Red Sea shipping crisis

that caused freight rate increases up to 173% for Asia-Europe routes, has further emphasized the importance of optimal space utilization [1]. Optimal loading maximizes space utilization, reduces trip frequency, and lowers operational expenses and carbon emissions [2]. However, loading operations for heterogeneous items remain predominantly manual [3], with shipping managers often planning arrangements based solely on personal experience due to the absence of decision support tools. This manual approach produces efficiency highly dependent on individual operator capabilities and cannot guarantee consistent load stability—a safety concern particularly critical in logistics operations [3]. The container loading problem is formally known as the Three-Dimensional Bin Packing Problem (3D-BPP), an optimization challenge of placing items into containers to minimize empty space or the number of required containers. 3D-BPP is classified as NP-hard, meaning computational complexity increases exponentially as problem size grows [4]. Complexity escalates when considering practical constraints such as maximum weight limits, load stability requirements, item orientation restrictions, and load-bearing capacity [5]. The problem becomes particularly challenging with numerous heterogeneous item types of highly variable dimensions, as commonly encountered in diverse logistics sectors from e-commerce to manufacturing operations [4]. Various approaches have been developed to solve 3D-BPP, including classical heuristic algorithms that place items sequentially according to predetermined rules [6], layer-building methods that construct horizontal layers then stack them vertically [2], and metaheuristic approaches using genetic algorithms that respect practical stability constraints [3, 1]. Recent Deep Reinforcement Learning advances have shown improvements in space utilization and computational efficiency [7, 8, 9].

Despite these algorithmic advances, a critical gap remains in translating optimization solutions into practical operational guidance. Existing validation and visualization tools are desktop-based and primarily targeted at researchers for algorithm verification rather than operational field use [5]. While web-based 3D visualization for container loading has been demonstrated [10], these implementations lack integrated step-by-step loading guidance that connects algorithmic results with physical execution sequences for warehouse operators. Furthermore, algorithm evaluation often relies on randomly generated datasets that inadequately reflect the heterogeneous item distributions characteristic of real warehouse operations [11]. A gap exists for web-based systems that integrate bin packing algorithms with interactive 3D visualization, provide step-by-step loading guidance accessible without specialized software installation, and validate performance using realistic heterogeneous scenarios reflective of actual logistics operations.

This research develops a web-based container loading planning system with three primary objectives. First, to design and implement a layered architecture that separates the 3D bin packing computation engine—implementing stability checking and gravity simulation—from data management and presentation layers. Second, to develop interactive 3D visualization using WebGL-based rendering that provides step-by-step loading guidance, enabling operators to visualize the item placement sequence chronologically. Third, to evaluate system effectiveness using realistic heterogeneous item scenarios, measuring achieved space utilization, computation time scalability, and algorithm configuration impact on loading performance. Unlike existing desktop-based tools targeted at researchers [5] or standalone visualization implementations [10], this system provides an integrated solution accessible through standard web browsers without specialized software installation, addressing the operational gap where algorithmic solutions lack practical guidance for warehouse staff [2, 3].

2. RESEARCH METHOD

The methodology comprises formulating the 3D bin packing problem with practical constraints, designing a layered system architecture that separates computation from data management, integrating

the packing algorithm, and defining evaluation scenarios.

2.1. Problem Formulation

The Three-Dimensional Bin Packing Problem (3D-BPP) is a combinatorial optimization problem that involves placing a set of three-dimensional items into containers (bins) to maximize space utilization [4]. Given that 3D-BPP is NP-hard [4], exact methods are computationally intractable for industrial-scale problems involving hundreds of heterogeneous items. Therefore, a heuristic approach is applied to obtain efficient solutions within practical time constraints, following common practice in recent studies [3, 12, 2].

In the mathematical model, a container is defined with dimensions length (L), width (W), and height (H), along with maximum load capacity M. For each item i from a total of n items, the following attributes are defined: dimensions (l_i, w_i, h_i), volume v_i, and mass m_i. Decision variables include $\eta_i \in \{0, 1\}$ indicating whether item i is successfully loaded, position coordinates (x_i, y_i, z_i), and orientation indicator $r_{i,p}$ representing the selected rotation.

The primary objective is expressed in Equation 1:

$$\max \sum_{i=1}^{n} v_i \cdot \eta_i \tag{1}$$

The objective is to maximize the total volume of all items successfully placed ($\eta_i = 1$) within a single container. Maximizing loaded volume minimizes void space, reducing transportation cost per unit.

Volume utilization (U_v) serves as the primary effectiveness metric, formulated in Equation 2:

$$U_v = \frac{\sum_{i=1}^{n} v_i \cdot \eta_i}{L \times W \times H} \times 100\% \tag{2}$$

This equation represents the loading efficiency ratio in percentage form. The numerator is the total volume of successfully loaded items, while the denominator is the container capacity ($L \times W \times H$). A value near 100% indicates efficient arrangement.

Weight utilization (U_w) measures the ratio of loaded weight to maximum load capacity:

$$U_w = \frac{\sum_{i=1}^{n} m_i \cdot \eta_i}{M} \times 100\% \tag{3}$$

This metric is important for ensuring that load capacity is efficiently utilized without exceeding safety limits, a key consideration in logistics operations [12]. Fill rate (F) measures the ratio of successfully loaded items to total requested items:

$$F = \frac{n_{loaded}}{n_{total}} \times 100\% \tag{4}$$

To ensure that generated solutions are not only volume-optimal but also physically feasible and stable, the model incorporates six operational constraints. Following the constraint categorization established in the literature [5, 3], Table 1 summarizes these constraints. The stability constraint receives particular attention, requiring a minimum support area (θ) of 75% of the item's base area to prevent shifting during transportation, consistent with the Minimal Supporting Area approach in 3D-BPP research [3, 5]. The load bearing constraint addresses the physical strength limitations of stacked items, preventing damage during transport [5].

Table 1. Operational Constraints in 3D-BPP

Constraint	Formulation and Description
Volume	$\sum_{i=1}^{n} v_i \cdot \eta_i \leq L \times W \times H$. Total item volume does not exceed container capacity [3].
Mass	$\sum_{i=1}^{n} m_i \cdot \eta_i \leq M$. Total weight does not exceed maximum load capacity [5].
Orientation	$\sum_{p=1}^{6} r_{i,p} = 1, \ \forall i$. Each item occupies exactly one of six orthogonal rotation orientations [3].
Non-overlap	$(x_i + d_{ix} \leq x_j) \vee (x_j + d_{jx} \leq x_i) \vee \ldots, \ \forall i \neq j$. Items must not occupy the same space [5].
Stability	$z_i = 0 \ \vee \ \sum_{j \in S_i} \text{OverlapArea}(i,j) \geq \theta \cdot A_i$. Items must rest on the floor or be supported by at least 75% of base area [3].
Load Bearing	$D_i \leq R_i$, where $D_i = \sum_{j \in C_i} (m_j + D_j)$. Cumulative load above an item must not exceed its load-bearing capacity [5].

2.2. System Architecture

This system employs a layered architecture to separate operational data management from computationally intensive processes. This architectural approach ensures scalability for complex loading calculations and aligns with modern Transportation Management System (TMS) design principles, where separation of concerns enables integration with existing enterprise systems [13]. The system comprises four components as illustrated in Figure 1: Web Frontend, Backend API, Database, and Packing Service.

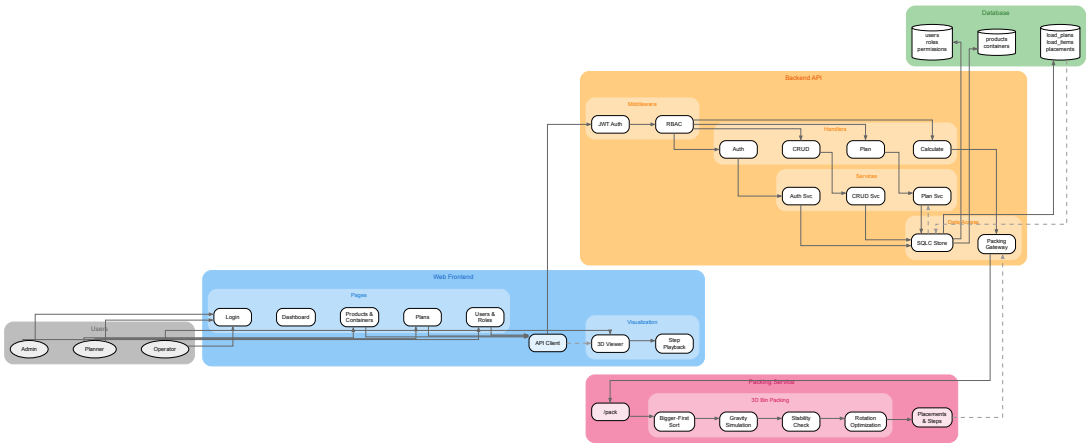

Figure 1. Container loading planning system architecture

The **Web Frontend** handles user interaction and visualization of loading results. Three user roles are supported: Admin for user management and access configuration, Planner for creating and managing loading plans, and Operator for viewing loading guidance during physical execution. An interactive 3D visualization module was implemented using WebGL-based rendering. Unlike static visualization tools intended for researchers [5], this system assists field operators in executing physical loading plans through a step playback feature.

The **Backend API** serves as the system gateway, handling user authentication through JSON

Web Token (JWT) and authorization through Role-Based Access Control (RBAC). The handler layer receives HTTP requests and forwards them to the service layer for business logic processing. The data access layer interacts with the database or invokes external services through the Packing Gateway.

The **Database** stores all persistent system data, organized into three categories: authentication and RBAC data (`users`, `roles`, `permissions`), master data (`products`, `containers`), and planning data (`load_plans`, `load_items`, `plan_placements`).

The **Packing Service** operates as a separate algorithmic calculation engine. This separation ensures that business logic in the main API remains unaffected by intensive 3D-BPP computation. The service encapsulates the 3D bin packing library and communicates with the backend through REST protocol. Calculation results, consisting of a placement list with position coordinates and step sequence numbers, are returned to the backend for storage and visualization display (Figure 2).

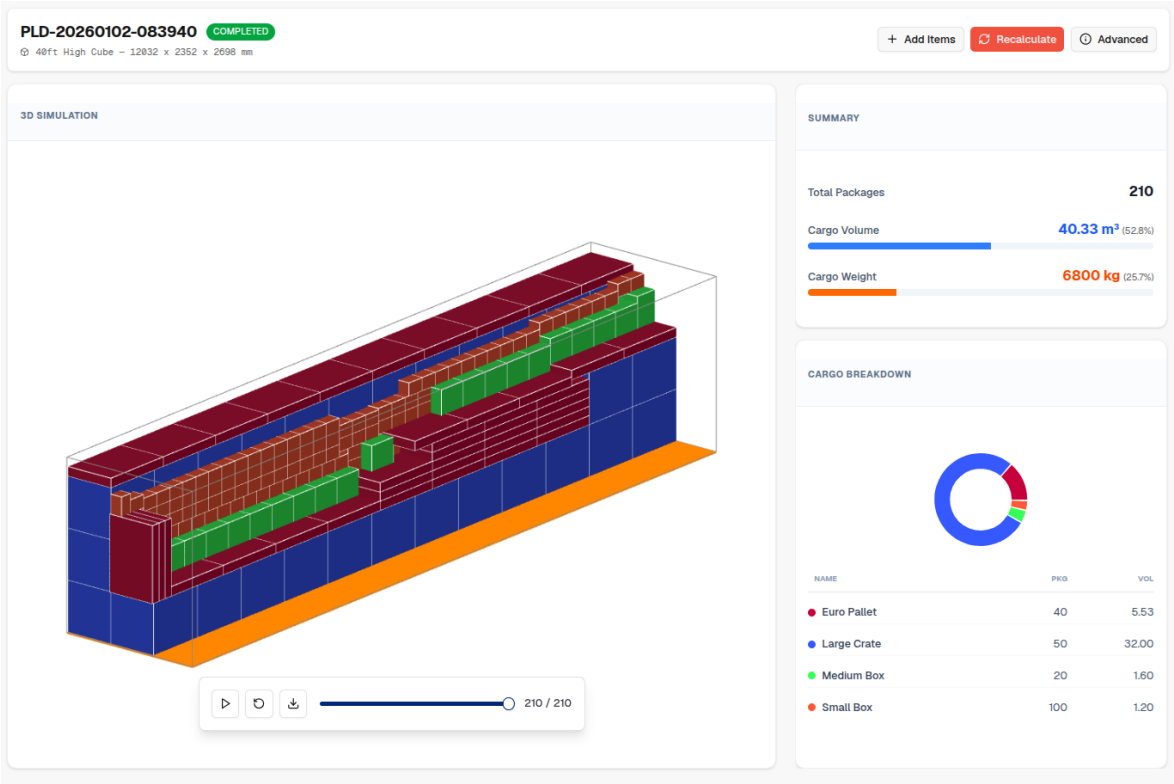

Figure 2. 3D visualization interface with playback controls and loading summary

The visualization design represents items as BoxGeometry objects with color codes distinguishing product types. A key feature is the loading sequence playback mechanism, which uses the `step_number` variable from calculation results. This mechanism displays items in placement order, enabling operators to follow loading guidance intuitively. This design addresses the operational gap where algorithmic solutions often lack practical loading sequence information for warehouse staff [5, 3].

2.3. Packing Algorithm Integration

The packing service integrates heuristic algorithms into the system workflow through a constructive heuristic approach based on sequential item placement with candidate position evaluation

[4]. The algorithm generates candidate positions, analogous to extreme points in the literature [12], where items can be placed, then selects the optimal position based on placement criteria. This sequential placement approach inherently exhibits $O(n^2)$ time complexity, as each new item must be checked for collisions against all previously placed items [7]. While advanced data structures such as stacking trees can reduce this to $O(n \log n)$ [7], the standard implementation provides sufficient performance for the target use case of offline planning with hundreds of items.

The primary sorting strategy is **Bigger First**, where items are sorted by volume in descending order before sequential placement. This strategy prioritizes larger items to establish a stable foundation and ensure efficient space utilization from the beginning of the loading process [4]. Larger items are difficult to fit later, making early placement beneficial. For each item in the sorted sequence, the algorithm evaluates candidate positions and selects the placement that minimizes wasted space while satisfying all constraints.

To evaluate the contribution of individual algorithm components and sorting strategies, three configuration variants are tested. The baseline configuration, **Bigger First with Stability** (BF+S), sorts items by descending volume with stability checking enabled. The second variant, **Bigger First without Stability** (BF-S), disables stability checking to isolate its impact on utilization and computation time. The third variant, **Smaller First with Stability** (SF+S), reverses the sorting order to ascending volume, enabling assessment of how sorting strategy affects packing quality.

Algorithm integration is implemented through a Python service that encapsulates functions from the 3D bin packing library. Each calculation activates three features: **Fix Point** simulates gravity to position items at the lowest valid vertical location; **Check Stable** validates sufficient support area from underlying items when enabled; and **Rotation Optimization** evaluates six orthogonal orientations to identify the optimal spatial fit. The data transformation flow from request to result is illustrated in Figure 3.

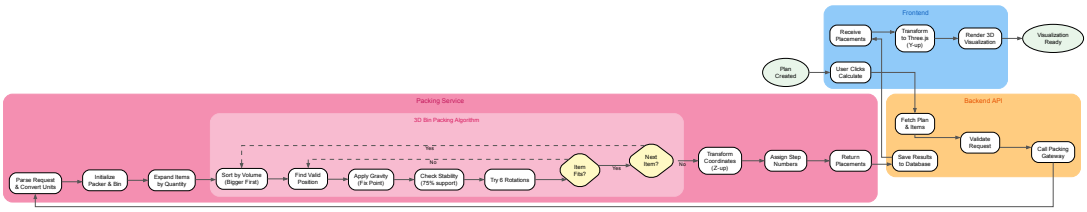

Figure 3. Loading transformation and calculation process flow

Coordinate consistency between system components requires attention. The calculation library uses the Y axis for height, while the system API uses the Z axis following logistics conventions. The Three.js frontend uses the Y-up convention. These differences are reconciled through coordinate transformation in the packing service and frontend rendering layers. The complete integration procedure is summarized in Algorithm 1.

2.4. Evaluation Design

System performance is evaluated across various loading complexity levels. Test scenarios follow literature recommendations for logistics algorithm testing, which emphasize the importance of heterogeneous item sets that reflect real-world order characteristics [11]. Five scenarios (S1–S5)

Algorithm 1: Bin Packing Algorithm Integration Procedure

Input: container dimensions (L, W, H, M), item list, algorithm options (sorting_strategy, stability_check)
Output: sorted placement list with step numbers

```
Convert all item and container dimension units to centimeters;
Initialize Packer object and add container as Bin;
foreach item in request list do
    Add item to Packer queue based on its quantity;
Execute Packer.pack(sorting_strategy, fix_point=true, check_stable=stability_check);
foreach successfully loaded item in placement order do
    Perform coordinate axis transformation for API convention;
    Calculate rotation_code from item orientation;
    Assign step_number based on placement sequence;
    Append to placement result collection;
return placements in JSON format with coordinates and step numbers;
```

are structured based on increasing item counts and product heterogeneity levels (Table 2), enabling assessment of computation time scalability and algorithm consistency in maintaining space utilization across varying complexity [14, 3].

Table 2. System Performance Test Scenarios

Scenario	Items	Product Types	Heterogeneity	Evaluation Purpose
S1	50	1	Homogeneous	Baseline accuracy with uniform items
S2	100	2	Light	Standard operational case with limited variety
S3	150	3	Moderate	Medium complexity with increasing diversity
S4	200	4	High	High complexity stress testing
S5	300	4	Very High	Maximum scalability and algorithm robustness testing

Four metrics are used for evaluation. Volume utilization (U_v) and weight utilization (U_w), calculated using Equations 2 and 3 respectively, measure the percentage of container capacity occupied by loaded items. Computation time captures the duration from request submission to result generation in milliseconds. Fill rate (F), calculated using Equation 4, measures the proportion of successfully loaded items relative to total requested items.

Visual validation through the 3D interface verifies the absence of placement anomalies such as overlapping or floating items resulting from constraint violations. Each scenario is executed across multiple runs to assess result consistency, with mean values and standard deviations reported for quantitative metrics.

3. RESULTS AND DISCUSSION

Testing was conducted based on the scenarios defined in the methodology. Results address space utilization, computation time scalability, algorithm variant performance, and visual validation.

3.1. Experimental Setup

Testing was conducted on a system with Intel Core i5-6440HQ @ 2.60GHz processor, 16 GB RAM, and Linux operating system. The application used Python 3.11 for the packing service and Go 1.21 for the backend API. Each scenario was executed 25 times to obtain mean values and standard deviations.

The container used throughout testing was a 40-foot High Cube type with internal dimensions of 12.032 $\times$ 2.352 $\times$ 2.698 meters (volume 76.35 m^3) and maximum load capacity of 26,460 kg. Four product types with varying dimensions were defined to simulate item heterogeneity: Euro Pallet (1200$\times$800$\times$144 mm), Large Crate (1000$\times$600$\times$500 mm), Medium Box (600$\times$400$\times$400 mm), and Small Box (400$\times$300$\times$200 mm). Each scenario uses a subset of these product types as specified in Table 2.

3.2. Performance Results

Table 3 presents testing results for all five scenarios using the baseline algorithm configuration (Bigger First strategy with stability checking enabled). Measured metrics include volume utilization, weight utilization, fill rate, and computation time.

Table 3. Loading Algorithm Performance Testing Results

Scenario	Items	Vol. Util. (%)	Weight Util. (%)	Fill Rate (%)	Time (ms)
S1	50	6.29 $\pm$ 0.00	2.83 $\pm$ 0.00	100.00	320 $\pm$ 6
S2	100	8.80 $\pm$ 0.00	4.16 $\pm$ 0.00	100.00	1,684 $\pm$ 131
S3	150	24.83 $\pm$ 0.00	11.15 $\pm$ 0.00	100.00	5,213 $\pm$ 169
S4	200	35.45 $\pm$ 0.00	16.25 $\pm$ 0.00	100.00	10,826 $\pm$ 168
S5	300	55.26 $\pm$ 0.00	25.32 $\pm$ 0.00	100.00	38,343 $\pm$ 1,209

The standard deviation of 0.00 for utilization metrics reflects the deterministic nature of the algorithm, where identical inputs produce identical placement results. Only computation time exhibits variance due to system-level factors such as process scheduling and memory allocation.

The algorithm successfully placed all items (100% fill rate) across all scenarios, indicating that the 40-foot High Cube container capacity is adequate for the tested item configurations. The relatively low utilization values in scenarios S1 and S2 reflect the test design rather than algorithm inefficiency; these scenarios intentionally use fewer items than required to fill the container, allowing assessment of algorithm behavior across varying load levels. Volume utilization increases proportionally with item count, from 6.29% in S1 to 55.26% in S5. Weight utilization values consistently lower than volume utilization indicate that the configuration is volume-constrained rather than weight-constrained.

Figure 4 visualizes the comparison of volume and weight utilization for each scenario. Volume utilization consistently exceeds weight utilization with approximately a 2:1 ratio, confirming the volume-constrained characteristics of the test configuration.

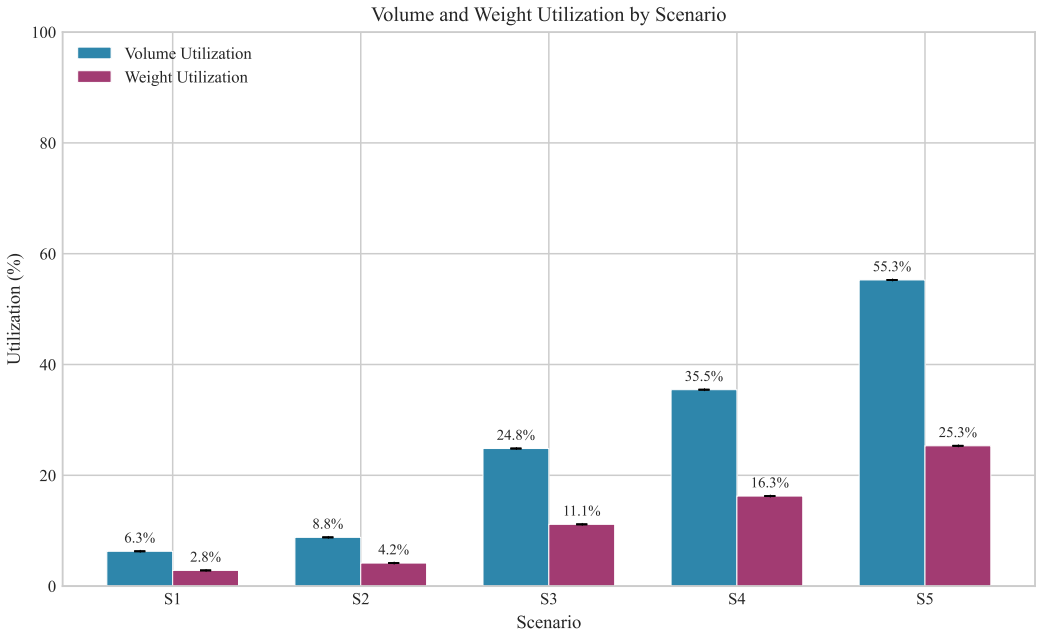

Figure 4. Comparison of volume and weight utilization in each scenario

3.3. Computation Time Analysis

An important consideration in 3D-BPP evaluation is computation time scalability with increasing item counts. Table 4 presents detailed computation time statistics, including minimum, average, and maximum values for each scenario.

Table 4. Computation Time Scalability Analysis

Scenario	Items	Min (ms)	Avg (ms)	Max (ms)
S1	50	311	320	325
S2	100	1,489	1,684	1,791
S3	150	5,012	5,213	5,391
S4	200	10,544	10,826	10,972
S5	300	36,334	38,343	39,288

Figure 5 illustrates the relationship between item count and computation time. Computation time exhibits quadratic growth ($O(n^2)$), consistent with heuristic-based bin packing algorithm characteristics [4]. For each item placement, the algorithm evaluates potential collisions with all previously positioned items, resulting in quadratic complexity.

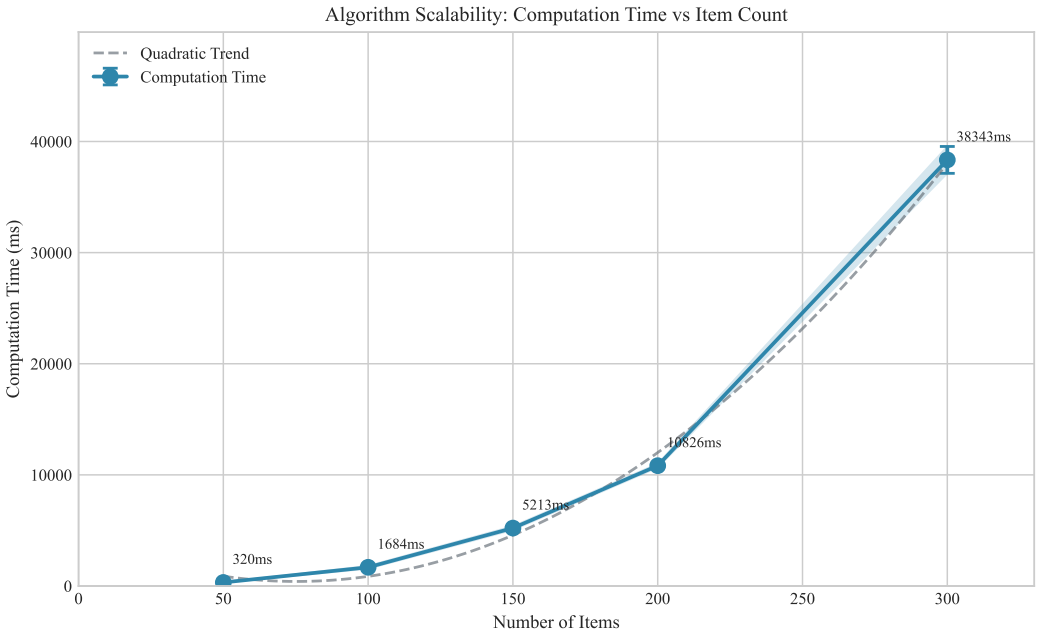

Figure 5. Computation time scalability with respect to item count

Although computation time in scenario S5 (300 items) reaches an average of 38 seconds, this value remains acceptable for offline loading planning applications. In logistics operational contexts, loading planning is typically performed before physical loading commences, making wait times under one minute acceptable for operational requirements [2].

3.4. Algorithm Variant Comparison

To assess the contribution of each algorithm component, comparative testing was performed with three configuration variants on scenario S3 (150 items). Table 5 presents comparison results between the baseline configuration (Bigger First with Stability), a variant without stability checking, and a variant using the Smaller First strategy.

Table 5. Algorithm Variant Comparison in Scenario S3 (150 items)

Variant	Vol. Util. (%)	Fill Rate (%)	Time (ms)	Items Packed
Bigger First + Stability	24.83	100.00	5,386	150
Bigger First (No Stability)	24.83	100.00	4,960	150
Smaller First + Stability	17.37	87.33	32,886	131

The comparison reveals that the Bigger First strategy substantially outperforms Smaller First, yielding 43% relatively higher volume utilization (24.83% vs. 17.37%) and achieving complete placement compared to only 87.33% fill rate. This confirms the heuristic principle that placing large items first leaves small gaps fillable by smaller items, whereas the reverse approach creates suboptimal fragmentation [4]. The Smaller First variant also requires 6.1 times longer execution (32,886 ms vs. 5,386 ms) due to increased failed placement attempts when small items occupy spaces more suitable for large items.

Disabling stability checking saves only approximately 8% computation time (4,960 ms vs. 5,386 ms) without affecting utilization results. This indicates that stability checking overhead is relatively small and worth retaining to ensure physical loading safety.

3.5. Validation Results

Placement feasibility was validated across all five scenarios comprising 800 total item placements. Three geometric validation criteria were evaluated: non-overlapping (no item intersections), boundary containment (all items within container walls), and stability compliance (minimum 75% base support area for elevated items). Validation employed computational geometric checks during the packing process combined with visual verification through the implemented 3D visualization interface (Figure 2).

Non-overlapping verification using axis-aligned bounding box collision detection confirmed zero geometric intersections across all scenarios. For the most complex scenario S5 with 300 items, pairwise collision checking validated 44,850 potential item pairs without detecting any overlaps. Boundary containment verification confirmed that all item coordinates remained within container dimensions (12,032 × 2,352 × 2,698 mm), with no items exceeding wall boundaries or floor limits. These geometric validations were performed automatically during the packing computation, with results persisted alongside placement coordinates for subsequent verification.

Stability validation confirmed that all elevated items (items with z-position ¿ 0) achieved at least 75% base support area from underlying items or the container floor, consistent with the configured `support_surface_ratio` parameter. Visual examination through the 3D visualization interface corroborated these results, with no floating or insufficiently supported items observed in any scenario. The step-by-step playback feature further verified adherence to the Bigger First sorting strategy, consistently showing large items (Euro Pallets and Large Crates) placed in early steps before smaller items filled remaining spaces across all test scenarios.

3.6. Comparative Analysis

The developed system successfully processes up to 300 heterogeneous items with 100% fill rate, achieves volume utilization up to 55.26%, and completes calculations within 38 seconds for the largest scenario (Figure 6). The system produces physically stable arrangements with minimum 75% support and provides step-by-step visual guidance to assist operators in physical loading execution.

Comparison with existing literature indicates that the achieved volume utilization (55.26% in the densest scenario) falls within expected ranges for heterogeneous cases with stability constraints. Table 6 summarizes performance metrics from related studies for contextual comparison.

Table 6. Comparison with Related Literature

Study	Method	Utilization/Fill Rate	Computation Time
Ananno & Ribeiro [3]	Multi-heuristic + GA	50–70%	Varies by order
Ma et al. [4]	Block-building + GA + SA	60–75%	Not specified
Hoa et al. [1]	GA + Wall-building	Up to 91.67% fill	Not specified
Tresca et al. [2]	MILP + Layer-building	High quality	<30s per bin
Fontaine & Minner [15]	Branch-and-repair MILP	Optimal for bin selection	30% faster than baseline
Zhao et al. [7]	DRL + Stacking tree	+10% vs baseline	Training required
This research	Heuristic (Bigger First)	55.26%	38s (300 items)

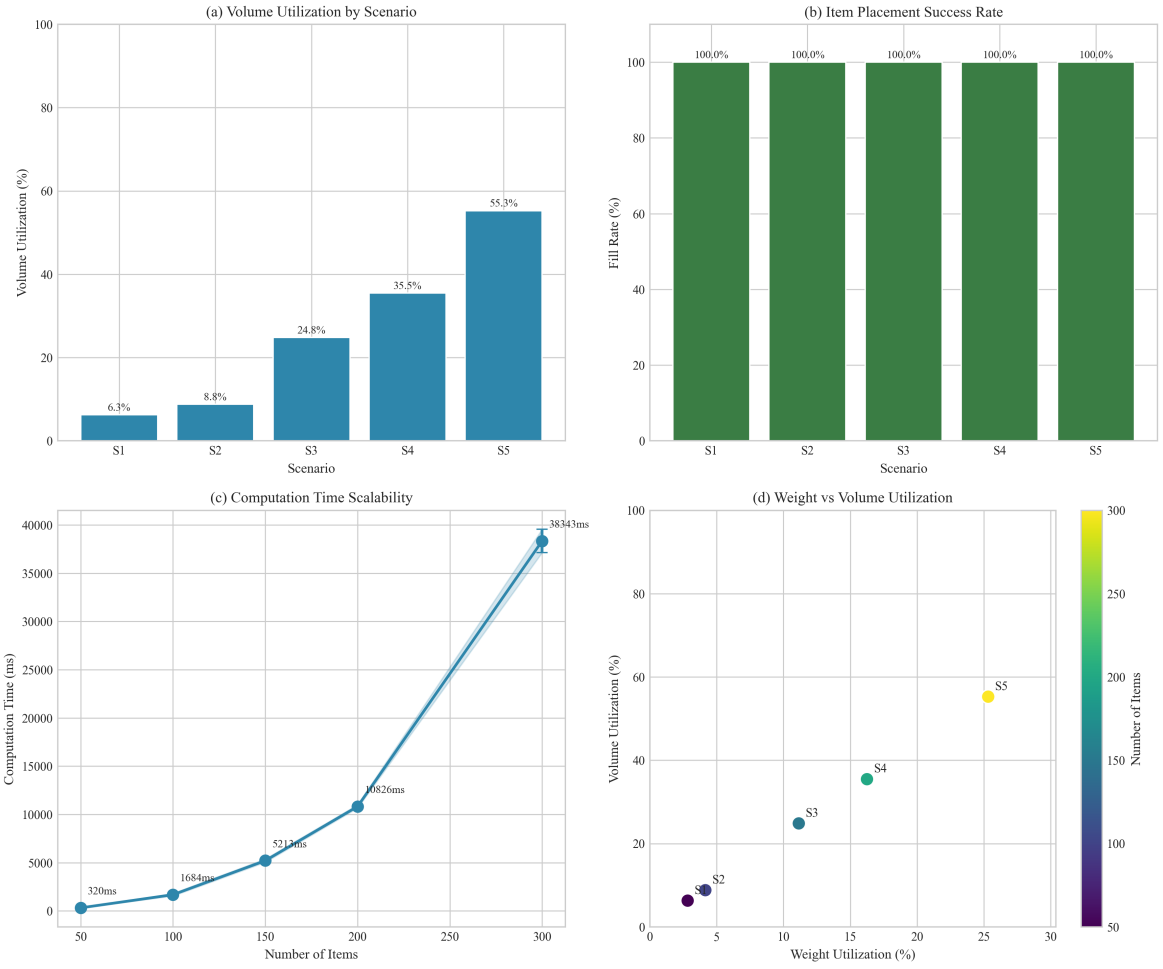

Figure 6. Performance metric summary: (a) volume utilization, (b) fill rate, (c) time scalability, (d) volume and weight utilization correlation

Ananno and Ribeiro [3] reported 50–70% utilization for industrial cases with similar stability constraints (75% minimum support area), while Ma et al. [4] achieved 60–75% on datasets with more homogeneous items. Hoa et al. [1] obtained higher fill rates (up to 91.67%) for textile industry applications using genetic algorithms with wall-building heuristics, though their configuration involved different constraint priorities including purchase order sequencing; earlier work using simulated annealing achieved approximately 85% fill rates with deadline-based prioritization [1]. The utilization difference in this research can be attributed to the high degree of dimensional heterogeneity in the test configuration, which presents greater packing challenges than homogeneous item sets. Additionally, metaheuristic approaches such as GA typically achieve higher utilization by exploring larger solution spaces at the cost of longer computation times. Direct comparison across studies is inherently limited due to variations in container types, item characteristics, and constraint specifications.

From a computational perspective, the quadratic time complexity observed aligns with findings by Tresca et al. [2], who reported similar scaling behavior for greedy heuristics in container loading problems. Their matheuristic approach achieved configurations in under 30 seconds per bin on average, comparable to the performance observed in this research. Fontaine and Minner [15] demonstrated that

MILP-based approaches for e-commerce bin selection can achieve 30% runtime reduction through branch-and-repair methods, though such exact methods are typically applied to smaller problem instances. The sub-minute computation times for 300 items compare favorably with metaheuristic approaches such as those by Zhao et al. [7], which may require longer execution times for equivalent problem sizes but can yield marginally higher utilization through more exhaustive search. This trade-off between computation speed and solution quality supports the selection of heuristic methods for interactive planning applications where rapid feedback is prioritized over marginal utilization gains.

4. CONCLUSION

This research achieved its three primary objectives in developing a web-based container loading planning system. First, a layered system architecture was successfully designed and implemented, effectively separating data management from the 3D bin packing calculation engine while enabling seamless integration through a RESTful API. The architecture supports role-based access control for administrators, planners, and operators with distinct permission levels. Second, interactive 3D visualization with step-by-step playback was developed using WebGL-based rendering, providing actionable loading guidance that bridges algorithmic optimization with physical execution requirements. The visualization enables operators to follow item placement sequences chronologically without requiring specialized desktop software installation. Third, system effectiveness was validated through five heterogeneous item scenarios with varying complexity levels, demonstrating practical applicability for offline warehouse operational planning.

Experimental evaluation revealed several key quantitative findings. The system achieved 55.26% volume utilization in the most complex scenario featuring four distinct product types and 300 item units, with placement success rates (fill rate) consistently reaching 100% across all test scenarios. The Bigger First sorting strategy combined with static stability constraints produced optimal space filling, outperforming Smaller First by 43% in volume utilization. This performance aligns with literature findings where similar deterministic approaches achieve 60–85% utilization depending on item heterogeneity. Stability checking added minimal computational overhead of approximately 8% while ensuring physical feasibility through 75% minimum base area support requirements. Computation time exhibited quadratic growth patterns with item population, reaching approximately 38 seconds for 300 units—a duration well within tolerance limits for offline planning operations where plans are prepared before physical loading begins.

Several limitations should be acknowledged. The current implementation employs a deterministic heuristic algorithm without solution space exploration capabilities that metaheuristic approaches such as genetic algorithms or simulated annealing would provide, potentially limiting achievable utilization rates. Test scenarios utilized standard rectangular items, excluding irregular or deformable shapes that occur in certain logistics contexts such as fresh food delivery. Furthermore, while the system was tested with realistic heterogeneous item configurations, no formal user study with actual warehouse operators was conducted to validate usability and practical effectiveness in field conditions. Weight constraints, although implemented, were not tested at capacity limits across scenarios.

Future research can extend this work in multiple directions. Integration with metaheuristic optimization algorithms such as genetic algorithms or simulated annealing could improve space utilization beyond deterministic heuristic limits through broader solution space exploration. Deep Reinforcement Learning approaches offer potential for adaptive online packing decisions that respond to real-time item arrivals, as demonstrated by recent advances achieving up to 10% utilization improvements over conventional methods. IoT sensor integration would enable automatic item

dimension capture and real-time tracking during loading operations, facilitating validation between planned and actual placements. A formal user study with warehouse operators would provide empirical validation of the visualization interface effectiveness and identify areas for usability improvement. Finally, mobile interface development would enhance field accessibility for operators who require step-by-step guidance during physical loading execution.

ACKNOWLEDGEMENT

The acknowledgment section is optional. The funding source of the research can be put here.

REFERENCES

[1] N. T. X. Hoa, "Optimizing container fill rates for the textile and garment industry using a 3d bin packing approach," *HighTech and Innovation Journal*, vol. 5, pp. 462–478, 2024.

[2] G. Tresca, G. Cavone, R. Carli, A. Cerviotti, and M. Dotoli, "Automating bin packing: A layer building matheuristics for cost effective logistics," *IEEE Transactions on Automation Science and Engineering*, vol. 19, pp. 1599–1613, 7 2022.

[3] A. A. Ananno and L. Ribeiro, "A multi-heuristic algorithm for multi-container 3-d bin packing problem optimization using real world constraints," *IEEE Access*, vol. 12, pp. 42 105–42 130, 2024.

[4] Y. Ma, Y. Zhou, Q. Fang, S. Xia, and W. Chen, "A three-dimensional container loading algorithm for solving logistics packing problem," *EURO Journal on Transportation and Logistics*, vol. 14, 1 2025.

[5] C. Krebs and J. F. Ehmke, "Solution validator and visualizer for (combined) vehicle routing and container loading problems," *Annals of Operations Research*, vol. 326, pp. 561–579, 7 2023.

[6] E. Silva, J. F. Oliveira, and G. Wäscher, "The pallet loading problem: A review of solution methods and computational experiments," *International Transactions in Operational Research*, vol. 23, pp. 147–172, 1 2016.

[7] H. Zhao, C. Zhu, X. Xu, H. Huang, and K. Xu, "Learning practically feasible policies for online 3d bin packing," *Science China Information Sciences*, vol. 65, 1 2022.

[8] C. C. Wong, T. T. Tsai, and C. K. Ou, "Integrating heuristic methods with deep reinforcement learning for online 3d bin-packing optimization," *Sensors*, vol. 24, 8 2024.

[9] B. Zhang, Y. Yao, H. K. Kan, and W. Luo, "A gan-based genetic algorithm for solving the 3d bin packing problem," *Scientific Reports*, vol. 14, 12 2024.

[10] . E. Poerwandono, F. Fiddin, and E. Poerwandono, "Implementasi 3d bin packing problem menggunakan algoritma tabu search," *Jurnal Sains dan Teknologi*, vol. 5, pp. 477–482, 2023.

[11] L. Ribeiro and A. A. Ananno, "A software toolbox for realistic dataset generation for testing online and offline 3d bin packing algorithms," *Processes*, vol. 11, 7 2023.

[12] K. Heßler, T. Hintsch, and L. Wienkamp, "A fast optimization approach for a complex real-life 3d multiple bin size bin packing problem," *European Journal of Operational Research*, vol. 327, pp. 820–837, 2025.

[13] R. Ranjangaonkar, "Tms in connected systems enhancing logistics efficiency and integration," *International Journal of Computer Trends and Technology*, vol. 72, pp. 152–156, 2024.

[14] P. Jomthong, T. Wongrakthai, P. Thanarungcharoenkit, C. Inthawongse, and N. Sangkhiew, "Combining vehicle routing and bin packing problem for vehicle routing planning: A case study of a chemical factory," *Journal of Applied Research on Science and Technology (JARST)*, 7 2024.

[15] P. Fontaine and S. Minner, "A branch-and-repair method for three-dimensional bin selection and packing in e-commerce," *Operations Research*, vol. 71, pp. 1846–1864, 2023.