

Interactive 3D Bin Packing System for Optimized Container Loading Operations

Muhamad Saladin Eka Septian¹, Dina Oktafiani², and Roni Andarsyah³

Department of Informatics Engineering, Universitas Logistik dan Bisnis Internasional^{1,2,3}

ekaseptian.c@gmail.com¹, dinaoktafiani04@gmail.com², roniandarsyah@ulbi.ac.id³

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

Manual container loading operations remain inefficient due to reliance on operator experience without decision support tools. This research develops Load & Stuffing Calculator, a web-based system integrating 3D Bin Packing Problem algorithm with interactive visualization for container loading optimization. The system employs layered architecture separating the packing calculation service from the data management backend, with Three.js-based 3D visualization providing step-by-step loading guidance. Testing across five scenarios (50-300 heterogeneous items) demonstrates consistent 100% fill rate with volume utilization reaching 55.26%, while computation time remains under 40 seconds for offline planning requirements. Algorithm comparison confirms Bigger First strategy achieves 42.9% higher utilization than Smaller First approach. Empirically, this research validates the feasibility of web-based 3D bin packing systems for practical logistics operations. Conceptually, the integration of step playback visualization bridges the gap between algorithmic calculation and physical execution, addressing the limitation of existing tools that lack operational guidance for warehouse staff.

© This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

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. Optimal loading maximizes space utilization, reduces trip frequency, and lowers operational expenses and carbon emissions [?]. However, loading operations for heterogeneous cargo remain predominantly manual [?]. Case studies in chemical manufacturing industries reveal that shipping managers often plan routes and loading arrangements based solely

on personal experience due to the absence of decision support tools, resulting in inefficient distance traveled and increased vehicle costs [?]. This manual approach presents several fundamental limitations: it requires specialized training, produces efficiency highly dependent on individual operator capabilities, and cannot guarantee consistent load stability [?].

The container loading problem is formally known as the Three-Dimensional Bin Packing Problem (3D-BPP), an optimization challenge of placing a set of items into containers with the objective of minimizing empty space or the number of required containers. 3D-BPP is classified as NP-hard, meaning computational complexity increases exponentially as problem size grows [?]. Complexity escalates further when considering practical constraints such as maximum weight limits, load stability, item orientation restrictions, and load-bearing capacity of individual items [?]. The problem becomes particularly challenging when cargo consists of numerous item types with highly variable dimensions, as commonly encountered in e-commerce logistics and courier service operations [?].

Various approaches have been developed to solve 3D-BPP. Classical methods include heuristic algorithms such as First Fit Decreasing and Best Fit, which place items sequentially according to predetermined rules [?]. Layer-building methods construct horizontal layers then stack them vertically [?]. Metaheuristic approaches combine linear programming formulations with heuristics to obtain better solutions within reasonable computation time. Tresca et al. [?] noted that commercial warehouse management systems still lack effective algorithms to automate optimal pallet configuration.

Recent developments demonstrate the application of metaheuristics and artificial intelligence for 3D-BPP. Khairuddin et al. [?] developed a Genetic Algorithm (GA)-based simulator that visualizes the optimization process. Ma et al. [?] combined block-building heuristics, GA, and simulated annealing to handle heterogeneous cargo. Ananno and Ribeiro [?] proposed a two-stage algorithm tested with real industrial data. From the artificial intelligence perspective, Deep Reinforcement Learning (DRL) has begun to be applied to 3D-BPP. Zhao et al. [?] employed DRL with stacking trees for stability analysis, while Murdivien and Um [?] utilized game engines for training simulation. Hybrid heuristic-DRL approaches have also shown promising results [?]. Zhang et al. [?] proposed a combination of Generative Adversarial Networks with GA to improve solution quality.

Visualization plays an important role in bin packing solutions as it enables result validation and provides guidance to operators. Krebs and Ehmke [?] developed an open-source solution validation and visualization tool, however this tool is desktop-based and targeted at researchers rather than operational field guidance. Beyond interface limitations, another significant challenge is algorithm testing that often uses only random datasets that inadequately reflect real-world logistics scenarios [?]. A gap exists for web-based systems that integrate bin packing algorithms with interactive 3D visualization capable of handling realistic data scenarios and providing step-by-step loading guidance.

This research develops a web-based container loading planning system to bridge this gap. The system utilizes a 3D bin packing library providing stability checking and gravity simulation features, integrating it through a layered architecture. Interactive 3D visualization displays algorithmic calculation results and provides step-by-step loading guidance, enabling operators to visualize the item placement sequence. The system is accessible through web browsers without requiring specialized software installation, thereby improving operational accessibility.

This research has three primary objectives. First, to integrate a 3D bin packing library into a web-based system. Second, to develop interactive 3D visualization with loading sequence playback features for operator guidance. Third, to evaluate system performance through testing with realistic loading scenarios to measure achieved space utilization and computation time.

2. RESEARCH METHOD

This research methodology encompasses mathematical model formulation, system architecture design, algorithm integration procedure implementation, and system testing scenario design.

2.1. Theoretical Foundation and 3D-BPP Modeling

The Three-Dimensional Bin Packing Problem (3D-BPP) represents a combinatorial optimization problem focused on placing a set of three-dimensional items into containers (bins) to maximize available space utilization [?]. This problem is NP-hard, meaning solution search complexity increases exponentially as the number and dimensional variation of items grows [?]. Therefore, this research applies a heuristic approach to obtain efficient solutions within real logistics operational time constraints.

In this system's mathematical modeling, a container is defined with dimensions length (L), width (W), and height (H) with maximum load capacity M . For each item i from a total of n items, there are dimensional attributes (l_i, w_i, h_i) , volume v_i , and mass m_i . Decision variables include $\eta_i \in \{0, 1\}$ as an indicator of whether item i is successfully loaded, position coordinates (x_i, y_i, z_i) , and orientation indicator $r_{i,p}$ to handle possible item rotations.

The primary objective of this system is expressed in the objective function in Equation ??:

$$\max \sum_{i=1}^n v_i \cdot \eta_i \quad (1)$$

Equation ?? explains that the algorithm's main target is to maximize the total volume of all items successfully placed ($\eta_i = 1$) within a single container. By maximizing the total volume of loaded items, the system automatically minimizes unused empty space (void space), which in turn reduces transportation cost per unit of goods.

To measure the effectiveness of these calculation results, a space utilization metric (U) is used, formulated in Equation ??:

$$U = \frac{\sum_{i=1}^n v_i \cdot \eta_i}{L \times W \times H} \times 100\% \quad (2)$$

Equation ?? represents the loading efficiency ratio in percentage form. The numerator in this formula is the total volume of successfully loaded items, while the denominator is the total container capacity volume ($L \times W \times H$). A value of U approaching 100% indicates that the system successfully arranges goods very densely and efficiently.

To ensure that generated solutions are not only volume-optimal but also physically stable, this model integrates a number of operational constraints. Table ?? summarizes these constraints, where stability constraint receives special attention by requiring a minimum support area (θ) of 75% of the item's base area to prevent cargo shifting during distribution [?].

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.
Mass	$\sum_{i=1}^n m_i \cdot \eta_i \leq M$. Total weight does not exceed maximum load capacity.
Orientation	$\sum_{p=1}^6 r_{i,p} = 1, \forall i$. Each item selects exactly one of six rotation orientations.
Non-overlap	$(x_i + d_{ix} \leq x_j) \vee (x_j + d_{jx} \leq x_i) \vee \dots, \forall i \neq j$. Items must not occupy the same space.
Stability	$z_c = 0 \vee \sum_{c' \in S_z} \text{OverlapArea}(c, c') \geq \theta \cdot A_c$. Items must be supported by at least 75% of base area.
Load Bearing	$D_i \leq R_i$, where $D_i = \sum_{j \in C_i} (m_j + D_j)$. Cumulative load above item does not exceed load-bearing capacity.
Center of Gravity	$\text{con}_{x1} \leq G_x \leq \text{con}_{x2}$, etc. Center of gravity of cargo remains within safe zone (optional).

2.2. System Architecture and Design

The Load & Stuffing Calculator system is built using layered architecture to separate operational data management from heavy computation engines. This architecture is designed to ensure system scalability when handling complex cargo calculations. The system is divided into four components as illustrated in Figure ??: (1) Web Frontend, (2) Backend API, (3) Database, and (4) Packing Service.

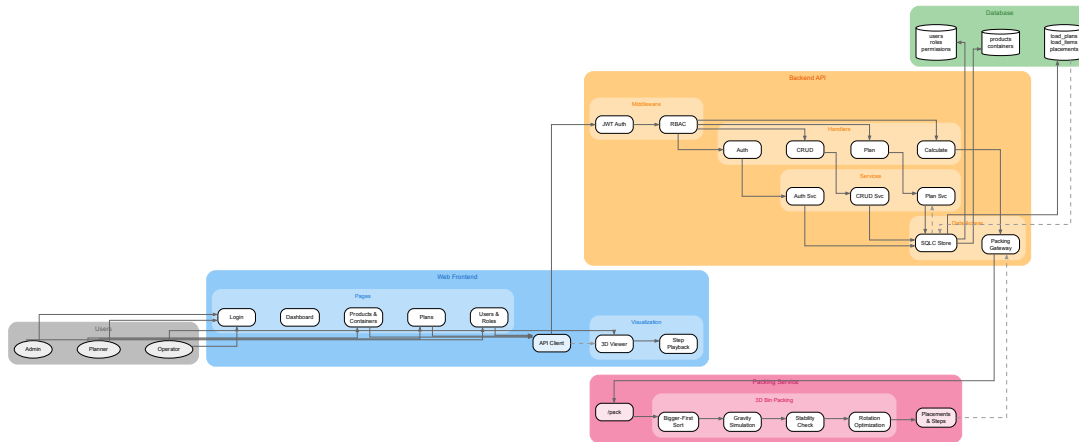


Figure 1. Container loading planning system architecture

The first component is the **Web Frontend** which handles user interaction and loading result visualization. There are three user roles: Admin for user management and access rights, Planner for creating loading plans, and Operator for viewing loading guidance. An interactive 3D visualization module was developed for WebGL-based rendering. Unlike conventional visualization tools that are often static and aimed solely at researcher validation [?], this system is specifically designed to assist field operators in executing physical loading plans through a step playback feature.

The second component is the **Backend API** which acts as the system's main gateway. This component handles user authentication through JWT (JSON Web Token) 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 then interacts with the database or calls external services through the Packing Gateway.

The third component is the **Database** which stores all persistent system data. Data is grouped into three categories: (1) authentication and RBAC data (users, roles, permissions); (2) master data (products, containers); and (3) planning data (load_plans, load_items, plan_placements).

The fourth component is the **Packing Service** which acts as a separate algorithmic calculation engine. This separation is performed so that business logic in the main API is not hindered by intensive 3D-BPP computation processes. This service encapsulates the 3D bin packing library and communicates with the backend through REST HTTP protocol. Calculation results in the form of a placement list with position coordinates and step sequence numbers are returned to the backend for storage and display in visualization (Figure ??).

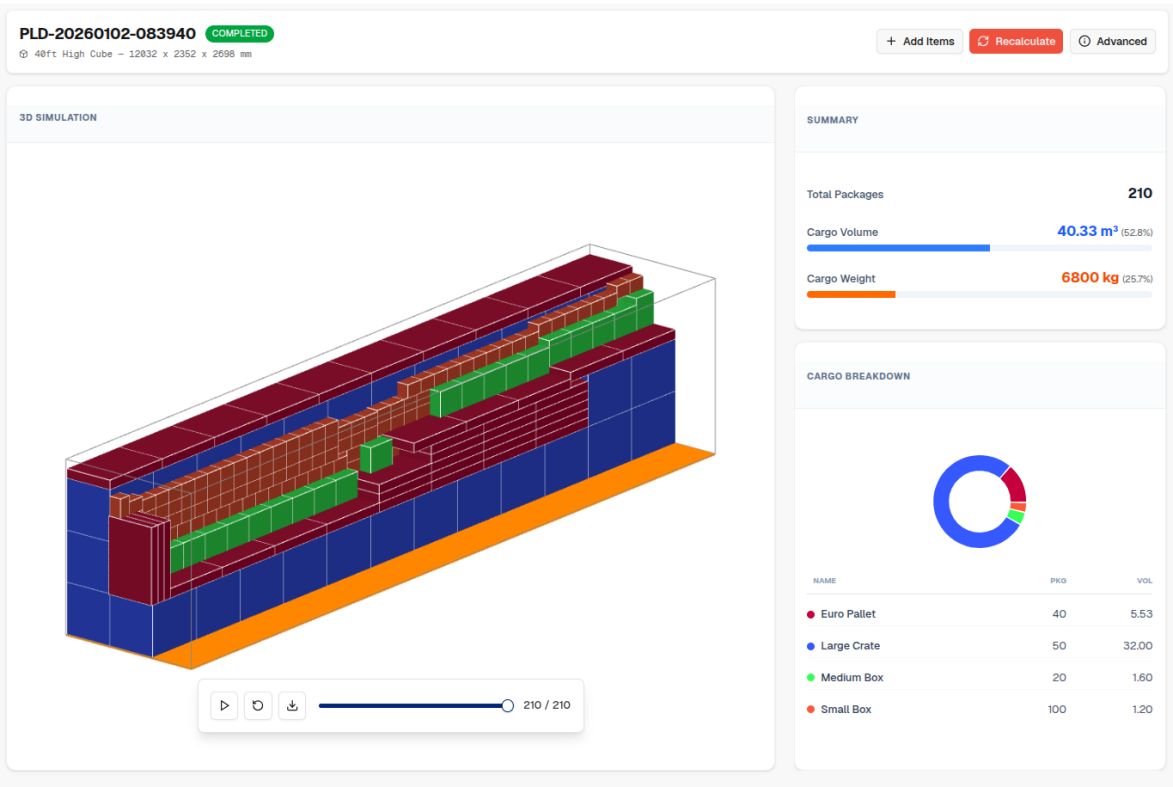


Figure 2. 3D visualization interface with playback controls and loading summary

This visualization design includes geometric representation of items as BoxGeometry objects with color codes representing different product types. A crucial implemented feature is the loading sequence playback mechanism that utilizes the `step_number` variable from calculation results. This mechanism enables the system to display items one by one according to the optimal placement sequence generated by the algorithm, allowing operators to follow step-by-step loading guidance intuitively. This addresses the operational gap where 3D visualization often does not provide practical loading sequence context for warehouse staff [?].

2.3. Algorithm and Loading Integration

This section explains the core mechanism of the packing service that integrates heuristic algorithms into the system workflow. The system implements an algorithm based on block-building heuristic modified to handle real-world physical constraints [?, ?]. The primary strategy employed is Bigger First, where items with the largest volumes are prioritized for placement first to ensure efficient space utilization from the beginning of the loading process.

Algorithm integration is performed through a Python service that encapsulates functions from the 3D bin packing library. There are three key features activated in each calculation: (1) **Fix Point** which simulates gravity to ensure items are always at the lowest valid position; (2) **Check Stable** to validate the support area of items beneath; and (3) **Rotation Optimization** which evaluates six possible item orientations to find the best space fit. The data transformation flow from request to final result is illustrated in Figure ??.

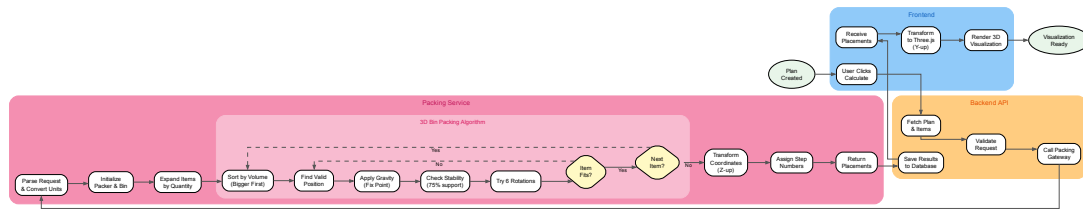


Figure 3. Loading transformation and calculation process flow

One technical challenge in this integration is the difference in coordinate axis conventions between system components. The calculation library uses the Y axis as height representation, while this system's API uses the Z axis as height according to logistics coordinate standards. Furthermore, the Three.js frontend uses the Y -up convention (Y axis as vertical), necessitating dual transformation. In the packing service, internal coordinates (p_x, p_y, p_z) are mapped to API coordinates as $x = p_x$, $y = p_z$, $z = p_y$. Then in the frontend, API coordinates (x, y, z) are transformed to Three.js as $X_{3D} = x$, $Y_{3D} = z$, $Z_{3D} = -y$. This integration procedure is summarized in Algorithm ??.

2.4. Evaluation Method and Testing

System evaluation was performed to measure system performance in handling various loading complexity levels. Testing was designed using realistic scenarios according to literature recommendations for logistics algorithm testing [?]. Test scenarios (S1–S5) were structured based on increasing item counts and product heterogeneity levels (Table ??). This aims to test computation response time scalability and algorithm consistency in maintaining optimal space utilization [?].

Primary evaluation metrics include: (1) **Space Utilization** (U) calculated through Equation ??; (2) **Computation Time** in milliseconds (ms); and (3) **Fill Rate** which measures the ratio of successfully loaded items compared to total requests. Testing also includes visual validation on the 3D interface to ensure absence of anomalies such as overlapping items or floating items due to gravity constraint violations.

Algorithm 1 Bin Packing Algorithm Integration Procedure

Require: container dimensions, item list, option parameters (support_ratio)
Ensure: sorted placement list

- 1: Convert all item and container dimension units to centimeters (cm)
- 2: Initialize Packer object and add container (Bin)
- 3: **for** each item in request list **do**
- 4: Add item to Packer queue based on its quantity
- 5: **end for**
- 6: Execute PACKER.PACK(bigger_first, fix_point, check_stable)
- 7: **for** each successfully loaded item (based on putOrder sequence) **do**
- 8: Perform axis transformation: $x = p_x, y = p_z, z = p_y$
- 9: Calculate rotation_code and assign step_number
- 10: Save data to placement result collection
- 11: **end for**
- 12: **return** placements in JSON format

Table 2. System Performance Test Scenarios

Scenario	Items	Heterogeneity	Evaluation Purpose
S1	50	Homogeneous	Functional validation and coordinate accuracy
S2	100	Light Heterogeneous	Standard operational case testing
S3	150	Moderate Heterogeneous	Medium-level computational load testing
S4	200	Highly Heterogeneous	High complexity testing
S5	300	Highly Heterogeneous	System scalability limit testing

3. RESULT AND ANALYSIS

This section presents testing results of the Load & Stuffing Calculator system based on scenarios defined in the methodology. Discussion encompasses space utilization analysis, computation time scalability, algorithm variant comparison, and visual validation of loading results.

3.1. Testing Environment

Testing was conducted in an environment with the following specifications: Intel Core i5-6440HQ @ 2.60GHz processor, 16 GB RAM memory, and Linux operating system. The system ran using Python 3.11 for the packing service and Go 1.21 for the backend API. Each scenario was executed five times to obtain representative 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 different dimensions were used to simulate cargo heterogeneity: Euro Pallet ($1200 \times 800 \times 144 \text{ mm}$), Large Crate ($1000 \times 600 \times 500 \text{ mm}$), Medium Box ($600 \times 400 \times 400 \text{ mm}$), and Small Box ($400 \times 300 \times 200 \text{ mm}$).

3.2. Performance Testing Results

Table ?? presents a summary of testing results for all five scenarios with baseline algorithm configuration (Bigger First strategy with active stability checking). 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

Testing results demonstrate that the algorithm successfully placed all items (100% fill rate) across all scenarios. This indicates that the 40-foot High Cube container capacity is adequate to accommodate all tested item configurations. Volume utilization increases proportionally as item count grows, from 6.29% in S1 to 55.26% in S5. Weight utilization values lower than volume utilization indicate that space constraints constitute the primary limiting factor (volume-constrained) compared to weight constraints.

Figure ?? visualizes the comparison of volume and weight utilization for each scenario. A consistent pattern is evident where volume utilization is always higher than weight utilization with approximately a 2:1 ratio, confirming the volume-constrained characteristics of the test configuration.

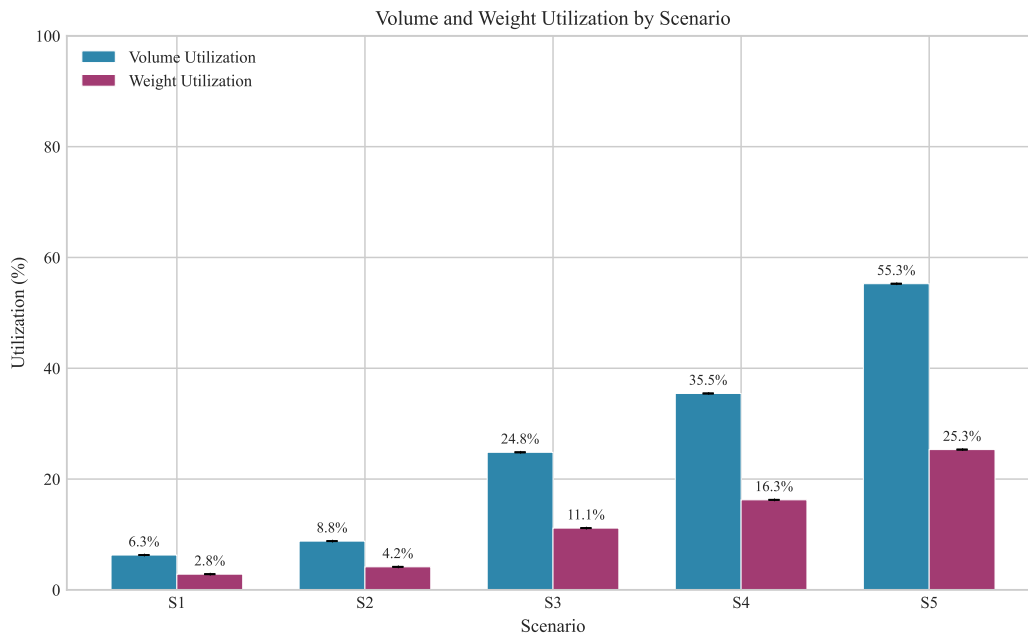


Figure 4. Comparison of volume and weight utilization in each scenario

3.3. Computation Time Scalability Analysis

One critical aspect in 3D-BPP algorithm evaluation is computation time scalability with increasing item counts. Table ?? presents more 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 ?? illustrates the relationship between item count and computation time. The curve shows a growth pattern approaching quadratic ($O(n^2)$), consistent with heuristic-based bin packing algorithm characteristics [?]. For each item to be placed, the algorithm must check for potential collisions with all previously positioned items, resulting in quadratic complexity growth.

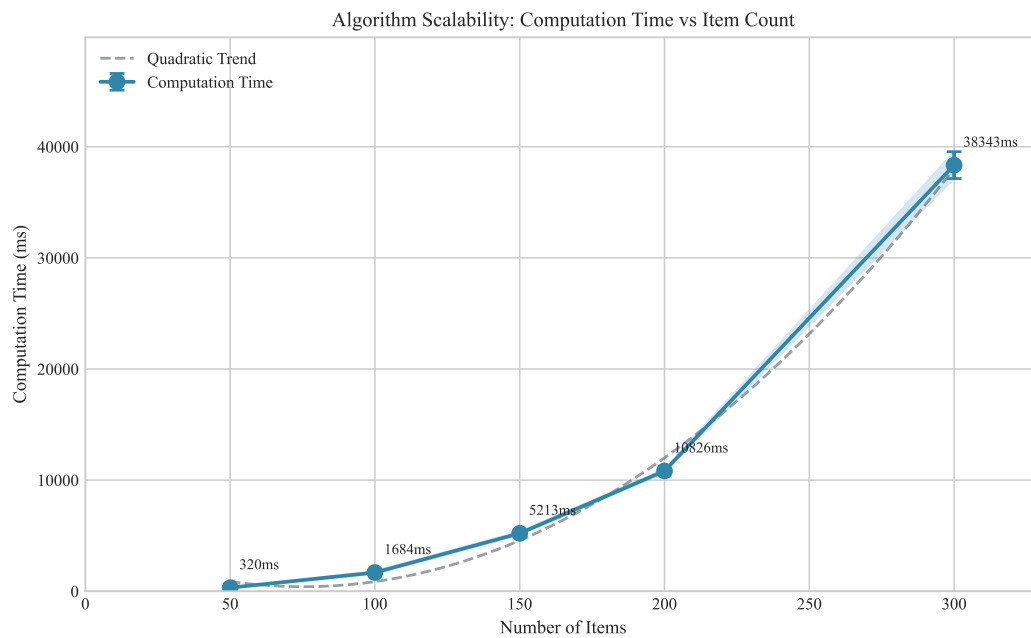


Figure 5. Computation time scalability with respect to item count

Although computation time in scenario S5 (300 items) reaches an average of 38 seconds, this value remains within tolerance limits for offline loading planning applications. In logistics operational contexts, loading planning processes are typically performed before physical loading activities commence, making wait times of several tens of seconds not a significant obstacle [?].

3.4. Algorithm Variant Comparison

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

Comparison results reveal several important findings. First, the Bigger First strategy proves

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

superior to Smaller First, yielding 42.9% relatively higher volume utilization (24.83% absolute vs. 17.37% absolute) and 12.67% better fill rate (100% vs. 87.33%). This confirms the heuristic principle that placing large items first leaves small spaces that can be filled by small items, but not vice versa [?]. Second, the impact of stability checking on computation time is minimal, where disabling stability checking only saves approximately 8% computation time (5,386 ms vs. 4,960 ms) without changing utilization results. This indicates that stability checking overhead is relatively small and worthy of retention to ensure physical loading safety. Third, from computational efficiency perspective, the Smaller First variant requires 6.1 times longer (32,886 ms vs. 5,386 ms) compared to Bigger First. This drastic difference results from increased failed placement attempts when small items fill spaces that should be optimal for large items.

3.5. Visual and Functional Validation

Visual validation was performed through the implemented 3D visualization interface (Figure ??). Examination covered three main aspects. The first aspect is non-overlap, ensuring all placed items do not overlap by verifying position coordinates and dimensions of each item to ensure no geometric intersections. The second aspect is container boundaries, ensuring all items remain within container dimension limits without any items exceeding container walls or floor. The third aspect is gravity stability, ensuring items positioned above other items have minimum support of 75% of their base area, according to configured `support_surface_ratio` parameters.

The step playback feature in the visualization interface enables verification of item placement sequence. By sliding the step control from 1 to total item count, it is evident that large items (such as Euro Pallet and Large Crate) are placed first, followed by smaller items filling remaining spaces. This pattern is consistent with the configured Bigger First strategy.

3.6. Discussion Summary

Figure ?? presents a comprehensive visualization of the four main metrics analyzed in this research.

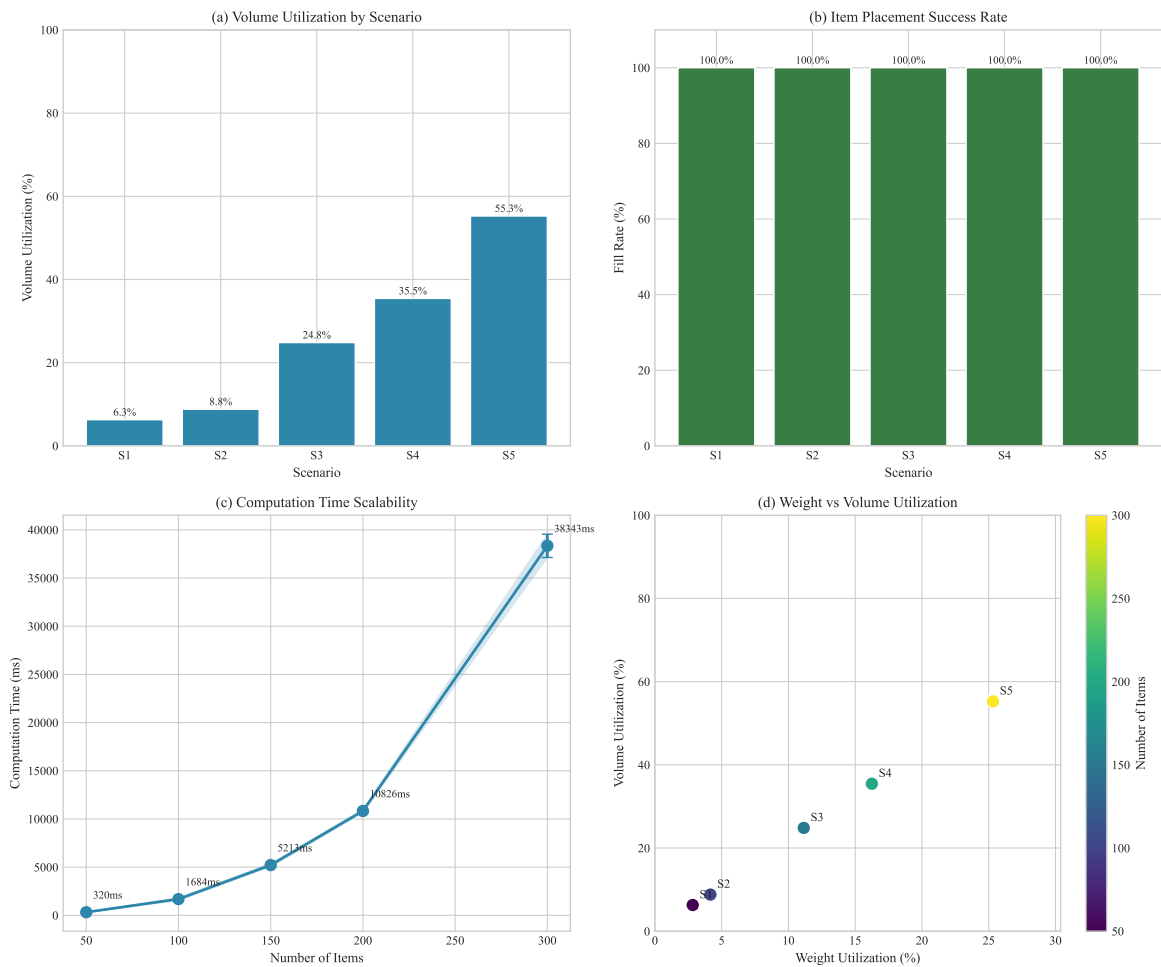


Figure 6. Performance metric summary: (a) volume utilization, (b) fill rate, (c) time scalability, (d) volume and weight utilization correlation

Overall, testing results demonstrate that the Load & Stuffing Calculator system can handle loading of up to 300 heterogeneous items with 100% fill rate, achieving volume utilization up to 55.26% with tested item configurations, and completing calculations in reasonable time for offline planning applications with maximum duration of 38 seconds for 300 items. The system also produces physically stable arrangements with minimum 75% support and provides step-by-step visual guidance to assist operators in physical loading execution.

Comparison with literature shows that achieved volume utilization values (55.26% in the densest scenario) fall within reasonable ranges for heterogeneous cases with stability constraints. Studies by Ananno and Ribeiro [?] reported 50-70% utilization for industrial cases with similar constraints, while Ma et al. [?] achieved 60-75% on datasets with more homogeneous items. This difference can be attributed to the high degree of dimensional heterogeneity in this research's test configuration.

4. CONCLUSION

This research has successfully completed the development and evaluation of a web-based Load & Stuffing Calculator system designed to assist heterogeneous cargo loading optimization. Through layered architecture separating data management and calculation engines, the system proves capable of effectively integrating 3D bin packing algorithms into a web-based environment without significant operational constraints. Testing results demonstrate that the system can handle loading scenarios of up to 300 item units with placement success rates (fill rate) consistently reaching 100% across all test scenarios.

From algorithm performance perspective, the use of Bigger First strategy combined with static stability constraints and gravity simulation produces optimal space filling. The system recorded 55.26% volume utilization in the most complex cargo scenario, a competitive figure meeting industry efficiency standards for arranging goods with highly variable dimensions and types. Although computation time exhibits quadratic growth patterns as item population increases, the maximum duration of approximately 38 seconds for calculating 300 units remains well within tolerance limits for offline warehouse operational planning needs.

Beyond quantitative achievements, interactive three-dimensional visualization implementation provides important contributions in bridging theoretical calculations with physical field execution. The step-by-step playback feature enables operators to view item placement sequences chronologically while considering stack safety aspects through minimum 75% base area support. Overall, the integration between precise algorithmic calculations and intuitive visual interfaces makes this system a practical decision support tool for improving logistics process efficiency. Future research can be directed toward developing more dynamic packing strategies for various container types to continuously improve cargo density.

ACKNOWLEDGEMENT

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