Enhancing Military Load Planning: A Prioritized 2-D Orthogonal Packing Approach

William K. Kirschenman[a], H. Sebastian Heese[b], Michael G. Kay[a], Russell E. King[a,c], Brandon M. McConnell[a,c]

[a]*Department of Industrial and Systems Engineering, North Carolina State University, Campus Box 7906, Raleigh, NC, United States*
[b]*Poole College of Management, North Carolina State University, 2300 Nelson Hall, Raleigh, NC, United States*
[c]*Center for Additive Manufacturing and Logistics (CAMAL), North Carolina State University, 915 Partners Way, Box 7906, Raleigh, NC, United States*

Abstract

Military combat loading requires arranging equipment on maritime transport vessels to enable rapid, prioritized off-loading while maintaining unit cohesion and vessel stability. Related maritime deck-loading settings can involve similar access, grouping, and balance requirements. This paper extends a prioritized two-dimensional orthogonal packing framework to incorporate global load balancing requirements alongside existing prioritization objectives.

We study three solution approaches for this globally constrained problem: a monolithic mixed-integer linear programming (MILP) approach, a sliding-window matheuristic, and a sliding-window matheuristic with in-stride load balancing penalties. For any sliding-window solution that fails to achieve both feasible packing and load balancing in the initial stage, we develop a universal post-processing strategy that selectively relaxes and re-optimizes item positions to achieve balance with minimal disruption to the prioritized layout. Computational experiments demonstrate that the matheuristic approaches fundamentally outperform the monolithic MILP approach in load balance reliability, solution quality, and computational efficiency, providing practical guidance for integrating automated optimization into military load planning systems. Among these, the simpler sliding-window matheuristic followed by post-processing repair emerges as the recommended practical configuration, offering the strongest overall combination of balance success, solution quality, and runtime, while the in-stride variant remains a narrower alternative when direct first-stage balance attainment is paramount. The matheuristic pipelines generate high-quality, load-balanced solutions for single-vessel scenarios within a few minutes on average, enabling rapid evaluation of multiple loading configurations during time-critical deployment planning.

Keywords: Integer programming, Facilities planning and design, Packing, Combinatorial optimization, Prioritization, Logistics

1. Introduction

Efficiently loading military equipment onto transport vessels is a crucial yet complex task in large-scale combat operations (LSCO). The challenges associated with traditional manual load planning were starkly highlighted during operations like Desert Storm, where the sheer scale of the effort impeded operations [1]. In response, modern software tools such as the Integrated Computerized Deployment System (ICODES), the Department of Defense (DoD) program of record for load planning, offer significant assistance [2, 1]. ICODES provides invaluable functionality by performing automated feasibility checks for a load plan against numerous physical and safety constraints. However, achieving a

layout that is also optimized for specific combat loading priorities—such as tactical sequencing for rapid offload—often still relies on extensive manual adjustments by the planner. Although military combat loading is the primary application and data source studied here, the same fixed-bin prioritized-layout question also appears in broader deck-loading and roll-on/roll-off settings where the assigned cargo set is given and balance feasibility remains mandatory. To bridge this gap between feasibility checking and tactical optimization, we propose an automated approach that generates high-quality initial layouts.

This research builds directly on the prioritized two-dimensional orthogonal packing framework of Kirschenman et al. [3], which introduced the weighted access-point proximity and cohesion objective for the unbalanced variant. Here, the central extension is hard load balancing. The decision stage studied here is not classical bin minimization or strip compaction, since the bin and assigned cargo set are fixed upstream, and the remaining task is the internal arrangement of those assigned items within a single bin. We retain the inherited prioritization structure, but add center-of-gravity feasibility constraints that couple all item placements through a shared global requirement. Because vessel-specific stability data are not available for our test fleet, throughout this paper the target center of gravity is set at the deck's geometric center, with permissible deviations expressed as fixed percentages of the vessel dimensions. We also incorporate fixed deck obstacles and material separation, which are operationally important in systems such as ICODES, and we show how the framework can accommodate additional operational constraints alongside load balancing.

This global coupling presents a significant computational challenge, particularly for the matheuristic solution methods that proved highly effective for the standard prioritized packing problem [3]. A monolithic Single MILP approach, while exact in theory, is unlikely to scale to realistic problem sizes. Therefore, the central investigation of this paper is a methodological comparison of three solution techniques: a Single MILP approach and two sliding-window (SW) matheuristic variants that differ in their treatment of global balancing constraints. Both iteratively employ exact mathematical programming solvers on subproblems within an algorithmic framework [4]. We evaluate a standard SW Iterative method that optimizes only the prioritization objective in Stage 1, and a SW In-Stride variant that temporarily adds load balancing penalties to the prioritization objective within the matheuristic's subproblems to proactively guide the layout toward constraint satisfaction. For either SW approach, any Stage 1 solution that is not both feasibly packed and load balanced is sent to a universal post-processing strategy—using iterative MILP adjustments—to achieve constraint satisfaction with minimal disruption to the prioritized layout. Our work aims not to replace planning tools like ICODES, but to provide a complementary optimization engine capable of automatically generating high-quality, balanced, and tactically sound layouts that can serve as superior starting points for planners, drastically reducing manual effort and planning time.

The primary contributions of this paper are threefold. First, we extend the inherited prioritized packing formulation with hard center-of-gravity feasibility constraints and show why that addition changes the problem from a purely prioritized packing task into a globally coupled feasibility problem. Second, we develop and evaluate the balance-handling solution package needed to make that balanced variant workable in practice, namely balance-guided in-stride placement and limited-disruption post-processing repair. Third, we provide a computational analysis of the resulting trade-offs in success rate, solution quality, and runtime; this analysis identifies the simpler SW Iterative plus post-processing pipeline as the preferred practical configuration, with in-stride guidance retained as a narrower option for settings that prioritize direct first-stage balance attainment. Obstacle avoidance and material separation are also part of the modeling contribution, while the main computational study centers on

load balancing.

To illustrate the trade-off introduced by load balancing, consider a small six-item example with three item groups, a center-left access point, and a fixed rectangular deck of length 700 and width 350. Table 1 gives the item dimensions, group and item priority information, and item weights. The item dimensions and most weights are unitless scaled values based on actual equipment, while item 5's weight is intentionally inflated to make the center-of-gravity (CG) effect visible in this small illustrative example. The group and item priorities determine the access-point and cohesion weights used by the prioritization objective, while the item weights determine the cargo CG.

Table 1: Illustrative six-item instance with unitless dimensions and weights.

Item	Length	Width	Weight	Group	Item Priority	Group Priority
1	258	131	74,400	2	1	1
2	312	137	38,041	1	1	2
3	210	100	11,500	1	2	2
4	202	96	12,198	1	3	2
5	341	144	700,000	3	1	3
6	136	86	2,410	3	2	3

Figure 1 shows three optimal Single MILP layouts for this same instance, each under a different constraint setting. Panel (a) is the optimal solution when the model optimizes only the prioritization objective and does not consider load balance at all; the 15% tolerance box is shown only as a reference, and the achieved cargo CG falls outside that box. This prioritization-only solution has objective value 5.93e3. Panel (b) is the optimal solution when load balance is included with a lenient 15% center-of-gravity tolerance, with objective value 6.02e3. Panel (c) is the optimal solution when load balance is included with a rigorous 1% tolerance, with objective value 6.68e3. This comparison illustrates why the balanced problem is meaningfully different from the prioritization-only problem, since adding and tightening load balance can change the optimal layout, sometimes minimally and sometimes substantially, and naturally increases the prioritization objective because the model must also satisfy a global balance requirement.

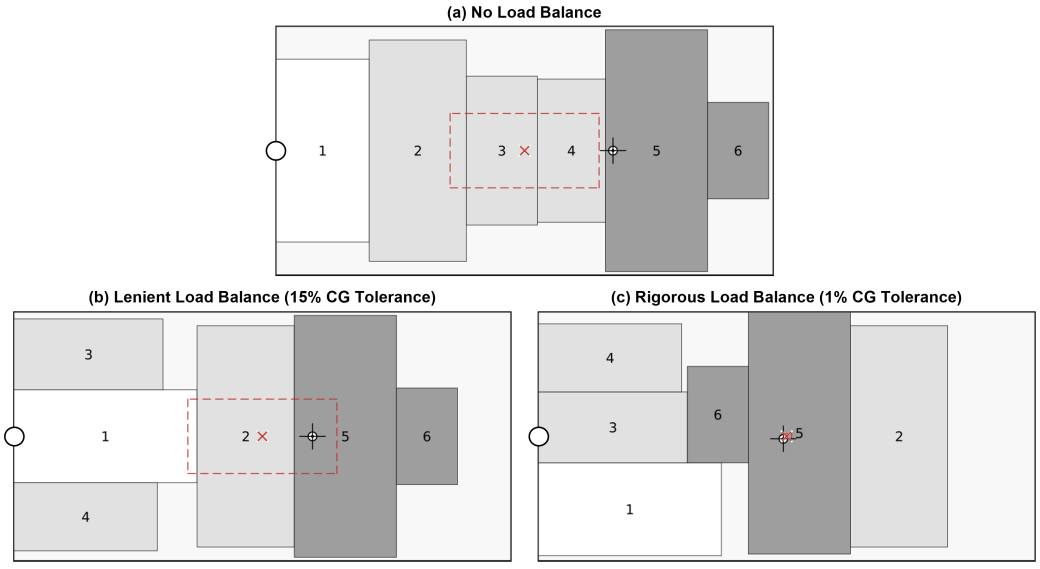

Figure 1: Optimal six-item layouts for the illustrative instance. Panel (a) shows the optimal prioritization-only layout with no load-balance constraint; panel (b) shows the optimal layout with a 15% center-of-gravity (CG) tolerance; and panel (c) shows the optimal layout with a 1% CG tolerance. Items are shaded by group and labeled by raw priority rank, where 1 is the highest-priority item. The circle on the center-left edge is the off-loading point. The dotted rectangle is the allowable target-CG region, the X is the target CG, and the bullseye marker is the achieved cargo CG.

The remainder of this paper is structured as follows. Section 2 reviews the relevant literature on

combat loading doctrine, prioritized packing, and the integration of operational constraints. Section 3 details the extended mathematical model. Section 4 describes the solution techniques, including the in-stride and post-processing matheuristic methods. Section 5 outlines the experimental design and data generation process. Section 6 presents our detailed numerical results and analysis. Finally, Section 7 concludes with a summary of findings, practical implications, and directions for future research.

2. Literature Review

In this section, we first discuss military doctrine and the planning tools that motivate this work. We then ground our approach in the context of packing with spatial preferences, drawing on our foundational model. Finally, we survey literature on incorporating the specific operational constraints that are central to this paper's contribution.

2.1. Combat Loading Doctrine and Planning Tools

Military load planning, particularly for deployments into contested environments, is governed by combat-loading doctrine, which prioritizes operational effectiveness over pure space utilization. Combat loading arranges equipment on transport assets so that it can be off-loaded rapidly, in the tactical sequence required for immediate employment, and with unit integrity maintained [5]. This differs from administrative loading, which typically focuses on maximizing cargo density. Historical amphibious operations reinforce these principles. Archival records from Inchon, Leyte, and Lingayen Gulf identify the requirement that "combat elements and their combat material go over the beaches together" [6] and document the consequences of failed sequencing, including "dangerous concentration of non-segregated cargo around the unloading point which produced troop logistic problems and potential losses" [7]. The same reports document worst-case space utilization of "thirty-five percent (35%) efficient when so loaded," equivalent to 65% utilization, which motivates our 65% to 85% experimental utilization range. With renewed attention to LSCO, these sequencing, cohesion, and utilization trade-offs remain directly relevant to modern load planning [8, 9].

The primary tool used by the U.S. DoD for load planning is the Integrated Computerized Deployment System (ICODES) [2]. ICODES is an advanced information-centric system using software agents to perform feasibility analysis on load plans against numerous constraints, including vessel trim and stability, hazardous material (HAZMAT) segregation, and accessibility [1]. While indispensable for this feasibility analysis, creating an initial, tactically-optimized plan often requires significant manual effort. Its automated stowage features are designed to find a feasible layout that respects physical constraints, but they do not optimize for the nuanced, multi-level priorities of combat loading. Consequently, planners frequently resort to manual adjustments to meet a commander's intent regarding off-load sequencing and unit cohesion [10, 11], a process that is time-consuming and offers no guarantee of optimality or near-optimality. At the same time, the underlying optimization structure is not unique to military embarkation. Closely related prioritized deck-loading and roll-on/roll-off planning settings arise whenever an assigned cargo set must be arranged within a fixed deck while respecting sequencing, accessibility, and balance requirements.

2.2. Packing with Spatial Preferences

Our work is built upon the foundation of the two-dimensional bin packing problem, a class of NP-hard problems focused on arranging items in a container [12]. Classical bin packing literature is vast, with objectives centered on spatial efficiency, such as minimizing the number of bins or maximizing packing density [13, 14, 15]. The geometric nature of the problem has led to a variety of solution methods, including mixed-integer linear programming (MILP) formulations [16, 17, 18] and a wide range of

heuristics [19, 20]. However, these classical approaches traditionally lack mechanisms to address the relative placement of items, which is critical in many logistical applications.

A separate but conceptually related field, Facility Layout Planning (FLP), addresses the optimal arrangement of departments or machines to minimize material handling costs, which are typically a function of the distances between facilities [21]. Early FLP models were often formulated as a Quadratic Assignment Problem (QAP) [22], while later work on the Unequal-Area Facility Layout Problem (UA-FLP) allowed for more flexible, non-uniform shapes [23, 24, 25]. While FLP provides the core concept of a distance-based objective, it does not typically handle the strict, non-overlapping geometric constraints required in packing problems.

In prior work, Kirschenman et al. [3] bridged this gap by introducing the prioritized two-dimensional orthogonal packing model. This model integrates a flexible, FLP-inspired weighted rectilinear distance objective into a classical packing formulation. This objective simultaneously manages item-to-access-point proximity and intra-group cohesion, allowing for the encoding of multi-level operational priorities. That work established the value of this integrated approach and, crucially, demonstrated that a sliding-window matheuristic technique significantly outperformed a Single MILP in both time and solution quality for this new problem class. This paper builds on that foundation by incorporating hard center-of-gravity feasibility constraints and studying the resulting globally coupled variant. In doing so, it shifts the emphasis from prioritized packing alone to the broader literature on operational feasibility constraints, especially load balancing in logistics and maritime settings.

2.3. Operational Constraints in Packing Literature

While our foundational model addresses prioritization, military applications demand adherence to physical and safety constraints. As noted, the three constraints investigated here—load balancing, obstacle avoidance, and material separation—are chosen because they represent primary feasibility checks within planning systems like ICODES. Within that broader space, maritime ship-loading and stowage work explicitly treats vessel safety and feasibility at the deck level, including ship-loading surveys and formulations that control floating state or stowage safety in maritime settings [26, 27, 28]. Broader logistics loading models then provide closely related balance-aware container and cargo-loading formulations, but those usually optimize standard loading objectives or route- or container-level balance rather than a prioritized single-bin arrangement objective. The literature contains various approaches to the integration of load balancing into packing problems. Some studies formulate load balancing as an objective to be minimized, typically the deviation of the cargo's center of gravity (CG) from a target point [29, 30]. Others treat it as a hard constraint, requiring the CG to lie within a predefined tolerance box [31, 32, 33, 34]. These methods can be integrated into the initial packing process [29] or applied as a post-processing step to adjust an existing layout [35, 36, 37]. However, most of this research either optimizes standard loading objectives such as utilization, bin count, or multi-container balance, or it stays within monolithic solution models. The central challenge addressed in our work—efficiently integrating global load balancing constraints within a matheuristic framework designed for a complex, non-standard prioritization objective—remains largely unexplored in the literature. We therefore position this work most directly against the hard-constraint and repair-oriented strands of that literature. Unlike objective-only balancing formulations such as Trivella and Pisinger [29], our target CG box is a feasibility requirement; unlike utilization-driven models that distribute weight across multiple bins or containers, our model optimizes a weighted single-bin access and cohesion arrangement; and unlike monolithic models, our main question is how to adapt a more efficient decomposed matheuristic so that it can enforce that requirement while preserving the prioritized layout objective.

Our model also has the flexibility to include other constraints. Obstacle avoidance can be handled by modeling obstacles as fixed, pre-positioned items, a standard technique in both packing and FLP literature [13, 24]. Material separation, for ensuring minimum distances between hazardous items, has been modeled in related layout problems using modified non-overlap constraints [38]. In this paper, these two extensions are incorporated directly into the formulation discussion and illustrated through proof-of-concept demonstrations.

3. Methodology

This section details the mathematical model for the prioritized two-dimensional packing problem, extended to address military combat loading. We begin by establishing the core model which integrates a prioritization-based objective into a packing framework. We then introduce the three critical operational constraints motivated by military planning systems like ICODES: load balancing, fixed deck obstacles, and material separation. We present the full mixed-integer linear programming (MILP) formulation that incorporates load balancing directly, then note how extensions for obstacle avoidance and material separation can be incorporated.

Load balancing introduces a global coupling constraint: the center of gravity depends on all item placements simultaneously, unlike the local geometric constraints of non-overlap, obstacle avoidance, and material separation. This global coupling requires different solution techniques, motivating our experimental focus on load balancing performance.

We consider a single rectangular bin (e.g., a vessel deck) of length L and width W, into which a set of n rectangular items $i \in I = \{1, 2, \ldots, n\}$ must be placed. Each item i has length p_i, width q_i, and mass m_i. Our objective is not classical space minimization but rather the optimization of a prioritization scheme that pulls high-priority items toward a designated access point (x^o, y^o) while also encouraging functionally related (or task organized) items to cluster together. This is encoded in a pre-computed prioritization matrix, $\boldsymbol{\pi}$, with entries π_{ik} for each pair of items $i, k \in I$ where $i \leq k$. The diagonal entries, π_{ii}, represent the priority weight for placing item i close to the access point, while the off-diagonal entries, π_{ik} for $i < k$, represent the cohesion priority between items i and k.

To formulate the model, we define the following variables. Continuous variables x_i and y_i denote the coordinates of the bottom-left corner of item i, while x_i^r and y_i^r denote the top-right corner. Binary variables a_{ik}^x and a_{ik}^y indicate the relative placement of item i with respect to item k; specifically, $a_{ik}^x = 1$ means item k is entirely to the left of item i (i.e., the right edge of k, x_k^r, lies to the left of the left edge of i, x_i), and analogously for $a_{ik}^y = 1$ in the y-dimension (item k entirely below item i). The orientation of each item is controlled by binary variables $t_{i,c,d}$, which employ a dimension-mapping approach where $c \in \{1, 2\}$ indexes bin dimensions (x, y) and $d \in \{1, 2\}$ indexes item dimensions (length, width). While four binary variables are defined per item to represent this mapping, they collectively encode exactly two orientations—0° and 90° rotation—as required for orthogonal packing, ensuring each item dimension is aligned to one of the bin dimensions. Auxiliary variables ρ_{ik}^x and ρ_{ik}^y capture the rectilinear distance components: when $i = k$, ρ_{ii}^x and ρ_{ii}^y represent the distance from item i's centroid to the access point in the x and y dimensions respectively, while for $i < k$, ρ_{ik}^x and ρ_{ik}^y represent the distance between the centroids of items i and k in each dimension. Finally, for load balancing, we introduce continuous variables B^x and B^y to represent the final center of gravity (CG) of the packed cargo. The target CG is $(B_{\texttt{opt}}^x, B_{\texttt{opt}}^y)$ with permissible deviations δ_x and δ_y, which represent vessel stability constraints. These tolerances may be widened for vessels equipped with ballast adjustment capabilities, which can compensate for moderate CG deviations through operational countermeasures. The total mass of all items is $M^E = \sum_{i \in I} m_i$. The full MILP formulation with integrated load balancing is as follows:

$$\min \quad \sum_{i\in I}\sum_{k\in I,\, i\le k} \pi_{ik}\left(\rho^x_{ik}+\rho^y_{ik}\right) \tag{1}$$

$$\text{s.t.} \quad x^r_i - x_i = t_{i,1,1}p_i + t_{i,1,2}q_i \qquad \forall i\in I \tag{2}$$

$$y^r_i - y_i = t_{i,2,1}p_i + t_{i,2,2}q_i \qquad \forall i\in I \tag{3}$$

$$a^x_{ik} + a^x_{ki} + a^y_{ik} + a^y_{ki} \ge 1 \qquad \forall i,k\in I, i<k \tag{4}$$

$$\sum_{c\in\{1,2\}} t_{i,c,d} = 1 \qquad \forall i\in I, d\in\{1,2\} \tag{5}$$

$$\sum_{d\in\{1,2\}} t_{i,c,d} = 1 \qquad \forall i\in I, c\in\{1,2\} \tag{6}$$

$$x^r_k \le x_i + (1-a^x_{ik})L \qquad \forall i,k\in I, i\ne k \tag{7}$$

$$x^r_i \le x_k + (1-a^x_{ki})L \qquad \forall i,k\in I, i\ne k \tag{8}$$

$$y^r_k \le y_i + (1-a^y_{ik})W \qquad \forall i,k\in I, i\ne k \tag{9}$$

$$y^r_i \le y_k + (1-a^y_{ki})W \qquad \forall i,k\in I, i\ne k \tag{10}$$

$$\frac{x_i+x^r_i}{2} - \frac{x_k+x^r_k}{2} \le \rho^x_{ik} \qquad \forall i,k\in I,\ i<k \tag{11}$$

$$\frac{x_k+x^r_k}{2} - \frac{x_i+x^r_i}{2} \le \rho^x_{ik} \qquad \forall i,k\in I,\ i<k \tag{12}$$

$$\frac{y_i+y^r_i}{2} - \frac{y_k+y^r_k}{2} \le \rho^y_{ik} \qquad \forall i,k\in I,\ i<k \tag{13}$$

$$\frac{y_k+y^r_k}{2} - \frac{y_i+y^r_i}{2} \le \rho^y_{ik} \qquad \forall i,k\in I,\ i<k \tag{14}$$

$$\frac{x_i+x^r_i}{2} - x^o \le \rho^x_{ii} \qquad \forall i\in I \tag{15}$$

$$x^o - \frac{x_i+x^r_i}{2} \le \rho^x_{ii} \qquad \forall i\in I \tag{16}$$

$$\frac{y_i+y^r_i}{2} - y^o \le \rho^y_{ii} \qquad \forall i\in I \tag{17}$$

$$y^o - \frac{y_i+y^r_i}{2} \le \rho^y_{ii} \qquad \forall i\in I \tag{18}$$

$$\sum_{i\in I} m_i \cdot \frac{x_i+x^r_i}{2} = M^E \cdot B^x \tag{19}$$

$$\sum_{i\in I} m_i \cdot \frac{y_i+y^r_i}{2} = M^E \cdot B^y \tag{20}$$

$$B^x_{\texttt{opt}} - \delta_x \le B^x \le B^x_{\texttt{opt}} + \delta_x \tag{21}$$

$$B^y_{\texttt{opt}} - \delta_y \le B^y \le B^y_{\texttt{opt}} + \delta_y \tag{22}$$

$$0 \le B^x \le L,\ 0 \le B^y \le W \tag{23}$$

$$0 \le x_i \le x^r_i \le L, \quad 0 \le y_i \le y^r_i \le W \qquad \forall i\in I \tag{24}$$

$$\rho^x_{ik}, \rho^y_{ik} \ge 0 \qquad \forall i,k\in I,\ i\le k \tag{25}$$

$$a^x_{ik}, a^y_{ik} \in \{0,1\} \qquad \forall i,k\in I,\ i\ne k \tag{26}$$

$$t_{i,c,d} \in \{0,1\} \qquad \forall i\in I,\ c,d\in\{1,2\} \tag{27}$$

The objective function (1) minimizes the sum of weighted rectilinear distances, promoting the desired prioritized layout. Constraints (2) and (3) set the placed dimensions of each item based on its chosen orientation. Constraints (5) and (6) manage item rotation, while constraint (4) together with constraints (7)–(10) enforce pairwise geometric separation. The midpoint expressions $(x_i + x^r_i)/2$ and $(y_i + y^r_i)/2$ represent the induced centroid coordinates of each placed item, so both the distance terms

and the balance equations are built from the same placement variables.

Constraints (11) and (12) linearize the absolute x-distance between the centroids of items i and k, while constraints (13) and (14) do the same for the absolute y-distance between those two item centroids. Constraints (15) and (16) analogously linearize the centroid-to-access-point distance in the x-dimension, and constraints (17) and (18) do so in the y-dimension. Reading constraints (11)–(18) one pair at a time therefore shows that each pair of inequalities captures the positive and negative sides of one absolute-value distance term.

The balance block can be read in three steps. First, those midpoint expressions identify the longitudinal and transverse coordinates of each item's centroid. Second, constraints (19) and (20) sum the corresponding mass moments over all items and define the shared cargo center-of-gravity variables B^x and B^y. Third, constraints (21) and (22) require that cargo centroid to lie within the prescribed longitudinal and transverse tolerance box around $(B^x_{\texttt{opt}}, B^y_{\texttt{opt}})$. This first-moment treatment is connected to load-balanced packing formulations such as Trivella and Pisinger [29], but is used here as a hard feasibility condition within the prioritization model. The left/right and front/rear restrictions are therefore global moment conditions induced by all item positions and masses, not local pairwise placement rules.

From a formulation-size perspective, the dominant burden is still the base prioritized-packing model's $O(n^2)$ separation and distance structure. Relative to the unbalanced formulation, the load-balancing extension adds only two shared continuous variables, B^x and B^y, plus the four displayed balance rows in constraints (19)–(22); equivalently, those rows represent six scalar linear equalities/inequalities if the two double-sided box constraints are counted one-sided. It materially changes tractability not by dramatically expanding raw model size, but by destroying locality, since every item's position contributes to the same global balance equations.

3.1. Modeling Additional Operational Constraints

While load balancing is the only globally coupling constraint evaluated in the full computational study, the formulation can also accommodate fixed obstacles and material separation requirements that arise in military load planning systems such as ICODES.

Vessel decks may contain fixed obstacles such as structural pillars. We model these by partitioning the item set into cargo items to be placed, I^E, and fixed obstacle items, I^F. For each obstacle $f \in I^F$, its coordinates are known parameters $(x^*_f, y^*_f, x^{r*}_f, y^{r*}_f)$, and we impose

$$x_f = x^*_f, \quad y_f = y^*_f, \quad x^r_f = x^{r*}_f, \quad y^r_f = y^{r*}_f \qquad \forall f \in I^F \tag{28}$$

The non-overlap constraints (4)–(10) then apply to all pairs in $I = I^E \cup I^F$, so cargo items cannot occupy the fixed obstacle regions. Because obstacle coordinates are known parameters, the associated orientation variables are implied by the obstacle geometry and can be preset or handled during solver preprocessing. This is a standard modeling device in the packing and FLP literature [13, 24].

To enforce material separation, let $H \subseteq I^E \times I^E$ denote the set of item pairs requiring a minimum edge-to-edge separation of d^H, and let h_{ik} be a binary parameter equal to 1 if $(i,k) \in H$ and 0 otherwise. We modify the geometric non-overlap constraints for those pairs as

$$x^r_k \le x_i - h_{ik} \cdot d^H + (1 - a^x_{ik})L \qquad \forall i,k \in I^E, i < k \tag{29}$$

with analogous additions to constraints (8), (9), and (10). If separation is required ($h_{ik} = 1$), an additional gap of d^H is enforced between the item edges. This embeds separation rules directly into the existing non-overlap logic without changing the paper's core prioritization structure, similar to

safety-distance treatments used in related layout models [38].

Both constraint types can be activated independently or simultaneously, and the same solution architecture remains applicable. The corresponding proof-of-concept demonstrations are reported later in the paper.

4. Solution Techniques

This section presents the solution methods developed to address the prioritized two-dimensional packing problem with load balancing constraints. Figure 2 provides a high-level roadmap before the technical subsections. Before turning to the individual methods, it is useful to distinguish the present problem class from classical two-dimensional packing families such as minimum bin count, minimum strip height, maximum packed value, or pure packing feasibility [39]. Here the bin and assigned item set are already fixed upstream, and the remaining decision is the internal single-bin arrangement under weighted access-point proximity, within-group cohesion, and hard center-of-gravity feasibility. This creates a hybrid problem that retains the geometric feasibility burden of classical packing, but uses a weighted distance-based objective more typical of facility layout planning [24]. The big-M non-overlap constraints and $O(n^2)$ binary separation variables therefore still create weak relaxations and rapidly growing combinatorial search spaces, while decomposition approaches such as column generation are a poor fit because all items are already assigned to one bin, leaving no natural assignment master problem. As a result, utilization-driven packing rules and standard multi-bin decomposition ideas do not transfer directly, since a feasible local placement can still perform poorly on the prioritization objective or conflict with the final global balance requirement.

Our solution framework consists of two stages. In Stage 1, we employ one of three alternative approaches: (i) the Single MILP solves the complete formulation with center-of-gravity (CG) constraints directly; (ii) the sliding-window (SW) Iterative method optimizes only the prioritization objective without incorporating load balancing into the subproblems; and (iii) the SW In-Stride method augments SW subproblems with load balancing penalties to proactively guide the solution toward balance. The grid-based heuristic (GBH), shown with dashed arrows in Figure 2, guides penalty scaling for the in-stride approach. After Stage 1, a feasibility test verifies whether the resulting layout is both feasibly packed and load balanced. If not, Stage 2 applies a post-processing adjustment strategy that employs short, iterative MILP adjustments to selectively unfix items (using priority-based, x^r-based, mass-weighted x^r, or greedy CG-impact sequences) to achieve balance with minimal disruption to the prioritized layout, continuing until limits are reached. This two-stage architecture is the algorithmic package studied here; we do not append a third-stage generic metaheuristic improver because that would define a materially different method family from the one evaluated in this paper. The Single MILP approach, when successful, directly yields a balanced solution without requiring post-processing. If the Stage 1 Single MILP terminates with an infeasibility result, the target box is exactly certified infeasible. By contrast, if a sliding-window pipeline exhausts its allotted budget without producing a load-balanced layout, we record only that the procedure did not find one within the allotted search effort. We now describe each component in detail. We first describe the baseline Single MILP approach in Section 4.1, which serves as a computational benchmark. Section 4.2 introduces the grid-based heuristic, which provides the reference score used to scale the in-stride penalties. Section 4.3 reviews the sliding-window matheuristic framework from prior work [3], which forms the basis for our enhanced methods. Section 4.4 details our in-stride balancing strategies that proactively integrate load balancing penalties into the matheuristic subproblems. Finally, Section 4.5 presents the post-processing adjustment techniques that reactively correct unbalanced solutions through iterative item relaxation.

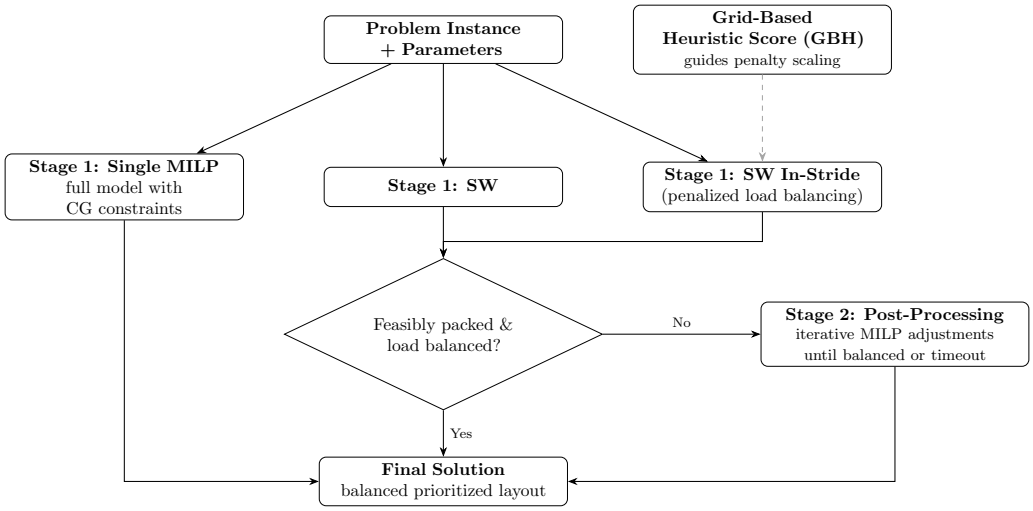

Figure 2: Solution techniques roadmap illustrating the two-stage framework. Stage 1 employs one of three approaches (Single MILP, SW Iterative, or SW In-Stride), with guidance from the grid-based heuristic (GBH; dashed arrows). Stage 2 applies post-processing adjustments when Stage 1 solutions are not load balanced.

4.1. Single MILP Approach

The most conceptually straightforward approach to solving the problem is to formulate and solve the complete MILP presented in Section 3 using a commercial solver. This method, which we refer to as the Single MILP approach, solves equations (1)–(22) directly for all n items simultaneously.

The primary advantage of this approach is its theoretical exactness—given sufficient computational resources, it will find the globally optimal solution that minimizes the prioritization objective while satisfying all constraints, including load balancing. Moreover, modern solvers provide optimality gaps throughout the solution process, offering valid bounds on solution quality even when terminated early. Since the load balancing constraints are integrated directly into the model, any feasible solution found by this approach is guaranteed to satisfy the center of gravity requirements.

This differs from the matheuristic feasibility-search procedures described in Section 4.3, and this distinction matters for the target-box infeasibility study introduced in Section 5.3 and reported in Section 6.3.2. Single MILP can certify infeasibility for the specified target box because it solves the full balance-constrained model directly. The matheuristics, by contrast, are feasibility-search procedures, so time-budget exhaustion is not an infeasibility proof. The attainable-CG screen, a simple item-centroid bound on possible cargo CG locations, is therefore only a precheck before exact certification or continued search on nontrivial cases.

However, the computational burden of this approach grows rapidly with problem size. Prior tests without load balancing already showed sharp growth from 8- to 9-item subproblems [3], and the present load balance constraints add global first-moment coupling that further exacerbates this computational challenge. This intractability motivates matheuristic approaches that solve smaller restricted subproblems and produce high-quality solutions more efficiently.

4.2. Grid-Based Heuristic

Before presenting our matheuristic strategies, we introduce the grid-based heuristic (GBH) to provide a scale-appropriate reference score for penalty scaling in the in-stride approaches. We present its construction here because understanding its mechanics is essential for understanding how the in-stride penalties are normalized across subproblems.

The GBH exploits a fundamental geometric property: Euclidean distances never exceed their rectilinear counterparts ($\|u - v\|_2 \leq \|u - v\|_1$). It constructs a Euclidean-distance surrogate by

decomposing items into unit squares that pack radially around the access point in order of priority, ignoring standard packing constraints but respecting the prioritization structure.

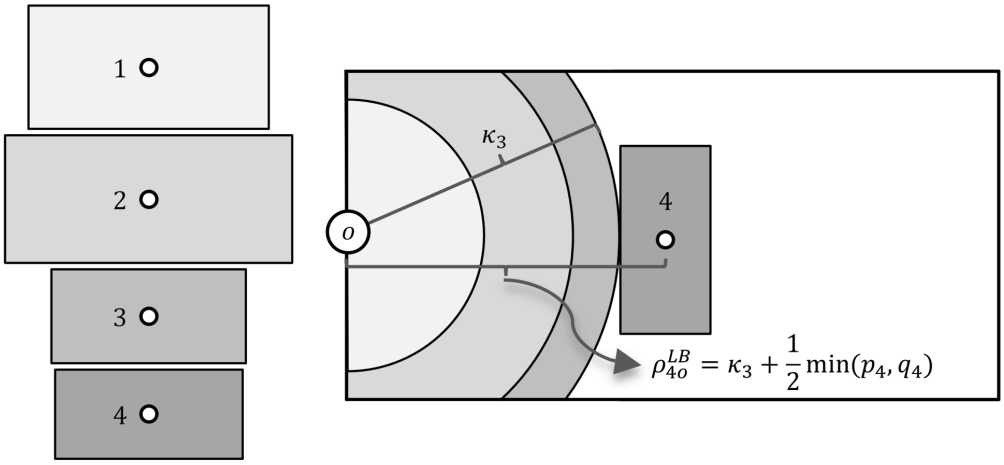

Figure 3: Grid-Based Heuristic (GBH) construction showing radial packing of unit squares and exclusion radius calculation.

The construction proceeds iteratively for items sorted by descending access-point priority ($\pi_{11} > \pi_{22} > \cdots > \pi_{nn}$). For each item i, we determine the minimum possible Euclidean distance from its centroid to the access point, given the space occupied by higher-priority items. Let κ_{i-1} denote the exclusion radius after placing items 1 through $i-1$. The corresponding score distance for item i to the access point is

$$\rho_{ii}^{GBH} = \kappa_{i-1} + \frac{1}{2}\min\{p_i, q_i\}. \tag{30}$$

For item-to-item distances, we use the corresponding score based on minimum dimensions,

$$\rho_{ik}^{GBH} = \frac{1}{2}\min\{p_i, q_i\} + \frac{1}{2}\min\{p_k, q_k\}, \tag{31}$$

so the total GBH is then

$$\text{GBH} = \sum_{i=1}^{n} \pi_{ii}\rho_{ii}^{GBH} + \sum_{i<k} \pi_{ik}\rho_{ik}^{GBH}. \tag{32}$$

This reference score provides a scale-appropriate benchmark for the prioritization objective within each subproblem of our matheuristic methods. Crucially, as we will show in Section 4.4, the GBH enables principled scaling of penalty terms relative to the primary objective during in-stride approaches. At each iteration, we conduct progressively larger subproblem solves without complete knowledge of the full problem scale. By using the GBH to scale the penalty coefficient, we can appropriately balance the interaction between the original objective function and the penalization term—scaling the penalty sufficiently to influence load distribution while avoiding severe detriment to solution quality.

4.3. Sliding-Window Matheuristic

The sliding-window matheuristic, introduced in Kirschenman et al. [3], provides an effective framework for solving large prioritized packing instances by breaking them into manageable subproblems. This approach forms the foundation for two of our three solution methods: the standard sliding-window matheuristic and the sliding-window matheuristic with in-stride balancing. For any solution which is not load balanced, a universal post-processing stage is applied to achieve load balance. We review its mechanics here, as understanding this baseline matheuristic is essential for comprehending the enhancements that follow.

The method begins by creating a global priority ordering of all items, considering both group-level

and item-level priorities to produce a single sorted list in descending order of importance. Given a window size parameter w, the algorithm iteratively solves subproblems containing exactly w items. In the first iteration, we optimize the placement of the w highest-priority items. Upon completion, we fix the position of the single highest-priority item from that window and slide forward: removing this fixed item from consideration and introducing the next item from the global priority list, maintaining a constant subproblem size of w items.

This process continues until all items have been fixed in position. At each iteration t, the subproblem contains items from positions t through $t+w-1$ in the global priority list, with all items from positions 1 through $t-1$ having their coordinates fixed from previous iterations. The mathematical formulation for each subproblem mirrors the full model from Section 3, but with fixed coordinate values substituted for previously placed items.

The key advantage of this matheuristic is computational tractability. By limiting subproblem size to w items, we can apply reasonable time limits (e.g., 5–60 seconds) and still obtain high-quality solutions for each iteration. The approach also provides natural flexibility: smaller windows (e.g., $w = 5$) solve quickly but may miss beneficial arrangements among items, while larger windows (e.g., $w = 9$) capture more complex interactions at the cost of increased computation time per iteration.

However, the fundamental challenge for load balancing is that this matheuristic makes local decisions without full knowledge of the global center of gravity. Each subproblem can only consider the mass and placement of its w free items plus any previously fixed items, making it difficult to ensure the final configuration satisfies the global load balancing constraints (21)–(22). This limitation motivates our two enhanced strategies: proactively guiding placement through in-stride penalties and reactively correcting imbalances through post-processing adjustments.

4.4. Sliding-Window In-Stride Balancing Strategy

To address the challenge of incorporating global load balancing constraints within local matheuristic subproblems, we develop an in-stride balancing strategy that augments each sliding-window iteration with penalty terms that guide the solution toward a balanced configuration. This proactive approach modifies the objective function of each subproblem to include both the original prioritization terms and additional penalties for deviating from the target center of gravity.

4.4.1. Penalized Objective Formulation

For each sliding-window subproblem at iteration t, we introduce slack variables to measure center of gravity deviations and augment the objective with penalty terms. Let I_t^{free} denote the set of free items in iteration t and I_t^{fixed} the set of previously fixed items. We define slack variables,

$$s_x^+ \geq B_x - B_{\text{opt}}^x - \delta_x, \tag{33}$$

$$s_x^- \geq B_{\text{opt}}^x - B_x - \delta_x, \tag{34}$$

$$s_y^+ \geq B_y - B_{\text{opt}}^y - \delta_y, \text{ and} \tag{35}$$

$$s_y^- \geq B_{\text{opt}}^y - B_y - \delta_y, \tag{36}$$

where B_x and B_y are the center of gravity coordinates considering both free and fixed items in the current iteration. The modified objective becomes

$$\min \sum_{i \in I_t^{\text{free}}} \sum_{k \in I_t^{\text{free}},\, i \leq k} \pi_{ik}(\rho_{ik}^x + \rho_{ik}^y) + \lambda_t \cdot (s_x^+ + s_x^- + s_y^+ + s_y^-). \tag{37}$$

The parameter λ_t controls the trade-off between prioritization and load balancing in iteration t. Setting $\lambda_t = 0$ eliminates the balancing penalty, recovering the original sliding-window matheuristic from

Kirschenman et al. [3] that optimizes only the prioritization objective, while larger values increasingly emphasize center of gravity alignment.

4.4.2. Lambda Scaling with GBH

A critical challenge in multi-objective optimization is ensuring that penalty terms are appropriately scaled relative to the primary objective. We address this by using the grid-based heuristic to normalize the load balancing penalties. For iteration t, we compute the GBH for the current subproblem's free items and scale lambda as

$$\lambda_t = \lambda_{\text{factor}} \cdot \frac{\text{GBH}_t}{2(\delta_x + \delta_y)}, \tag{38}$$

where λ_{factor} is a user-specified parameter controlling the relative importance of load balancing, and the denominator normalizes by the total allowable deviation. This scaling ensures that the penalty term magnitude is proportional to the prioritization objective scale, providing consistent behavior across different problem instances and iterations.

4.4.3. Constant Lambda Schedule

The simplest in-stride approach applies a constant penalty factor throughout all iterations, setting λ_{factor} to a fixed value. This strategy maintains consistent pressure toward load balancing throughout the matheuristic process. In our experiments, we test values of $\lambda_{\text{factor}} \in \{0.05, 0.25\}$ to explore the sensitivity of solution quality to this parameter.

4.4.4. Dynamic Lambda Schedule

Dynamic lambda scheduling exploits the matheuristic's priority-ordered item placement. Early iterations process high-priority items with low balancing penalties, preserving access-point proximity. Later iterations process low-priority items with escalating penalties, placing the burden of achieving load balance on items with greater placement flexibility. We employ an exponential penalty function formulation similar to approaches used in multilevel optimization by DorMohammadi and Rais-Rohani [40], gradually increasing the load balancing emphasis with

$$\lambda_{\text{factor}}(t) = \lambda_{\max} \cdot \frac{e^{a(t/(N-b))}}{e^{a(1-b)+c}}, \tag{39}$$

where t is the current iteration, N is the total number of iterations, $\lambda_{\max}$ is the maximum penalty factor, and a, b, and c are shape parameters. We set $b = 0.75$ to shift emphasis toward later iterations and $c = 0.01$ to ensure near-zero initial penalties. The parameter $a = \gamma \times 5$ controls the steepness of the exponential growth, with test values of $\gamma \in \{1.0, 2.5, 5.0, 10.0\}$ and $\lambda_{\max} \in \{0.25, 0.5\}$.

4.4.5. Standard vs. In-Out Penalty Variants

We investigate two variants of the in-stride penalty structure. The standard variant penalizes only deviations outside the allowable tolerance box, as shown in equation (37). The in-out variant adds an additional term that continuously pulls the center of gravity toward the optimal position, even when already within tolerance. To maintain the linear programming structure, we represent the absolute value terms using auxiliary variables and constraints with

$$\min \sum_{i,k} \pi_{ik}(\rho^x_{ik} + \rho^y_{ik}) + \lambda^{\text{out}}_t(s^+_x + s^-_x + s^+_y + s^-_y) + \lambda^{\text{in}}_t(u_x + u_y), \tag{40}$$

where u_x and u_y are auxiliary variables representing the absolute deviations, constrained by $u_x \geq B_x - B^x_{\text{opt}}$, $u_x \geq B^x_{\text{opt}} - B_x$, $u_y \geq B_y - B^y_{\text{opt}}$, and $u_y \geq B^y_{\text{opt}} - B_y$.

The parameter λ^{out}_t follows the scaling from equation (38) and $\lambda^{\text{in}}_t = \lambda_{\text{inside}} \cdot \lambda^{\text{out}}_t$ with $\lambda_{\text{inside}} \in \{0.05, 0.1, 0.2\}$. This variant provides continuous guidance toward the optimal center of gravity by

penalizing deviations from the target position even when solutions already satisfy load balancing constraints, potentially improving convergence to balanced solutions.

4.5. Post-Processing Strategy

For sliding-window solutions that fail to achieve load balance in Stage 1, we employ an iterative balance adjustment procedure that selectively unfixes and re-optimizes item positions.

4.5.1. Iterative Balance Adjustment Framework

The post-processing strategy operates through iterative relaxation of fixed items. Given an unbalanced solution from Stage 1, we apply the following iterative balance adjustment procedure. Let I denote the set of all items, $n = |I|$ the total number of items, and $(x_i^{(1)}, y_i^{(1)})$ the Stage 1 coordinates for item i. The algorithm maintains a growing set of 'free' items whose positions can be adjusted, while other items remain fixed at their Stage 1 positions, with k denoting the current number of free items in the restricted repair problem; the procedure begins at k_{initial} and increases k by one after each unsuccessful restricted solve.

Algorithm 1 Iterative Balance Adjustment

Input: Stage 1 coordinates $\{(x_i^{(1)}, y_i^{(1)})\}_{i \in I}$, item dimensions and masses $\{(p_i, q_i, m_i)\}_{i \in I}$, CG tolerance box $[\delta_x, \delta_y]$

Output: Load-balanced coordinates $\{(x_i^*, y_i^*)\}_{i \in I}$ or no balanced repair found within the current search limits

Generate unfixing sequence (ordered list of item indices) using techniques from Section 4.5.2
Initialize $k \leftarrow k_{\text{initial}}$ ▷ Initial number of free items
while $k \leq n$ **do** ▷ n is total number of items
    Select first k items from unfixing sequence as free items
    Solve restricted MILP: minimize prioritization objective over free item coordinates, with non-free items fixed at $(x_i^{(1)}, y_i^{(1)})$
    if solution achieves load balance (CG within tolerance) **then**
        return balanced solution
    else
        $k \leftarrow k + 1$ ▷ Add next item from unfixing sequence
    end if
end while
return no balanced repair found ▷ All items exhausted without achieving balance under the current search limits

Figure 4 makes the repair logic concrete using the same six-item data from Table 1. Panel (a) is the ending Stage 1 layout, which is the same prioritization-only layout shown in Figure 1(a), with objective value 5.93e3. It is feasible for packing, but its achieved cargo CG lies outside the 10% tolerance box used in this repair example. Panel (b) then identifies the released subset selected by the Reverse Priority unfixing sequence, where the three thick-outlined items, items 6, 5, and 4, are unfixed and allowed to move in the restricted repair problem while items 1–3 remain fixed. Panel (c) shows the resulting configuration after restricted reoptimization of only that released subset; the achieved cargo CG is now inside the allowable target-CG region, and the repaired layout has objective value 6.37e3. In each panel, the dotted rectangle is the allowable target-CG region. On this small instance, the direct Single MILP and post-processing repair are naturally comparable because the full model is still easy to solve. For example, at the same 10% tolerance, this repair matches the direct Single MILP objective of 6.37e3; at 1%, 5%, and 15%, the corresponding Reverse Priority repairs require 4, 4, and 3 unfixed items and produce objectives 7.23e3, 6.82e3, and 6.37e3, compared with direct Single MILP objectives 6.68e3, 6.51e3, and 6.02e3. Thus, this toy example should not be read as evidence that

either method yields the better objective value in every setting. Instead, it illustrates how a small, targeted repair can recover load balance without restarting the full model and while preserving most of the prioritization-only layout. The computational results later show why this mechanism matters at realistic scale, where the Single MILP often fails to find a feasible load-balanced solution or struggles to improve the prioritization objective after finding one.

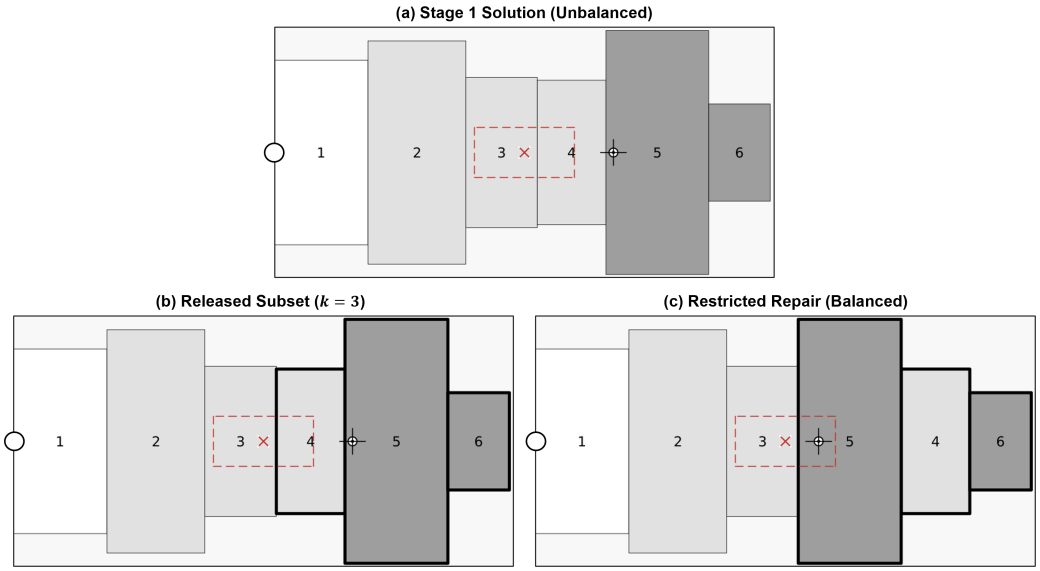

Figure 4: Illustrative Stage 2 post-processing repair on the same six-item example used in Figure 1, using a 10% center-of-gravity tolerance. Panel (a) shows the ending Stage 1 layout. Panel (b) identifies the released subset, where the three thick-outlined items are unfixed and allowed to move during restricted repair while the remaining items stay fixed. Panel (c) shows the resulting load-balanced configuration after restricted reoptimization of only that released subset. Items are shaded by group and labeled by raw priority rank, where 1 is the highest-priority item. The dotted rectangle is the allowable target-CG region, the X is the target CG, and the bullseye marker is the achieved cargo CG.

The restricted MILP in each iteration includes the full prioritization objective and all constraints from Section 3, but with coordinate variables fixed for non-free items.

In our experiments, we initialize k at 6 because a 6-item problem can solve to optimality in fractions of a second with a commercial solver, providing low computational cost while hoping to find load balanced solutions quickly. This iterative balance adjustment performs a k-way rearrangement of items to minimally achieve load balance, starting with a small subset that can be solved efficiently.

4.5.2. Unfixing Sequence Generation

The effectiveness of the post-processing strategy depends critically on the order in which items are considered for relaxation. We investigate four unfixing sequence generation techniques: Reverse Priority Sequence, Decreasing x^r Sequence, Mass-Weighted x^r Sequence, and Greedy CG Impact Sequence.

Reverse Priority Sequence The simplest approach unfixes items in ascending order of their access-point priority weight (π_{ii}), starting with the lowest-priority items. This strategy assumes that high-priority items near the access point should remain fixed to preserve the quality of the Stage 1 solution, while lower-priority items have more flexibility for repositioning.

Decreasing x^r Sequence This sequence orders items by decreasing distance from the access point measured along the relevant access direction. For the center-left access geometry used in our experiments, each item's right edge x_i^r is its far edge relative to the access point, so the DXR sequence starts with the item having the largest x_i^r value and then proceeds in descending x_i^r order. This ordering uses x_i^r as a proxy for distance from access, which is why we retain the shorthand DXR in the results section. For other access geometries, the same idea would be applied along the appropriate direction (for example, decreasing y_i^r for a center-bottom access point). The intuition is unchanged because items farther from

the access point generally incur less degradation to the prioritization objective when moved during repair.

Mass-Weighted x^r Sequence This technique combines spatial position with item mass to identify items that can significantly impact the center of gravity. For each item i, we first determine if it can move in the required direction to improve load balance (e.g., if CG is too far right, can the item move left without obstruction). For moveable items, we compute a score,

$$\text{score}_i = \alpha \cdot \frac{x_i^r}{\max_j x_j^r} + (1 - \alpha) \cdot \frac{m_i}{\max_j m_j} \cdot \text{moveable}_i, \tag{41}$$

where $\alpha = 0.5$ balances spatial and mass considerations, and moveable_i is a binary indicator that equals 1 if item i can move in the beneficial direction and 0 otherwise. Items are then unfixed in descending order of score.

Greedy CG Impact Sequence The most sophisticated approach explicitly considers each item's potential impact on correcting the center of gravity imbalance. While the first three sequence generation techniques have $O(n)$ time complexity, this greedy approach requires $O(n^2)$ operations due to the iterative evaluation of all remaining items at each step. However, in practice, the computational overhead is minimal—both the simpler methods and this greedy approach complete in a few milliseconds for typical problem sizes, making the time complexity difference negligible for practical applications. Let I denote the item set, $M^E = \sum_{i \in I} m_i$ the total loaded mass, and π_{ik} the prioritization penalty weights from the prioritization matrix. The detailed procedure is shown in Algorithm 2.

Algorithm 2 Greedy CG Impact Sequence Generation

Input: Stage 1 coordinates $\{(x_i^{(1)}, y_i^{(1)})\}_{i \in I}$, item masses $\{m_i\}_{i \in I}$, current CG (B_x, B_y), target CG $(B_{\text{opt}}^x, B_{\text{opt}}^y)$
Output: Unfixing sequence (ordered list of item indices)

```
Identify primary correction direction based on largest CG deviation
Initialize empty sequence, set of unsequenced items U = I
while U ≠ ∅ do
    for each item i ∈ U do
        Calculate maximum moveable distance d_i for item i in primary correction direction, considering bin boundaries and other items
        Impact_i = (m_i / M^E) · d_i                          ▷ CG correction potential
        Cost_i = π_ii · d_i + Σ_{k≠i} π_ik · d_i              ▷ Prioritization disruption
        Score_i = Impact_i / (Cost_i + ε)                     ▷ ε > 0 prevents division by zero
    end for
    Select i* = arg max_{i∈U} Score_i
    Append i* to sequence
    Update CG assuming i* moves full distance d_{i*}
    Remove i* from U
end while
```

4.5.3. Integration with Stage 1 Methods

The post-processing strategy serves as a universal second stage that can be applied to any Stage 1 solution that fails to achieve load balance. In our experimental framework, the integration varies by method type. For the Single MILP approach, post-processing is not applicable since any feasible solution inherently satisfies load balancing constraints. For the standard sliding-window matheuristic, post-processing is applied when the final configuration violates center of gravity constraints. For SW In-Stride variants, post-processing is applied when the penalized approach fails to achieve balance

despite the guiding penalties incorporated throughout the matheuristic process.

This flexibility allows us to evaluate whether proactive (in-stride and Single MILP) or reactive (post-processing) strategies prove more effective for integrating global constraints into matheuristic frameworks, forming a central component of our computational study.

5. Data Sets and Experimental Design

This section presents our approach to generating realistic test instances and the experimental framework for evaluating the proposed solution techniques. We describe our data generation strategy using military equipment databases and doctrine in Section 5.1, the prioritization matrix construction for our experiments in Section 5.2, and the experimental setup and evaluation metrics in Section 5.3.

5.1. Data Generation Strategy

To create test instances that reflect realistic military combat loading scenarios, we developed a systematic data generation approach using authoritative U.S. Army equipment databases and organizational structures. This strategy ensures reproducibility while capturing the complexity of actual military deployments.

5.1.1. Equipment Data Sources and Filtering

Our equipment data derives from two primary sources: the Joint Equipment Characteristic Database (JECD) for detailed physical specifications (dimensions, weight) and the Modified Table of Organizational Equipment (MTOE) for a representative U.S. Army Armored Brigade Combat Team (ABCT) from the 1st Armored Division [41, 42]. The JECD provides comprehensive technical data for military equipment, while the MTOE defines the types and quantities of equipment authorized for specific unit structures. We filtered the JECD data to include only self-propelled and towable vehicles suitable for roll-on/roll-off combat loading. Complete filtering criteria are provided in Kirschenman et al. [43].

5.1.2. Group Formation and Priority Assignment

Items are organized into functional groups based on their Unit Identification Code (UIC) and Paragraph Number (PARNO) from the MTOE structure. Our ABCT dataset contains approximately two armored brigades worth of equipment.

We assign group-level priorities sequentially (1 through n) based on their order of appearance in the MTOE, which plausibly reflects operational priority as these documents are structured to list units in tactically meaningful sequences. This approach avoids the subjectivity inherent in any deviation from the established MTOE ordering, as alternative prioritization schemes would require mission-specific judgments that vary across operations. Importantly, the purpose of our experimentation is to demonstrate the effectiveness of the prioritized packing framework itself, not to advocate for any particular group prioritization scheme. The sequential MTOE ordering provides a reasonable and consistent baseline that maintains the doctrinal relationships between units [42].

Within each group, items receive randomized priorities (1 through m, where m is the group size) to simulate the variety of possible loading sequences that might be specified by tactical planners. Prioritization matrix construction is detailed in Section 5.2.

5.1.3. Vessel Specifications

We model six representative vessel types commonly used in amphibious and logistics operations, as detailed in Table 2. Vessel dimensions come from official U.S. Navy [44, 45] and U.S. Army [46] specifications. Detailed dimension derivations are in Kirschenman et al. [43]. The dimensions represent the primary cargo areas available for vehicle loading.

While not all vessel types are primarily designed for amphibious assault operations, these represent the range of transport assets currently available in the U.S. inventory that could potentially be employed for combat loading in large-scale combat operations. In contested environments where traditional amphibious platforms may be insufficient or unavailable, vessels could be outfitted and employed to support these requirements. Our vessel selection reflects this operational reality rather than advocating for specific tactical employment, focusing instead on demonstrating the optimization framework's applicability across diverse platforms.

Table 2: Vessel specifications used in instance generation. Dimensions obtained from official U.S. Navy and Army sources [44, 45, 46].

Class	**Hull Classification**	**Length (in)**	**Width (in)**
Whidbey Island	Dock Landing Ship (LSD)	5280	600
Wasp	Landing Helicopter Dock (LHD)	3192	600
Harpers Ferry	Dock Landing Ship (LSD)	2640	600
General Frank S. Besson	Logistics Support Vessel (LSV)	2291	660
America	Landing Helicopter Assault (LHA)	1920	600
Runnymede	Landing Craft Utility (LCU)	1800	420

For all vessels, we set the target center of gravity at the geometric center ($B^x_{\text{opt}} = L/2, B^y_{\text{opt}} = W/2$), a standard assumption in the absence of vessel-specific stability data. While the CG deviation tolerances (δ_x, δ_y) enter the formulation as absolute length quantities in constraints (21) and (22), we parameterize them as a fixed percentage τ of the corresponding vessel dimensions, $\delta_x = \tau L$ and $\delta_y = \tau W$. This percentage-based parameterization scales tolerances appropriately with vessel size.

5.1.4. Instance Generation Process

Using approximately two armored brigades worth of equipment from our filtered dataset, we generated 70 problem instances through a systematic assignment process. The instance generation algorithm cycles through vessel types and space utilization targets to create diverse problem instances.

The algorithm operates by creating vessel-target pairs from five utilization levels (65%, 70%, 75%, 80%, and 85%) across all available vessel types, then cycling through these pairs to assign equipment groups systematically.

This systematic cycling ensures relatively even distribution across utilization levels (14 instances per level) while creating natural variation in group composition. When equipment exceeds target utilization, groups are split across multiple instances, reflecting realistic scenarios where tactical units may be divided across vessels due to capacity constraints. Complete algorithm details are in Kirschenman et al. [43].

Table 3 summarizes the characteristics of the generated instances, showing the diversity achieved through this process.

Table 3: Problem instance characteristics by vessel class

Vessel Class	**Groups per Instance**		**Items per Instance**	
	Min-Max	Median	Min-Max	Median
Whidbey Island	5–26	15	63–111	78.5
Wasp	4–23	9.5	26–69	44.5
Harpers Ferry	3–18	6	19–70	50
General Frank S. Besson	2–13	7	17–51	39
America	1–11	4.5	14–45	26.5
Runnymede	1–6	3.5	10–23	18

5.2. Prioritization Matrix Construction

The prioritization matrix $\boldsymbol{\pi}$ encodes both access-point proximity and group cohesion objectives within a unified framework. Diagonal entries π_{ii} represent the priority weight for placing item i close to the designated access point, while off-diagonal entries π_{ik} for $i < k$ represent the cohesion priority between items i and k when they belong to the same group (entries are typically zero for items in different groups, though non-zero values may be specified to enforce separation constraints such as hazardous material segregation).

Construction proceeds by first sorting all items globally by group priority, then by item priority within each group. Items are assigned reversed ranks where higher-priority items receive larger values: if item i has raw rank $\alpha[i]$ (with 1 being highest priority), then $\pi_{ii} = |I| - \alpha[i] + 1$. This ensures stronger pull toward the access point for higher-priority items.

For items i and k in the same group, off-diagonal entries are calculated as $\pi_{ik} = \frac{G_{\max} - \Delta_{ik}}{G_{\max}}$, where Δ_{ik} is the absolute difference between their item-level priorities and $G_{\max}$ is the size of the largest group in the instance. This formulation yields values between 0 and 1, with items having similar priorities receiving higher cohesion weights. The deliberate scaling ensures diagonal entries (integers ≥ 1) dominate off-diagonal entries (fractions < 1), maintaining emphasis on access-point proximity while encouraging group cohesion. The mathematical foundation and detailed examples of this construction are provided in Kirschenman et al. [3].

For a compact illustration, suppose an instance contains $|I| = 8$ items and item i is third in the global ordering after sorting first by group priority and then by within-group item priority. Its diagonal weight is then $\pi_{ii} = 8 - 3 + 1 = 6$. If items i and k belong to the same group, have within-group priorities 1 and 3, and the largest group size is $G_{\max} = 5$, then $\Delta_{ik} = 2$ and $\pi_{ik} = (5 - 2)/5 = 0.6$. This compact example makes the intended scaling transparent because the diagonal access-priority term remains much larger than the within-group cohesion term, so access-point priority stays dominant while cohesion still influences same-group placement. In our experiments, the group ordering follows the doctrinal or organizational loading sequence used to build the instance set, while the within-group item ordering is randomized to represent plausible tactical sequences rather than a single canonical template.

5.3. Experimental Design

Taken together, the study varies four main factors: vessel class, utilization target, the resulting grouped-equipment assignment, and CG tolerance level. Item counts are intentionally not fixed across scenarios. Because vessel capacities, equipment sizes, and group structures differ materially, fixing the item count would require distorting either utilization targets or group composition. We therefore preserve vessel geometry, grouped equipment, and utilization targets simultaneously, and treat the resulting item-count variation as an outcome of that assignment process rather than as a separate controlled factor.

5.3.1. Solution Approaches and Parameter Settings

Our experimental framework evaluates each of the three primary solution approaches presented. The Single MILP approach directly solves the full model with integrated load balancing constraints, serving as a benchmark for solution quality when computationally feasible. For the Single MILP, we set time limits equal to the number of items multiplied by 5 seconds (consistent with Kirschenman et al. [3]). If no feasible solution is found within this initial limit, we extend the time limit to three times the original duration, ensuring Single MILP receives sufficient time to compete effectively with the matheuristic approaches.

Both sliding-window approaches (standard matheuristic and in-stride balancing) employ the same underlying framework detailed in Section 4, using a 7-item window with 5-second time limits per Stage 1 iteration. Because these controls are shared by both sliding-window families, we treat the inherited $w = 7$ window and 5-second per-window limit from Kirschenman et al. [3], together with the new $k_{\text{initial}} = 6$ repair start used for Stage 2 in this paper, only as initial defaults for the balanced variant rather than as presumptively best settings for that variant. As discussed further in Section 4.5.1, the $k_{\text{initial}} = 6$ choice starts the repair process at a problem size that still solves to optimality in fractions of a second while avoiding smaller starts that add little practical value. We then stress-test this common control set over w, per-window time limit, and k_{initial} in Section 6.5. In-stride adds Stage 1 penalty terms while preserving baseline controls. If either sliding-window approach yields a Stage 1 solution that is not both feasible and load balanced, we apply the Stage 2 post-processing strategy from Section 4.5.1 under the same 5-second Stage 2 limit.

For the SW In-Stride variants, we test both Standard (penalizing only deviations outside the tolerance box) and In-Out (adding continuous pull toward optimal CG) approaches using the constant and dynamic λ schedules described in Section 4.4, with post-processing applied using the four unfixing techniques detailed in Section 4.5 when needed to achieve load balance.

5.3.2. Test Matrix

Each of the 70 instances is evaluated with four CG deviation tolerance levels ($\tau \in \{1\%, 5\%, 10\%, 15\%\}$ of vessel dimensions), creating 280 instance-tolerance combinations. The tolerance box parameters are set as $\delta_x = \tau L$ and $\delta_y = \tau W$ from the vessel length and width respectively, creating progressively larger allowable regions around the optimal center of gravity. This range identifies algorithmic breaking points, starting with an extremely restrictive 1% deviation to test computational limits, then progressively relaxing constraints to determine at which tolerances load balancing becomes trivial or naturally achieved through space utilization alone. The range allows us to examine algorithm performance from highly constrained scenarios that demand precise load distribution to more relaxed constraints that provide greater placement flexibility. For each combination, we test all applicable solution approaches with their respective parameter settings, enabling comprehensive evaluation of how solution strategies perform under varying operational requirements.

Because this paper addresses a fixed-bin prioritized-layout problem rather than a classical 2-D bin-packing objective, classical benchmark sets are not the primary evaluation environment. The relevant benchmark connection is the standard sliding-window baseline from Kirschenman et al. [3], where the same $w = 7$, 5-second constructor outperformed adapted packing, layout, and metaheuristic baselines under equal budgets, attaining best-known solutions on 76 of 90 instances. The computational focus here is whether that already-competitive prioritized layout can be guided or repaired to satisfy center-of-gravity feasibility after adding load balancing.

We also conducted a focused parameter-sensitivity study on a fixed 10-instance subset chosen to span five vessel classes and four space-utilization levels. For each selected instance and each of the four CG-deviation levels used in the main study, we evaluated the full-factorial combination of window sizes $w \in \{5, 6, 7, 8, 9\}$, per-window time limits $t_{\text{window}} \in \{5, 10\}$, and initial repair-set sizes $k_{\text{initial}} \in \{4, 6, 8\}$, yielding 1,200 experiments. Because these parameter settings are used across both SW Iterative and SW In-Stride, this focused study revalidates the common sliding-window baseline under the balanced variant and supports practical tuning guidance. The resulting sensitivity findings are reported in Section 6.5.

To examine behavior under overly restrictive load-balance targets, we also conducted a small

target-box infeasibility study on a 10-instance subset selected to span vessel classes and space-utilization levels across the same four deviation levels, yielding 80 modified cases. These modified cases preserve the original bin geometry, item set, weights, grouping, and space-utilization profile, and modify only the target CG box rather than the underlying geometry or cargo set. For each base instance-deviation combination, we generated one clearly impossible target placed beyond a conservative attainable-CG interval on one axis and one boundary-stress target placed near that interval boundary while still overlapping it. This design isolates load-balance-target infeasibility rather than trivial whole-area or geometry checks, which would normally be screened out before instance generation, and separates certification behavior from feasibility-search behavior. For these modified cases, we record conservative-screen outcomes, exact Single MILP certification outcomes, and a representative SW Iterative feasibility-search outcome. The resulting findings are reported in Section 6.3.2.

For transparency and reproducibility, the full 70-instance test bed, the named subset instances used in the sensitivity and target-box follow-up studies, and additional configuration details for the tested variants are documented in the data repository for this paper [43].

5.3.3. Computational Environment

All experiments are conducted on an AMD Threadripper Pro 7985WX system with 64 cores (128 threads) operating at a base frequency of 3.20 GHz (up to 5.10 GHz boost) and equipped with 128 GB of DDR5 RAM operating at 5200 MHz. We implement all models in Julia 1.11.6 and solve using Gurobi 12.0.2. Experimental runs for each problem instance are configured to use up to 8 dedicated threads based on preliminary testing. Across the range of instance sizes, using 8 threads balances the solver's parallelization capabilities, the CPU architecture, and the problem instance characteristics. We use default Gurobi solver parameters throughout our experiments, as initial testing with alternative parameter configurations consistently produced inferior results.

5.3.4. Proof-of-Concept Extensions

To demonstrate the framework's extensibility beyond load balancing, we conduct limited experiments on a subset of instances modified to include fixed deck obstacles and material separation constraints. We use two representative instances from the 70-instance set documented in the paper's data repository [43]: instance 24 (15 cargo items in 3 priority groups) and instance 47 (18 cargo items in 6 priority groups), both under the Runnymede vessel configuration with CG deviation tolerance level $\tau = 15\%$. These smaller instances were selected to support clear visual verification of constraint enforcement. For obstacle avoidance, each instance is augmented with two fixed obstacles chosen to challenge the packing algorithm while remaining visually interpretable and to occupy less than 10% of total deck area; for instance 24, these are a wall protrusion from the left edge (120×48 units) and a central upper-deck obstacle (200×150 units), with analogous placements used for instance 47.

For material separation, the same two instances are modified so that three cargo items per instance (items 2, 6, and 11) require a minimum 60-inch edge-to-edge separation distance, representative of hazardous-material safety spacing in military logistics. All validation layouts are generated with the Single MILP method under the same time-limit regime used in the main study. The resulting layouts are discussed in Section 6.4.

6. Numerical Results

This section presents the results of computational experiments evaluating the performance and effectiveness of the proposed solution techniques across the 70 problem instances and 280 instance-tolerance combinations described in Section 5. The results first compare cross-method trade-offs among load-

balancing success, solution quality, and computational effort; then examine solution quality and bounding behavior in more detail; and then turn to load-balancing feasibility, additional-constraint demonstrations, and the focused sensitivity follow-up.

6.1. Analysis of Performance Trade-offs

This subsection examines the fundamental trade-offs between load balancing success, solution quality, and computational efficiency. The analysis reveals key insights for practitioners selecting appropriate methods based on operational priorities and resource constraints.

6.1.1. Trade-off Between Success Rate and Solution Quality

Figure 5 presents the Pareto frontier analysis comparing load balancing success rates against solution quality across all technique configurations. The plot reveals distinct performance clusters that highlight the trade-offs inherent in different solution strategies.

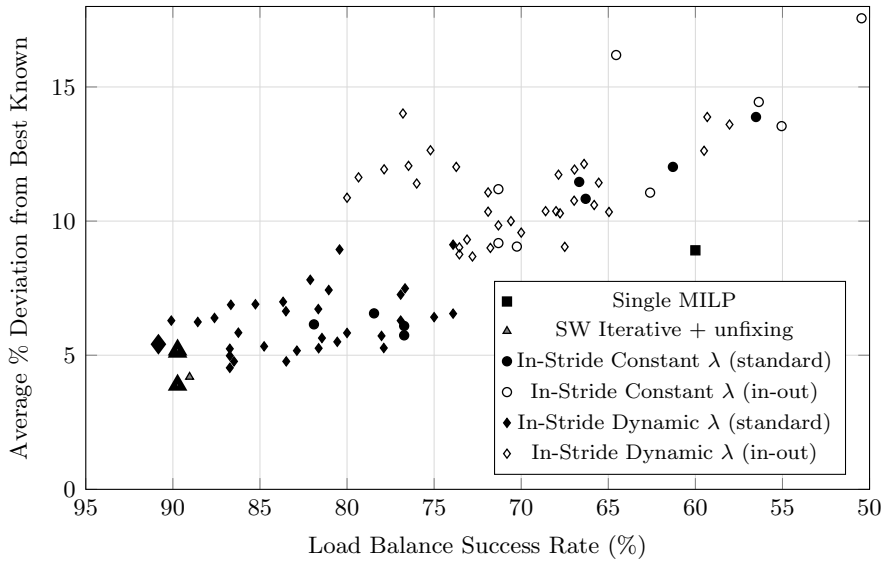

Figure 5: Pareto frontier analysis: Load balancing success rate versus solution quality. Points with thicker borders represent Pareto efficient configurations. Note the horizontal axis is reversed.

The Pareto analysis is conducted at the configuration level: each point represents a single parameter variant of a method, including its post-processing where applicable, evaluated across all 280 instance-tolerance combinations. These configuration-level success rates are therefore lower than the method-level rates reported in Tables 4 and 7, which count an instance as successful if any parameter variant of the method succeeds. Four configurations achieve Pareto efficiency: three SW Iterative variants (with DXR, GCG, and RP unfixing techniques) all achieve 89.7% configuration-level load balance success with deviations of 3.9%, 5.1%, and 5.2% respectively, while SW In-Stride with dynamic lambda schedules ($\gamma = 10.0$, $\lambda_{\max} = 0.25$, standard penalties) + GCG achieves 90.8% success and 5.4% deviation. Three of these are SW Iterative variants, demonstrating this method's dominance in achieving optimal trade-offs between reliability and solution quality.

The Single MILP approach is clearly dominated, achieving only 60.0% success rate with 8.9% average deviation (as detailed in Section 6.3.1). Standard SW In-Stride methods with dynamic λ_t schedules occupy an intermediate position, typically achieving higher success rates (73.9–90.8%) but at the cost of solution quality (approximately 6.9% average deviation), reflecting their emphasis on maintaining load balance throughout optimization.

Statistical analysis of the four Pareto efficient techniques reveals that while the SW Iterative + DXR configuration achieves numerically superior performance (3.9% deviation versus 5.1–5.4% for

other techniques), Kruskal-Wallis testing indicates these differences are not statistically significant ($H = 1.73$, $p = 0.422$). However, the consistent 1.2–1.5 percentage point advantage of DXR across all problem instances, combined with its nearly identical success rate to other SW Iterative variants, suggests practical value for operational deployments where solution quality optimization is prioritized.

At the method level, the SW Iterative method achieves a high overall success rate (97.5%), strong solution quality (2.1% median deviation), and moderate computational effort (167.5 s average solve time), with post-processing typically converging in a single iteration when using the Greedy CG Impact unfixing technique.

Table 4 provides a comprehensive comparison of all three performance metrics across solution methods.

Table 4: Performance comparison across all solution methods. Avg. Quality Gap calculated only for instances where feasible solutions were found. Success rates are method-level, counting an instance as successful if any parameter variant of the method succeeds.

Method	**Success Rate**	**Avg. Solve Time (s)**	**Avg. Quality Gap**
Single MILP	60.0%	366.8	8.9%
SW Iterative	97.5%	167.5	4.0%
SW In-Stride	98.9%	248.5	6.9%

Taken together, the computational analysis points to the SW Iterative approach as the strongest all-around trade-off rather than the pointwise best method on every metric. It combines near-best success, the strongest solution-quality profile, and lower runtime than the in-stride family, which makes it a strong candidate for practical military load planning applications.

These findings have important implications for integration with existing military planning systems like ICODES. The SW Iterative method's computational efficiency and reliability make it suitable for deployment as an optimization engine that can generate high-quality, load-balanced solutions within operationally relevant time frames. The method's consistent performance across diverse problem instances provides confidence in its robustness for real-world military logistics applications.

The performance analysis demonstrates that matheuristic approaches, particularly the SW Iterative method, consistently achieve high success rates, solution quality, and computational efficiency for complex military load planning problems. This method provides a practical foundation for automating these planning processes.

6.2. Solution Quality and Optimality Gap Analysis

This subsection evaluates the quality of solutions produced by each method, focusing on how closely they approach optimal prioritization objectives and the effectiveness of different lower bounding techniques for assessing solution quality.

6.2.1. Lower Bound Comparison

Accurate assessment of solution quality requires tight lower bounds on the optimal prioritization objective. We compare two approaches: Gurobi's built-in lower bounds from Single MILP solutions and the specialized rectangular-liquid lower bounds developed for this problem structure.

Table 5 presents the lower bound quality for the five largest problem instances by item count, examining all four CG deviation tolerance levels for each instance. While the Single MILP approach frequently failed to identify feasible load-balanced solutions for these challenging instances within the imposed time limits, we systematically extracted the strongest lower bounds achieved by Gurobi's branch-and-bound process prior to termination. The rectangular-liquid lower bounds provide an independent quality assessment across all instances, enabling comprehensive comparison of bounding

approaches.

Table 5: Comparison of lower bound quality for the five largest problem instances by item count. Values for Obj, LB, and RL-LB are scaled by 10^5 (e.g., 11.3 represents 1.13e6). % Gap columns show ((Obj − Bound)/Obj) × 100%.

Items	Groups	CG Dev	Obj	LB	RL-LB	LB Gap	RL-LB Gap
111	26	0.01	118.44	71.72	83.31	39.44%	29.66%
111	26	0.05	112.22	61.37	83.31	45.32%	25.76%
111	26	0.10	110.05	49.91	83.31	54.65%	24.30%
111	26	0.15	109.87	39.19	83.31	64.33%	24.17%
110	22	0.01	125.38	65.84	85.23	47.49%	32.02%
110	22	0.05	117.41	57.50	85.23	51.03%	27.41%
110	22	0.10	112.97	48.74	85.23	56.86%	24.56%
110	22	0.15	110.07	41.50	85.23	62.29%	22.57%
94	11	0.01	75.23	37.09	55.47	50.70%	26.27%
94	11	0.05	74.19	33.10	55.47	55.38%	25.23%
94	11	0.10	74.29	28.17	55.47	62.08%	25.33%
94	11	0.15	73.82	23.35	55.47	68.37%	24.86%
84	26	0.01	48.45	23.56	35.29	51.38%	27.16%
84	26	0.05	46.37	19.58	35.29	57.77%	23.89%
84	26	0.10	45.56	16.27	35.29	64.29%	22.54%
84	26	0.15	44.85	13.12	35.29	70.75%	21.32%
84	21	0.01	62.78	33.43	43.22	46.75%	31.16%
84	21	0.05	60.30	29.41	43.22	51.23%	28.33%
84	21	0.10	58.16	24.39	43.22	58.06%	25.69%
84	21	0.15	57.86	20.11	43.22	65.24%	25.30%

The results show that the rectangular-liquid lower bounds provide a consistently available deterministic benchmark for large-scale instances when Single MILP frequently fails to find feasible solutions within time limits. Across all 20 displayed rows, the rectangular-liquid lower bound is tighter than the solver LB while remaining below the reported objective values, indicating that it captures useful prioritization structure without requiring complete feasible solutions.

The table reveals consistent behavior across CG deviation tolerances. As expected, tighter CG deviation tolerances yield higher objective values due to increased constraint restrictions. In some cases, they also improve the solver-side lower-bound picture by shrinking the feasible region, although those solver bounds can still remain weak relative to the best available objective values. The rectangular-liquid lower-bound gap percentages remain relatively stable across different CG deviation levels for each instance, providing a consistent quality benchmark across constraint levels. This stability makes the rectangular-liquid lower bound a useful companion to the solver bounds across varying operational requirements.

To complement these deterministic bounds, we employ Wilson's statistical lower bound approach based on extreme value theory [47, 3], which estimates proximity to the theoretical optimum and provides a useful complementary perspective on solution quality when deterministic bounds may be conservative due to the complex multi-objective nature of the problem. We apply this approach to obtain complementary bounds for our three largest instances. Following the method detailed in Section 4.2.2 of Kirschenman et al. [3], we generated 100 perturbed variants for each instance-CG deviation tolerance combination by altering per-group priority hierarchies while maintaining all other problem characteristics. Each variant was solved, with objective values recalculated using the original prioritization structure to ensure consistent evaluation. Of the 12 instance-CG deviation tolerance combinations tested, all passed the required runs test for independence. A three-parameter Weibull distribution was then fitted to the observed minima, and Golden-Alt confidence intervals

were constructed for the theoretical optimum. Both Kolmogorov-Smirnov and Anderson-Darling tests confirmed satisfactory Weibull distribution fits, with the latter being particularly critical for tail accuracy in extreme value applications. Table 6 presents the statistical bounds for all qualifying combinations. Overall, these results demonstrate that our best solutions across different CG deviation levels achieve optimality gaps ranging from 1.30% to 3.95%, suggesting proximity to theoretical optima despite the challenging multi-objective nature of the problem.

Table 6: Wilson's Statistical Lower Bound Results for Qualifying Instance-CG Deviation Combinations. Min, Max, and Golden-Alt CI values are scaled by 10^6.

Instance	CG Dev	Min	Max	Golden-Alt CI	Opt. Gap
84 items	0.01	4.88	5.12	(4.76, 4.88)	1.70%
84 items	0.05	4.68	4.88	(4.57, 4.68)	1.52%
84 items	0.10	4.56	4.86	(4.41, 4.56)	3.09%
84 items	0.15	4.54	4.82	(4.38, 4.54)	2.30%
82 items	0.01	5.73	6.34	(5.63, 5.73)	1.30%
82 items	0.05	5.46	5.61	(5.36, 5.46)	2.09%
82 items	0.10	5.43	5.63	(5.29, 5.43)	2.83%
82 items	0.15	5.42	5.64	(5.30, 5.42)	2.43%
75 items	0.01	4.10	4.33	(3.93, 4.10)	3.48%
75 items	0.05	3.90	4.06	(3.77, 3.90)	2.70%
75 items	0.10	3.80	3.96	(3.65, 3.80)	3.95%
75 items	0.15	3.80	3.97	(3.66, 3.80)	2.89%

6.2.2. Distribution of Solution Quality

Figure 6 presents the distribution of solution quality across the top-performing technique configurations, measured as percentage deviation from the best-known objective value for each instance.

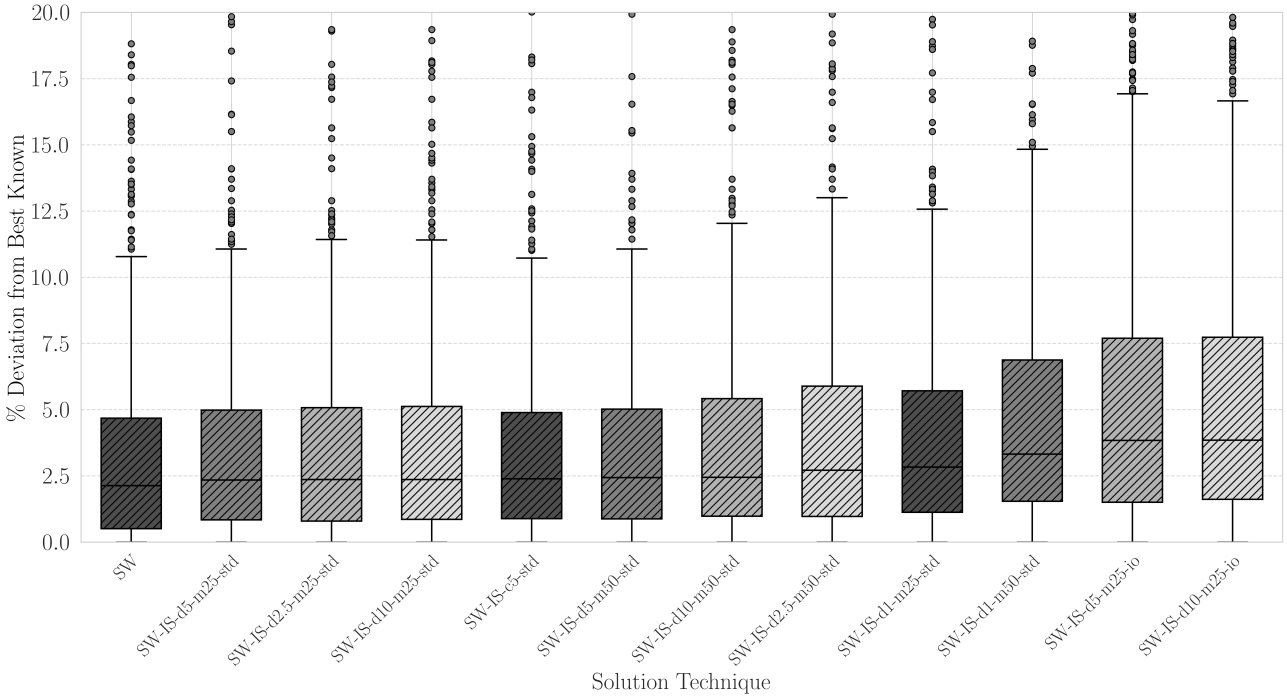

Figure 6: Distribution of solution quality (percentage deviation from best-known objective) for top-performing techniques. Abbreviations: SW = Sliding-Window Iterative (all unfixing techniques); SW-IS = Sliding-Window In-Stride with parameter notation c[X] for constant lambda schedules, d[X] for dynamic schedules with shape parameter X, m[Y] for maximum lambda values, std for standard approach (no inside penalty), and io for in-out approach (with inside penalty). Y-axis limited to 20% for clarity; some outliers exceed this range.

The SW Iterative method, aggregated across all unfixing techniques, achieves a median deviation

of 2.1% from best-known solutions. In contrast, the Single MILP approach (not shown in the figure as it falls outside the top 12 techniques when ranked by median deviation) demonstrates greater variability. This performance reflects the method's computational challenges, which often prevent finding high-quality solutions within time limits.

The SW In-Stride methods show a clear performance distinction between parameter approaches. The standard variants follow closely behind SW with median deviations ranging from 2.3% to 3.3%. In contrast, the in-out variants show degraded performance with median deviations between 3.8% and 4.2%. This confirms that penalizing items already within tolerance is counterproductive for solution quality, while the standard approach maintains both high prioritization quality and load balancing success.

Analysis of worst-performing instances provides additional insights into method limitations. The Single MILP approach's worst-case performance can reach deviations as high as 75.8% from the best-known solutions, typically occurring on highly constrained instances with tight space utilization. The SW Iterative method's occasional poor performance typically occurs on highly constrained instances with tight CG deviation tolerances, though such outliers are rare.

6.3. Load Balancing Feasibility Analysis

This subsection evaluates the fundamental capability of each solution technique to produce load-balanced solutions that satisfy the center of gravity constraints. The analysis employs a method-level aggregation approach designed to provide fair comparisons across solution techniques with multiple parameter variants.

For method-level comparisons throughout this section, we aggregate experimental results using the following logic: if any variant of a method successfully finds a load balanced solution for a given instance-tolerance combination, we count this as success for that method. Specifically, for the SW Iterative technique, success by any unfixing Stage 2 technique (Greedy CG Impact (GCG), Decreasing xr (DXR), Mass-Weighted xr (MWXR), or Reverse Priority (RP)) counts as success for the overall SW Iterative method. Similarly, for SW In-Stride variants, success by any combination of lambda schedule (constant or dynamic), penalty variant (standard or in-out), or lambda factor setting counts as success for the overall SW In-Stride method.

This aggregation strategy reflects practical deployment scenarios where method implementers would naturally select the best-performing variant of each technique. For specialized analyses comparing specific parameter choices (such as the SW In-Stride Balancing Performance analysis), we disaggregate results to examine individual variant performance. However, for fundamental capability assessments and cross-method comparisons, the aggregated approach provides the most meaningful evaluation of each method's potential effectiveness.

6.3.1. Overall Success Rate Comparison

Table 7 presents the overall load balancing success rates across all 280 unique instance-tolerance combinations, with each method's rate representing success by any of its parameter variants.

Table 7: Overall load balancing success rates by solution method. Single MILP is a single-stage method that produces a complete solution or none, yielding no intermediate results for Stage 2 adjustment.

Method	**Stage 1 Success**	**Overall Success**	**Success Rate**
Single MILP	168/280	168/280	60.0%
SW Iterative	140/280	273/280	97.5%
SW In-Stride	263/280	277/280	98.9%

The Single MILP approach achieves load balance in only 60.0% of instances, with performance

degrading significantly under tighter constraints. At 85% space utilization, the Single MILP success rate drops to 55.4%, and the method achieves success on only one Whidbey Island-class Landing Ship Dock (WIC-LSD) instance (2.1%), specifically the smallest problem instance (63 items, 80% space utilization, 10% CG deviation tolerance), demonstrating that Single MILP performance degrades severely as problem instance size increases. This poor performance stems from the solver's difficulty with the complex prioritization objective combined with global load balancing constraints, creating a computationally intractable optimization landscape. Crucially, the Single MILP approach cannot benefit from post-processing adjustments because it is not designed as a repair-oriented search pipeline. It either returns a layout that already satisfies the packing and load-balancing constraints or exits without a Stage 2 mechanism for exploiting partially satisfactory intermediate layouts. Unlike the matheuristic methods, it does not intentionally produce a feasibly packed but temporarily unbalanced layout for later correction.

In contrast, both matheuristic methods almost always achieve load balance. The SW Iterative method attains 97.5% overall success despite achieving Stage 1 load balance in only 50.0% of instances. Because Stage 1 succeeds in 140 of the 280 cases, Stage 2 is invoked on the remaining 140 cases and recovers 133 of them, yielding a 95.0% method-level recovery rate conditional on Stage 1 failure. The SW In-Stride method achieves the highest overall success rate at 98.9%, with 93.9% of instances achieving load balance directly in Stage 1.

The matheuristic methods maintain consistent performance across problem characteristics. Unlike the Single MILP approach, their performance remains above 95% across all space utilization levels and vessel types. As CG deviation tolerances become more restrictive, both methods maintain high success rates, with the SW Iterative method achieving 95.7% success even at the strictest 1% CG deviation tolerance.

6.3.2. Target-Box Infeasibility Study

To separate true load-balance-target infeasibility from search noncompletion, we ran the target-box study described in Section 5.3. Across the 80 modified cases, the conservative attainable-CG screen immediately rejected all 40 clearly impossible cases and none of the 40 boundary-stress cases, which is consistent with its intended role as a necessary-condition filter rather than a sufficient test. The exact Single MILP evaluation then automatically certified all 40 clearly impossible cases as infeasible and also certified 11 of the 40 boundary-stress cases as infeasible, while the remaining 29 boundary-stress cases reached the time limit without certification.

The representative SW Iterative evaluation served as feasibility search on the harder screen-nontrivial cases. It found a feasible balanced layout in one of the 40 boundary-stress cases and did not find one in the other 39, but those 39 outcomes remain unresolved under the matheuristic rather than certified infeasible. One boundary-stress case makes this distinction concrete because the exact method timed out, but the matheuristic found a feasible balanced layout. Taken together, the study supports a practical staged interpretation for overly restrictive target boxes in which the cheap screen is applied first, exact Single MILP certification is used when possible, and sliding-window methods are treated as feasibility-search procedures on the remaining nontrivial cases.

6.3.3. Sliding-Window In-Stride Balancing Performance

Among the SW In-Stride variants, dynamic lambda schedules achieve 98.9% overall success compared to 96.1% for constant lambda configurations. Dynamic schedules' ability to apply gentle pressure early in the matheuristic process while escalating balancing penalties for lower-priority items contributes to this performance difference.

Comparing the two in-stride approaches, the standard penalty approach (98.9% success) marginally outperforms the in-out approach (96.4% success). While the in-out approach achieves higher Stage 1 success rates (90.0% vs 86.8%), the standard approach demonstrates superior Stage 2 recovery capabilities (91.9% vs 64.3% recovery rate). This indicates that the standard approach's simpler penalty structure, which only penalizes deviations outside the tolerance zone, provides more reliable convergence to feasible solutions. Furthermore, within the in-out variant that applies continuous pull toward the optimal center of gravity, lower lambda inside factors perform better, with 0.05 outperforming 0.1 and 0.2. However, the standard approach's complete avoidance of penalties for items already within the tolerance box proves most effective, confirming that penalizing compliant placements unnecessarily compromises prioritization objectives without improving load balance success.

For practitioners, the optimal in-stride configuration combines dynamic lambda scheduling with the standard penalty approach. While specific lambda parameter tuning can provide marginal improvements, these refinements offer diminishing returns compared to the fundamental choice between matheuristic strategies.

6.3.4. Post-Processing Unfixing Efficiency

The effectiveness of the SW Iterative method depends critically on the Stage 2 post-processing strategy, as approximately half of the Stage 1 solutions fail to achieve load balance while optimizing the prioritization objective. Without this essential recovery mechanism, the method's overall load balancing success rate would be severely compromised. We evaluate post-processing effectiveness using the Stage 2 recovery rate, defined as the percentage of Stage 1 load balancing failures that are successfully corrected during iterative adjustment. Figure 7 presents the distribution of Stage 2 solve times for the four unfixing strategies across all successful load balancing attempts.

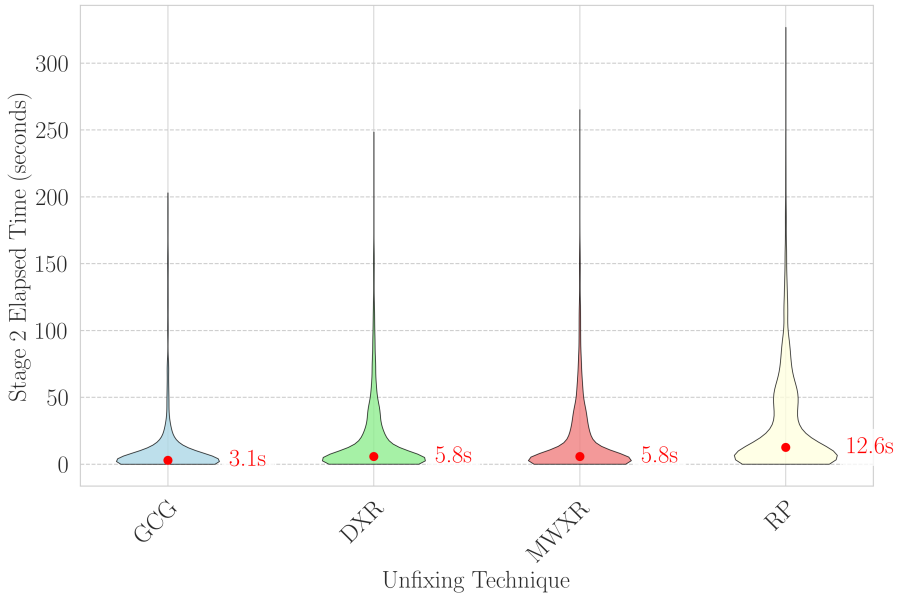

Figure 7: Distribution of Stage 2 solve times by unfixing technique for successful load balancing attempts. Marked points indicate median values. Standalone per-technique recovery rates among Stage 1 failures: GCG 71.1%, DXR 65.9%, MWXR 65.1%, and RP 65.7%.

The Greedy CG Impact technique shows greater efficiency, requiring a median of only 1 iteration compared to 4–6 iterations for the other methods. This translates to significantly faster Stage 2 solve times, with Greedy CG Impact averaging 10.8 seconds compared to 17.2–28.8 seconds for alternative techniques. The marked points in Figure 7 show median Stage 2 post-processing solve times only, while these averages summarize total Stage 2 computational effort across successful repairs. The

efficiency advantage stems from the technique's explicit consideration of each item's potential impact on correcting the center of gravity imbalance, balanced against the disruption to the prioritized layout.

Despite these efficiency differences, all unfixing techniques achieve comparable Stage 2 recovery rates (65.1% to 71.1%), meaning that when a Stage 1 solution requires post-processing adjustment, there is approximately a 65–71% probability of successfully achieving both feasible packing and load balance. These per-technique recovery rates are not directly comparable to the 95.0% method-level SW Iterative recovery rate reported above. The 95.0% value counts a Stage 1 failure as recovered if any of the four unfixing rules succeeds, whereas the 65.1–71.1% values treat each unfixing rule as a standalone Stage 2 method. This indicates that the choice of unfixing strategy primarily impacts computational efficiency rather than solution feasibility. The Greedy CG Impact technique's performance-to-effort ratio makes it a practical choice for operational deployment.

6.3.5. CG Deviation Analysis and Failed Solution Proximity

The performance of solution techniques exhibits distinct patterns across CG deviation tolerance levels. The SW Iterative method maintains high success rates across tolerance levels, achieving 95.7% success at the strictest 1% tolerance and reaching perfect or near-perfect success rates at 5% tolerance and above. This robustness stems from the method's ability to generate high-quality prioritized layouts in Stage 1, followed by targeted adjustments that preserve solution quality while achieving load balance. The Single MILP approach shows limited sensitivity to CG deviation changes, maintaining approximately 55.7–61.4% success across all tolerance levels, reflecting fundamental computational limitations rather than constraint-specific challenges. In-stride methods exhibit more complex behavior, with performance varying significantly based on specific parameter configurations, though the most reliable combinations achieve success rates comparable to the SW Iterative method at looser CG deviations.

To better understand the proximity of failed solutions to achieving load balance, we analyze the CG deviation thresholds at which unbalanced solutions would have satisfied load balancing constraints. This analysis examines solutions that failed both Stage 1 and Stage 2 load balancing at 1% and 5% CG deviations, determining the specific deviation tolerance required for each failed solution to achieve load balance. Analysis data is only available for these two strictest constraint levels, as solutions at 10% and 15% CG deviations universally achieved load balance during Stage 2 post-processing.

Figure 8 shows that solutions failing at 1% CG deviation would have achieved load balance at a median threshold of approximately 5.0%, while solutions failing at 5% CG deviation would have required a median threshold of approximately 6.5%. These findings demonstrate that many failed solutions represent near-miss cases rather than fundamental incompatibilities. The relatively modest additional tolerance required suggests that vessels equipped with ballast adjustment capabilities—which are not accounted for in these static tolerance constraints—could potentially accommodate these near-miss cases through operational countermeasures, effectively recovering solutions that appear unsuccessful under strict load balancing criteria.

6.4. Additional Operational-Constraint Demonstrations

To complement the main load-balancing study, we also carried out two small proof-of-concept validation exercises for the additional operational constraints introduced in Section 3.1. These demonstrations use instances 24 and 47 from the 70-instance set documented in the paper's data repository [43], selected because their 15- and 18-item sizes permit clear visual inspection while still representing the paper's underlying instance-generation process.

Figure 9 shows obstacle-avoidance validation for instances 24 and 47. In each case, the bottom row incorporates two fixed obstacles, shown in black, and the resulting layouts keep all cargo items out of

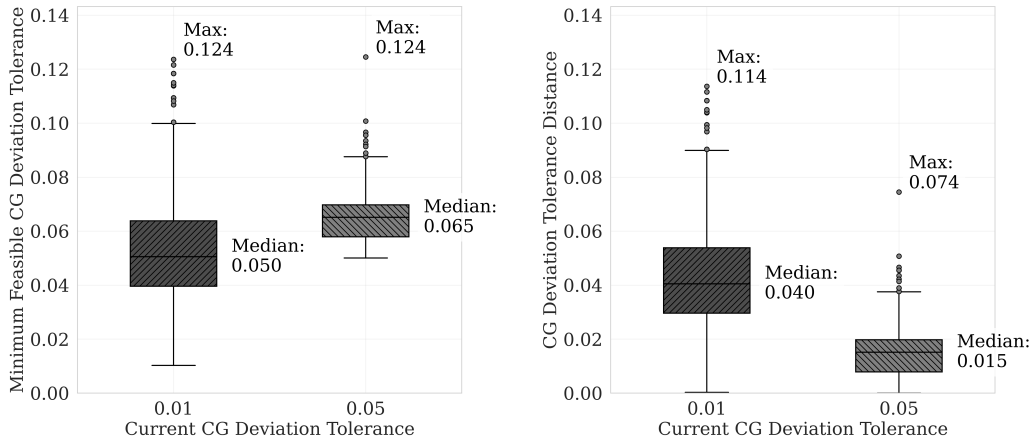

Figure 8: Distribution of CG deviation tolerance requirements for failed solutions. The left panel shows the minimum CG deviation tolerance thresholds at which failed solutions would have achieved load balance. The right panel shows the distance between actual CG deviations and the allowable tolerance thresholds for failed solutions at 1% and 5% CG deviation tolerance levels.

the prohibited regions while still satisfying the load-balancing constraints. Relative to the baseline layouts in the top row, the constrained solutions visibly adapt the packing pattern around the blocked regions without changing the underlying prioritization model.

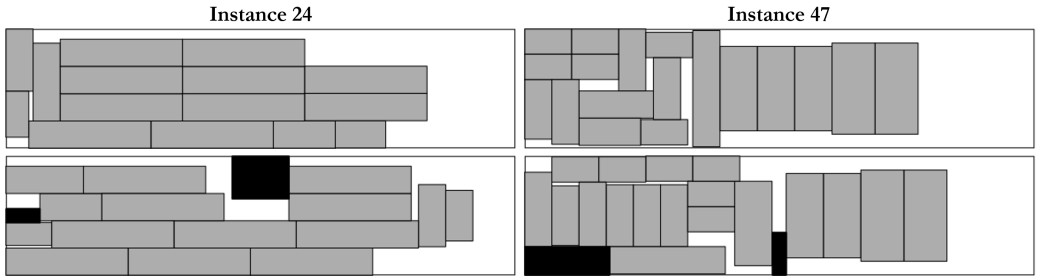

Figure 9: Obstacle avoidance validation for instances 24 (left panel) and 47 (right panel). Top row shows baseline Single MILP solutions without obstacle constraints, with all cargo items in gray fill. Bottom row shows solutions incorporating obstacle avoidance constraints, where cargo items (gray fill) successfully avoid fixed obstacles (black fill) while maintaining load balancing requirements.

Figure 10 shows material-separation validation on the same two instances. The designated hazardous-material items, shown in white, maintain the required 60-inch minimum edge-to-edge separation while the layouts continue to satisfy the packing, prioritization, and load-balancing requirements. These examples confirm that the added separation rules integrate directly into the non-overlap structure and can be activated without changing the overall solution architecture.

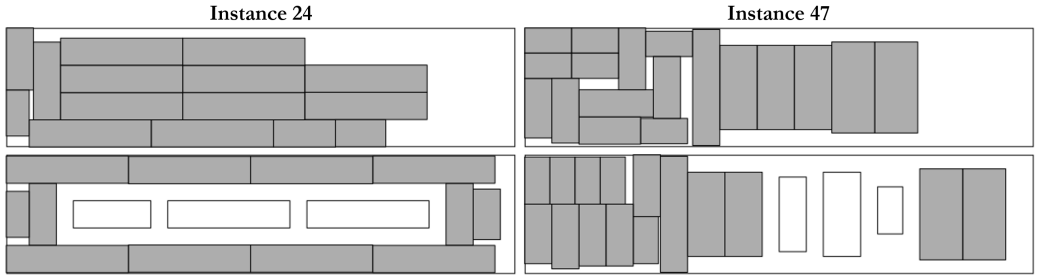

Figure 10: Material separation validation for instances 24 (left panel) and 47 (right panel). Top row shows baseline Single MILP solutions without separation constraints, with all cargo items in gray fill. Bottom row shows solutions incorporating material separation constraints, where items requiring separation (white fill) maintain the required 60-inch minimum edge-to-edge distances from each other and from regular cargo items (gray fill) while achieving load balancing.

6.5. Focused Sensitivity Follow-Up

As described in Section 5.3.2, the focused parameter-sensitivity study on the 10-instance subset assesses how the default sliding-window controls behave under the balanced variant. Across the 1,200 tested experiments, 1,176 were feasible, and all 24 infeasible experiments came from one difficult Whidbey Island-class (WIC-LSD) instance at $w = 5$ across both tested time limits, all three k_{initial} values, and all four CG-deviation levels. This is consistent with the trade-off of a very small window, which solves quickly but can commit too aggressively to local placements and leave the overall layout unrecoverable. Window size was the dominant tuning lever. The 5-item window was fastest but not robust, while $w = 6$ restored full feasibility yet still trailed $w = 7$ on quality, with the mean gap to the best solution's objective value for each case dropping from 3.18% to 1.85% as the median wall time of the best run increased from 79.5 to 150.9 seconds. Beyond that, $w = 8$ was very close to $w = 7$ in quality but slower, and $w = 9$ was strongest on quality at a 0.83% mean gap but much more expensive, with a 281.4-second median wall time of the best run. Table 8 summarizes the window-size slice of this study.

Table 8: Focused sensitivity summary by window size on the 10-instance subset. Feasibility denominator is 240 experiments per window size.

w	**Feasible**	**Mean Gap to Best Case Solution**	**Median Wall Time of Best Run (s)**
5	216	3.83%	14.1
6	240	3.18%	79.5
7	240	1.85%	150.9
8	240	1.91%	220.1
9	240	0.83%	281.4

On this subset, $w = 7$ remained the strongest practical default compromise; $w = 6$ is the main faster alternative when runtime is tight, whereas $w = 8$ or $w = 9$ are best reserved for quality-first uses that can absorb the added computation. By contrast, doubling t_{window} from 5 to 10 seconds did not recover any infeasible cases and changed the mean gap only from 2.37% to 2.21%, while increasing the median wall time of the best run from 106.8 to 144.9 seconds. Performance also showed low sensitivity to varying k_{initial} over $\{4, 6, 8\}$. Each k_{initial} setting appears in 400 experiments, all three settings were feasible in 392 of those 400, objective differences were small, and the median wall time of the best run stayed essentially unchanged near 118 seconds. We therefore retain $w = 7$, $t_{\text{window}} = 5$, and $k_{\text{initial}} = 6$ as practical defaults for this paper, but interpret them as empirically revalidated compromise settings rather than globally optimal choices.

7. Conclusion

This paper builds on the prioritized two-dimensional packing framework introduced in prior work by incorporating hard load-balancing requirements into the single-bin setting. The inherited prioritization model and sliding-window matheuristic framework remain the foundation, while the main additions here are the balance-constrained formulation, the solution mechanisms used to handle its global coupling, and the accompanying computational analysis. Across the realistic military scenarios tested here, those additions substantially improve balanced-solution attainment relative to the inherited Stage 1 constructor alone and clarify the trade-offs between proactive balance guidance, reactive repair, and exact monolithic optimization.

The proposed framework addresses a critical gap in military logistics automation. Current planning tools such as ICODES excel at feasibility checking but require extensive manual effort to achieve tactically optimized layouts. Our methods generate high-quality, load-balanced solutions that respect both combat loading priorities and physical constraints, providing superior starting points that can

drastically reduce manual planning time. At the same time, the same fixed-bin prioritized-loading structure is relevant beyond the immediate military setting, especially in roll-on/roll-off and other deck-loading environments where an assigned cargo set must be arranged under accessibility and balance requirements. The flexibility to encode multi-level priorities through the prioritization matrix enables the framework to accommodate diverse operational requirements, from amphibious assault operations requiring strict off-load sequencing to sustained logistics missions emphasizing unit cohesion.

Practical Considerations Our computational study reveals clear guidance for practitioners implementing automated load planning systems. The broader practical recommendation is to use the simpler SW Iterative plus post-processing pipeline, which attains 97.5% overall success, the strongest overall quality profile, and lower average runtime than the in-stride family. SW In-Stride remains informative as a narrower balance-guidance mechanism because it substantially raises direct Stage 1 balance attainment and slightly improves final success to 98.9%, but that marginal final gain comes with weaker prioritization quality and higher runtime. We therefore view SW In-Stride primarily as an option for settings that value direct Stage 1 balance attainment above the simpler pipeline's better all-around trade-off.

The post-processing strategy proves particularly valuable for recovering load balance when initial solutions violate center of gravity constraints. Our analysis shows that failed solutions typically require only modest additional CG deviation tolerances to achieve balance. Importantly, our matheuristic methods achieve load balance in the vast majority of cases, but even for the small percentage of instances where load balance is not achieved computationally, realistic vessel loading operations provide operational flexibility through ballast adjustment or strategic positioning of additional equipment to compensate for imbalances. This dual capability—algorithmic load balancing for most cases, with practical operational workarounds for edge cases—ensures robustness across diverse operational scenarios.

For integration with existing military planning systems, our framework offers several advantages. The matheuristic approach's predictable runtime behavior and consistent performance across diverse problem instances provides reliability essential for operational deployment. The method's ability to handle instances ranging from 10 to over 100 items without significant performance degradation ensures scalability for realistic military scenarios. Furthermore, the framework's extensibility to incorporate additional constraints such as obstacle avoidance and material separation, demonstrated in our proof-of-concept experiments, enables comprehensive modeling of military-specific requirements.

Future Directions While this study focuses on single-vessel load planning, military deployments typically involve coordinating multiple vessels with varying capabilities and destinations. Future research should extend the framework to multi-vessel scenarios, incorporating vessel selection decisions alongside spatial optimization. This extension would require balancing load distribution across vessels while maintaining tactical priorities and ensuring each vessel's configuration supports its specific mission requirements.

The integration of uncertainty into the planning process represents another important research direction. Military operations often face dynamic conditions requiring rapid replanning, such as equipment failures, changing threat assessments, or revised mission priorities. Developing robust optimization approaches that generate solutions resilient to operational uncertainties would enhance the framework's practical utility. This could include stochastic programming formulations that account for probabilistic equipment availability or robust optimization techniques that ensure feasibility across a range of operational scenarios.

Algorithmic enhancements offer additional opportunities for improvement. While our matheuristic approach demonstrates strong performance, specialized heuristics or metaheuristics tailored to the problem's structure could further reduce solution times or improve solution quality. Machine learning techniques could potentially guide matheuristic parameters or unfixing sequences based on problem characteristics, adapting the solution approach to specific instance features. Further integration with parallel computing architectures could also accelerate the solution process, particularly beneficial for large-scale deployments or real-time replanning scenarios.

The framework's application extends beyond military logistics to commercial operations with similar requirements. Port operations involving prioritized cargo handling, warehouse management with sequenced order fulfillment, and disaster relief logistics requiring rapid deployment of prioritized supplies all share characteristics with military combat loading. The limited-disruption Stage 2 repair is likewise not tied to military doctrine; it is a generic partial-reoptimization mechanism that can be layered onto other prioritized deck-loading layouts when balance feasibility must be recovered after an initial placement pass. Adapting the framework to these domains would require adjusting the prioritization structure and operational constraints while maintaining the core optimization approach.

This research demonstrates that effective integration of operational constraints into prioritized packing problems requires careful consideration of solution methodology. Our finding that matheuristic approaches dominate Single MILP optimization across all performance metrics challenges conventional wisdom about the trade-offs between solution quality and computational efficiency. By providing a practical, scalable framework for automated load planning that respects both tactical priorities and physical constraints, this work establishes a foundation for transforming military logistics planning from a labor-intensive manual process to an efficient, optimization-driven capability that enhances operational readiness and effectiveness.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used Claude (Anthropic) in order to improve readability and language. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Disclaimer

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Army, the Department of Defense, or the United States Government.

Acknowledgments

The first author was partially supported by the Omar N. Bradley Officer Research Fellowships in Mathematics.

References

[1] K. J. Pohl, P. Morosoff, ICODES: A load-planning system that demonstrates the value of ontologies in the realm of logistical command and control (C2), in: Proceedings of the Focus Symposium on Intelligent Information Management Systems, 23rd International Conference on Systems Research, Informatics and Cybernetics (InterSymp-2011), Baden-Baden, Germany, 2011.

[2] S. Goodman, J. G. Pohl, ICODES: A Ship Load-Planning System, in: Proceedings of the 13th International Ship Control Systems Symposium (SCSS), Orlando, FL, 2003, pp. 1–10.

[3] W. K. Kirschenman, H. S. Heese, M. G. Kay, R. E. King, B. M. McConnell, The 2-D orthogonal packing problem with multiple levels of prioritization: A spatial optimization perspective, 2025. Working paper, https://doi.org/10.31224/4647.

[4] M. Fischetti, M. Fischetti, Matheuristics, in: R. Martí, P. M. Pardalos, M. G. C. Resende (Eds.), Handbook of Heuristics, Springer, 2018, pp. 121–153. doi:10.1007/978-3-319-07124-4_14.

[5] Joint Chiefs of Staff, Joint Publication 3-02: Amphibious Operations, United States Department of Defense, 2019. URL: https://www.jcs.mil/Doctrine/Joint-Doctrine-Pubs/3-0-Operations-Series/.

[6] X Corps Headquarters, G-4 Summary, 15 Aug to 30 Sep 1950, Unpublished official record in X Corps War Diary Summary for Operation Chromite, 15 August to 30 September 1950, Part II, Section IV, 1951.

[7] Commander in Chief, U.S. Pacific Fleet, Project no. I.C.2. Attack Force, Unpublished official record in Interim Evaluation Report No. 1: U.S. Pacific Fleet Operations, Korean War, Volume IV, Part I, Section C, 1950.

[8] J. Klug, Establishing the realm of the possible: Logistics and military strategy, Military Strategy Magazine 9 (2023) 28–34. doi:10.64148/msm.v9i1.4.

[9] D. C. Major, C. E. Townsend Jr., A comprehensive view of deployment readiness challenges, Army Sustainment (2017). URL: https://www.army.mil/article/193244/a_comprehensive_view_of_deployment_readiness_challenges, digital Exclusive.

[10] D. C. Longhorn, S. V. Baybordi, J. T. Van Dyke, A. W. Winter, C. L. Jakes, Determining the best ship loading strategy during military deployments, Journal of Defense Analytics and Logistics 6 (2022) 98–119. doi:10.1108/JDAL-07-2022-0003.

[11] W. K. Kirschenman, B. M. McConnell, R. E. King, Automated vessel selection and combat load planning, Army Sustainment 56 (2024) 58—61. https://www.lib.ncsu.edu/resolver/1840.20/44301.

[12] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, San Francisco, 1979.

[13] A. Lodi, S. Martello, M. Monaci, Two-dimensional packing problems: A survey, European Journal of Operational Research 141 (2002) 241–252. doi:10.1016/S0377-2217(02)00123-6.

[14] G. Wäscher, H. Haußner, H. Schumann, An improved typology of cutting and packing problems, European Journal of Operational Research 183 (2007) 1109–1130. doi:10.1016/j.ejor.2005.12.047.

[15] E. Silva, J. F. Oliveira, T. Silveira, L. Mundim, M. A. Carravilla, The floating-cuts model: A general and flexible mixed-integer programming model for non-guillotine and guillotine rectangular cutting problems, Omega 114 (2023) 102738. doi:10.1016/j.omega.2022.102738.

[16] P. M. Castro, J. F. Oliveira, Scheduling inspired models for two-dimensional packing problems, European Journal of Operational Research 215 (2011) 45–56. doi:10.1016/j.ejor.2011.06.001.

[17] M. Delorme, M. Iori, S. Martello, Bin packing and cutting stock problems: Mathematical models and exact algorithms, European Journal of Operational Research 255 (2016) 1–20. doi:10.1016/j.ejor.2016.04.030.

[18] S. Polyakovskiy, R. M'Hallah, Just-in-time two-dimensional bin packing, Omega 102 (2021) 102311. doi:10.1016/j.omega.2020.102311.

[19] B. Chazelle, The bottom-left bin-packing heuristic: An efficient implementation, IEEE Transactions on Computers 32 (1983) 697–707. doi:10.1109/TC.1983.1676307.

[20] E. Hopper, B. C. Turton, An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem, European Journal of Operational Research 128 (2001) 34–57. doi:10.1016/S0377-2217(99)00357-4.

[21] A. Drira, H. Pierreval, S. Hajri-Gabouj, Facility layout problems: A survey, Annual Reviews in Control 31 (2007) 255–267. doi:10.1016/j.arcontrol.2007.04.001.

[22] T. C. Koopmans, M. Beckmann, Assignment problems and the location of economic activities, Econometrica 25 (1957) 53–76. doi:10.2307/1907742.

[23] Y. A. Bozer, R. D. Meller, A reexamination of the distance-based facility layout problem, IIE Transactions 29 (1997) 549–560. doi:10.1080/07408179708966365.

[24] R. D. Meller, V. Narayanan, P. H. Vance, Optimal facility layout design, Operations Research Letters 23 (1998) 117–127. doi:10.1016/S0167-6377(98)00024-8.

[25] J.-Y. Kim, Y.-D. Kim, Graph theoretic heuristics for unequal-sized facility layout problems, Omega 23 (1995) 391–401. doi:10.1016/0305-0483(95)00016-H.

[26] C. Iris, D. Pacino, A survey on the ship loading problem, in: International Conference on computational Logistics, Springer, 2015, pp. 238–251. doi:10.1007/978-3-319-24264-4_17.

[27] H. Shen, C. Liu, Y. Yin, X. Sun, Automatic stowage of bulk carrier based on ship floating-state control, Journal of Ship Research 64 (2020) 298–312. doi:10.5957/JOSR.11170070.
[28] X. Sun, S. Wang, Z. Wang, C. Liu, Y. Yin, A semi-automated approach to stowage planning for Ro-Ro ships, Ocean Engineering 247 (2022) 110648. doi:10.1016/j.oceaneng.2022.110648.
[29] A. Trivella, D. Pisinger, The load-balanced multi-dimensional bin-packing problem, Computers & Operations Research 74 (2016) 152–164. doi:10.1016/j.cor.2016.04.020.
[30] V. Lurkin, M. Schyns, The airline container loading problem with pickup and delivery, European Journal of Operational Research 244 (2015) 955–965. doi:10.1016/j.ejor.2015.02.027.
[31] I. Moon, T. V. L. Nguyen, Container packing problem with balance constraints, OR Spectrum 36 (2014) 837–878. doi:10.1007/s00291-013-0356-1.
[32] A. G. Ramos, E. Silva, J. F. Oliveira, A new load balance methodology for container loading problem in road transportation, European Journal of Operational Research 266 (2018) 1140–1152. doi:10.1016/j.ejor.2017.10.050.
[33] M. Alonso, R. Alvarez-Valdes, M. Iori, F. Parreño, J. Tamarit, Mathematical models for multicontainer loading problems, Omega 66 (2017) 106–117. doi:10.1016/j.omega.2016.02.002.
[34] M. Gajda, A. Trivella, R. Mansini, D. Pisinger, An optimization approach for a complex real-life container loading problem, Omega 107 (2022) 102559. doi:10.1016/j.omega.2021.102559.
[35] A. P. Davies, E. E. Bischoff, Weight distribution considerations in container loading, European Journal of Operational Research 114 (1999) 509–527. doi:10.1016/S0377-2217(98)00139-8.
[36] M. Eley, Solving container loading problems by block arrangement, European Journal of Operational Research 141 (2002) 393–409. doi:10.1016/S0377-2217(02)00133-9.
[37] E. E. Bischoff, M. Ratcliff, Issues in the development of approaches to container loading, Omega 23 (1995) 377–390. doi:10.1016/0305-0483(95)00015-G.
[38] D. Patsiatzis, G. Knight, L. Papageorgiou, An MILP approach to safe process plant layout, Chemical Engineering Research and Design 82 (2004) 579–586. doi:10.1205/026387604323142612.
[39] M. Iori, V. L. De Lima, S. Martello, F. K. Miyazawa, M. Monaci, Exact solution techniques for two-dimensional cutting and packing, European Journal of Operational Research 289 (2021) 399–415. doi:10.1016/j.ejor.2020.06.050.
[40] S. DorMohammadi, M. Rais-Rohani, Exponential penalty function formulation for multilevel optimization using the analytical target cascading framework, Structural and Multidisciplinary Optimization 47 (2013) 599–612. doi:10.1007/s00158-012-0861-x.
[41] Tapestry Solutions, Inc., JECD Codes: Definitions and Descriptions for ICODES Cargo Category and Shipping Configuration Codes, 2024. Internal documentation PDF for ICODES JECD.
[42] Headquarters, Department of the Army, Department of the Army Pamphlet 71–32: Force Development and Documentation Consolidated Procedures, Headquarters, Department of the Army, 2019. URL: https://armypubs.army.mil/epubs/DR_pubs/DR_a/ARN44160-PAM_71-32-002-WEB-4.pdf.
[43] W. K. Kirschenman, H. S. Heese, M. G. Kay, R. E. King, B. M. McConnell, Military maritime load planning instances for prioritized two-dimensional orthogonal packing, Dryad Repository, 2025. doi:10.5061/dryad.vt4b8gv5z.
[44] U.S. Navy, Fact file: Amphibious assault ships - LHD/LHA(R), 2025. Accessed 2026-02-13. https://www.navy.mil/Resources/Fact-Files/Display-FactFiles/Article/2169814/amphibious-assault-ships-lhdlhar/.
[45] U.S. Navy, Fact file: Dock landing ship - LSD, 2025. Accessed 2026-02-13. https://www.navy.mil/Resources/Fact-Files/Display-FactFiles/Article/2169901/dock-landing-ship-lsd/.
[46] Headquarters, Department of the Army, Army Watercraft Operations, U.S. Army Training and Doctrine Command, 2015. Accessed 2026-02-13, https://armypubs.army.mil/epubs/DR_pubs/DR_a/pdf/web/atp4_15.pdf.
[47] A. D. Wilson, R. E. King, J. R. Wilson, Case study on statistically estimating minimum makespan for flow line scheduling problems, European Journal of Operational Research 155 (2004) 439–454. doi:10.1016/S0377-2217(02)00910-4.