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. This paper extends a prioritized two-dimensional orthogonal packing framework to address the specific operational constraints of military logistics, incorporating global load balancing requirements alongside existing prioritization objectives.

We introduce three solution techniques 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. The proposed methods generate high-quality, load-balanced solutions for single-vessel scenarios in approximately two 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

*Corresponding author at: Department of Industrial and Systems Engineering, North Carolina State University, Campus Box 7906, Raleigh, NC, United States. Email: wkkirsch@ncsu.edu

Email addresses: wkkirsch@ncsu.edu (William K. Kirschenman), hsheese@ncsu.edu (H. Sebastian Heese), kay@ncsu.edu (Michael G. Kay), king@ncsu.edu (Russell E. King), mcconnell@ncsu.edu (Brandon M. McConnell)

scale of the effort impeded operations (Pohl and Morosoff, 2011). 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 (Goodman and Pohl, 2003; Pohl and Morosoff, 2011). 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. To bridge this gap between feasibility checking and tactical optimization, we propose an automated approach that generates high-quality initial layouts.

This research extends the prioritized two-dimensional orthogonal packing framework of Kirschenman et al. (2025a), which introduced a novel objective function to balance item placement near access points with the need to maintain cohesion among functionally related groups, to address the specific needs of military combat loading. Here, we adapt that model to incorporate critical operational constraints inherent to maritime military logistics. The selection of these constraints—fixed deck obstacles, material separation rules, and load balancing—is motivated by their importance in ensuring a safe and viable load plan, as they represent key feasibility considerations within planning systems like ICODES. The inclusion of load balancing, in particular, transforms the problem by imposing a global constraint that couples the placement of all items, a stark contrast to the local, geometric nature of standard packing constraints.

This global coupling presents a significant computational challenge, particularly for the matheuristic solution methods that proved highly effective for the standard prioritized packing problem (Kirschenman et al., 2025a). 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 (Fischetti and Fischetti, 2018). 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 present an extended mathematical formulation for the prioritized two-dimensional packing problem that incorporates load balancing and other operational constraints. Second, we design, implement, and conduct a comprehensive computational study of novel in-stride and post-processing matheuristic strategies for solving this complex, globally constrained packing problem. Third, our analysis provides

practical insights into the trade-offs between solution quality and computational effort, identifying robust automated techniques that consistently produce high-quality, balanced layouts for realistic military scenarios.

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 doctrines that prioritize operational effectiveness over pure space utilization. The concept of combat loading requires arranging equipment on transport assets such that it can be off-loaded rapidly, in the tactical sequence required for immediate employment, and with unit integrity maintained (Joint Chiefs of Staff, 2019). This contrasts sharply with administrative loading, which typically focuses on maximizing cargo density. The success of historical large-scale amphibious landings depended heavily on meticulous embarkation plans that ensured forces arrived ashore cohesively and ready for action (X Corps Headquarters, 1951). With a renewed focus on LSCO, these principles remain critically relevant (Klug, 2023; Townsend, 2017).

Historical validation of these principles comes from extensive archival research into major Pacific theater operations, including Operation Chromite (Inchon Landing), Operation King Two (Leyte Landing), and Operation Mike (Lingayen Gulf Landing). Analysis of records from the MacArthur Memorial Museum Library and Archives, combined with detailed 1st Marine Division documentation from the National Archives and Records Administration (NARA), identified the critical lesson that "combat elements and their combat material go over the beaches together" as a key planning requirement for future operations (X Corps Headquarters, 1951). Historical after-action reports documented worst-case space utilization rates of "thirty-five percent (35%) efficient when so loaded," translating to a 65% space utilization rate (Commander in Chief, U.S. Pacific Fleet, 1950) when inappropriate naval vessels were employed for amphibious assault. This finding informed our experimental range of 65% to 85% space utilization, with the upper bound limited by feasibility constraints observed in prior testing. These reports also emphasized the tension between Navy requirements for rapid ship unloading to minimize vessel exposure and landing force needs for preserving tactical sequence. Importantly, documentation revealed that failure to maintain proper loading sequence resulted in "dangerous concentration

of non-segregated cargo around the unloading point which produced troop logistic problems and potential losses" (Commander in Chief, U.S. Pacific Fleet, 1950), directly validating the emphasis on unit integrity and group-based priority structures in modern optimization approaches.

The primary tool used by the U.S. DoD for load planning is the Integrated Computerized Deployment System (ICODES) (Goodman and Pohl, 2003). 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 (Pohl and Morosoff, 2011). 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 (Longhorn et al., 2022; Kirschenman et al., 2024), a process that is time-consuming and offers no guarantee of optimality or near-optimality. Optimization approaches combining integer programming with purpose-built heuristics have proven effective for related military logistics challenges, including automated air load planning (Heidelberg et al., 1998), naval replenishment-at-sea scheduling (Brown et al., 2017), and weapon-target assignment (Kline et al., 2020).

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 (Garey and Johnson, 1979). Classical bin packing literature is vast, with objectives centered on spatial efficiency, such as minimizing the number of bins or maximizing packing density (Lodi et al., 2002; Wäscher et al., 2007; Lewis and Parker, 1982). The geometric nature of the problem has led to a variety of solution methods, including mixed-integer linear programming (MILP) formulations (Castro and Oliveira, 2011; Delorme et al., 2016) and a wide range of heuristics (Chazelle, 1983; Hopper and Turton, 2001). 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 (Drira et al., 2007). Early FLP models were often formulated as a Quadratic Assignment Problem (QAP) (Koopmans and Beckmann, 1957; Pierce and Crowston, 1971; Bazaraa and Sherali, 1980), while later work on the Unequal-Area Facility Layout Problem (UA-FLP) allowed for more flexible, non-uniform shapes (Bozer and Meller, 1997; Meller et al., 1998). 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. (2025a) bridge 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. The present paper uses this prioritized packing model and its matheuristic framework as the starting point for incorporating further operational constraints.

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. 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 (Trivella and Pisinger, 2016; Lurkin and Schyns, 2015). Others treat it as a hard constraint, requiring the CG to lie within a predefined tolerance box (Moon and Nguyen, 2014; Ramos et al., 2018). These methods can be integrated into the initial packing process (Trivella and Pisinger, 2016) or applied as a post-processing step to adjust an existing layout (Davies and Bischoff, 1999; Eley, 2002; Akyüz and Lee, 2014). However, most of this research exists in the context of standard packing objectives or 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.

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 (Lodi et al., 2002; Meller et al., 1998). Material separation, for ensuring minimum distances between hazardous items, has been modeled in related layout problems using modified non-overlap constraints (Patsiatzis et al., 2004). Our experimental campaign focuses on load balancing performance, while obstacle avoidance and material separation are demonstrated in proof-of-concept experiments (Appendix A).

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, 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 i is entirely to the left of item k (i.e., the right edge of i, x_i^r, lies to the left of the left edge of k, x_k), and analogously for $a_{ik}^y = 1$ in the y-dimension. 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:

$$\min \quad \sum_{i \in I} \sum_{k \in I,\, i \leq k} \pi_{ik} \left(\rho_{ik}^x + \rho_{ik}^y \right) \tag{1}$$

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

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

$$a_{ik}^x + a_{ki}^x + a_{ik}^y + a_{ki}^y \geq 1 \qquad \forall i, k \in I, i < k \tag{4}$$

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

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

$$x_k^r \leq x_i + (1 - a_{ik}^x) L \qquad \forall i, k \in I, i \neq k \tag{7}$$

$$x_i^r \leq x_k + (1 - a_{ki}^x) L \qquad \forall i, k \in I, i \neq k \tag{8}$$

$$y_k^r \leq y_i + (1 - a_{ik}^y)W \qquad \forall i, k \in I, i \neq k \tag{9}$$

$$y_i^r \leq y_k + (1 - a_{ki}^y)W \qquad \forall i, k \in I, i \neq k \tag{10}$$

$$\frac{x_i + x_i^r}{2} - \frac{x_k + x_k^r}{2} \leq \rho_{ik}^x \qquad \forall i, k \in I,\ i < k \tag{11}$$

$$\frac{x_k + x_k^r}{2} - \frac{x_i + x_i^r}{2} \leq \rho_{ik}^x \qquad \forall i, k \in I,\ i < k \tag{12}$$

$$\frac{y_i + y_i^r}{2} - \frac{y_k + y_k^r}{2} \leq \rho_{ik}^y \qquad \forall i, k \in I,\ i < k \tag{13}$$

$$\frac{y_k + y_k^r}{2} - \frac{y_i + y_i^r}{2} \leq \rho_{ik}^y \qquad \forall i, k \in I,\ i < k \tag{14}$$

$$\frac{x_i + x_i^r}{2} - x^o \leq \rho_{ii}^x \qquad \forall i \in I \tag{15}$$

$$x^o - \frac{x_i + x_i^r}{2} \leq \rho_{ii}^x \qquad \forall i \in I \tag{16}$$

$$\frac{y_i + y_i^r}{2} - y^o \leq \rho_{ii}^y \qquad \forall i \in I \tag{17}$$

$$y^o - \frac{y_i + y_i^r}{2} \leq \rho_{ii}^y \qquad \forall i \in I \tag{18}$$

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

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

$$B_{\texttt{opt}}^x - \delta_x \leq B^x \leq B_{\texttt{opt}}^x + \delta_x \tag{21}$$

$$B_{\texttt{opt}}^y - \delta_y \leq B^y \leq B_{\texttt{opt}}^y + \delta_y \tag{22}$$

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. Non-overlap is enforced by constraint (4), which dictates that any two items must be separated in at least one dimension, and constraints (7)–(10), which enforce the geometry of that separation. Constraints (5) and (6) manage item rotation. The series of constraints (11)–(18) define the rectilinear distance variables ρ used in the objective. Finally, constraints (19)–(22) enforce load balancing. The CG coordinates B^x and B^y are calculated as the mass-weighted average of the items' centroids, and these coordinates are then constrained to fall within the specified tolerance box around the target CG. This method of linearizing the CG calculation and using it as a hard constraint is inspired by similar approaches in the load-balanced packing literature (Trivella and Pisinger, 2016), but applied here to our non-standard prioritization objective.

3.1. Modeling Additional Operational Constraints

While not the focus of our main experimental campaign, our model can be readily extended to handle fixed obstacles and material separation requirements—two key constraints enforced by ICODES Single Load Planner alongside load balancing. Although our experimental evaluation concentrates on load balancing due to its nature as a global constraint that couples all placement decisions, Appendix A presents the mathematical formulations for obstacle avoidance and material separation, demonstrating that these constraints can be seamlessly integrated into our

prioritization framework through straightforward extensions of the non-overlapping constraints. That appendix also provides proof-of-concept validation experiments confirming successful constraint enforcement.

4. Solution Techniques

This section presents the solution methods developed to address the prioritized two-dimensional packing problem with load balancing constraints. Figure 1 provides a high-level roadmap before the technical subsections. 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 lower bound (GB-LB), shown with dashed arrows in Figure 1, guides penalty scaling for the in-stride approach and provides a metric for assessing solution quality. 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. The Single MILP approach, when successful, directly yields a balanced solution without requiring post-processing. If either the Stage 1 Single MILP or the Stage 2 post-processing cannot produce a feasible, load-balanced layout within the prescribed limits, we treat the instance as infeasible for that specific solution approach. 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 lower bound, a foundational concept that both guides our matheuristic strategies and enables solution quality assessment. Section 4.3 reviews the sliding-window matheuristic framework from prior work (Kirschenman et al., 2025a), 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.

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

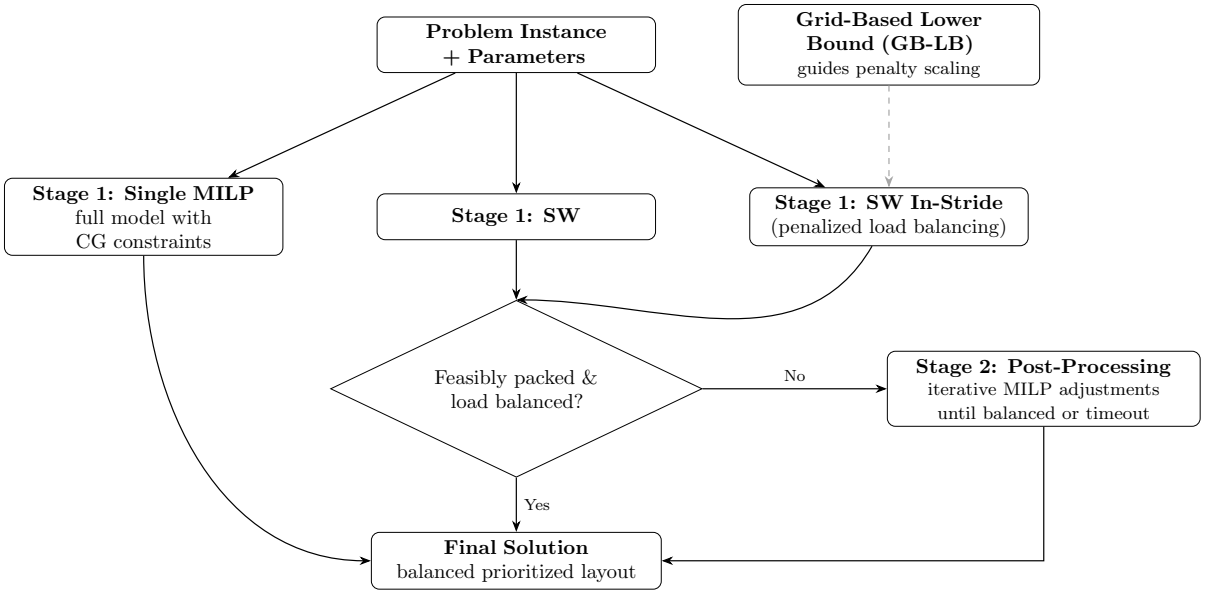

Figure 1: Solution techniques roadmap illustrating the two-stage framework. Stage 1 employs one of three approaches (Single MILP, SW Iterative, or SW In-Stride), with guidance from the grid-based lower bound (GB-LB; dashed arrows). Stage 2 applies post-processing adjustments when Stage 1 solutions are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

the model, any feasible solution found by this approach is guaranteed to satisfy the center of gravity requirements.

However, the computational burden of this approach grows rapidly with problem size. The combination of geometric packing constraints, distance calculations, and global load balancing requirements creates a highly complex search space. Our preliminary experiments reveal that instances with more than 9 items often require hours to solve to optimality, while instances exceeding a dozen items may not find provably optimal solutions within reasonable time limits. For instance, Kirschenman et al. (2025a) demonstrated that for the base prioritization problem without load balancing constraints, standalone 8-item subproblems averaged approximately 1.6 minutes to prove optimality, while 9-item subproblems averaged over 21 minutes, illustrating the exponential growth in computational complexity even for relatively small instances. The addition of global load balancing constraints in the present work further exacerbates this computational challenge. This intractability motivates the development of matheuristic approaches that can produce high-quality solutions more efficiently.

4.2. Grid-Based Lower Bound

Before presenting our matheuristic strategies, we introduce the grid-based lower bound (GB-LB) from Kirschenman et al. (2025a) which serves two critical roles in our enhanced solution methods: providing guidance for penalty scaling in the in-stride approaches and enabling rigorous solution quality assessment. We present the bound's construction and properties here, as understanding its mechanics is essential for comprehending how it guides the matheuristic strategies that follow.

The GB-LB exploits a fundamental geometric property: Euclidean distances never exceed their rectilinear counterparts ($\|u - v\|_2 \leq \|u - v\|_1$). By constructing a placement scenario using Euclidean distances, we thus obtain a valid lower bound for our rectilinear distance objective.

The bound assumes items can be decomposed into unit squares that pack radially around the access point in order of priority, ignoring standard packing constraints but respecting the prioritization structure.

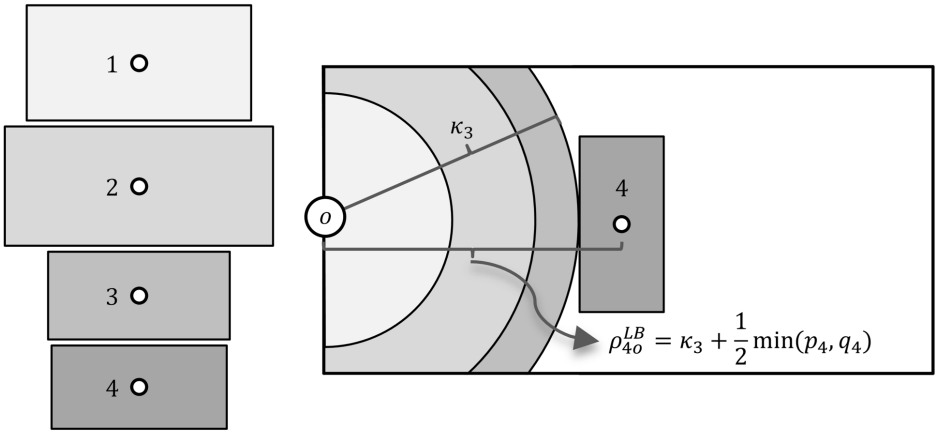

Figure 2: Grid-Based Lower Bound (GB-LB) construction showing radial packing of unit squares and exclusion radius calculation. Figure adapted from Kirschenman et al. (2025a).

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 lower bound distance for item i to the access point is

$$\rho_{ii}^{LB} = \kappa_{i-1} + \frac{1}{2}\min\{p_i, q_i\}. \tag{23}$$

For item-to-item distances, we use the conservative bound based on minimum dimensions,

$$\rho_{ik}^{LB} = \frac{1}{2}\min\{p_i, q_i\} + \frac{1}{2}\min\{p_k, q_k\}, \tag{24}$$

so the total GB-LB is then

$$\text{GB-LB} = \sum_{i=1}^{n} \pi_{ii}\rho_{ii}^{LB} + \sum_{i<k} \pi_{ik}\rho_{ik}^{LB}. \tag{25}$$

This bound 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 GB-LB 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 GB-LB 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. (2025a), 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,

$$s_x^+ \geq B_x - B_{\text{opt}}^x - \delta_x, \tag{26}$$

$$s_x^- \geq B_{\text{opt}}^x - B_x - \delta_x, \tag{27}$$

$$s_y^+ \geq B_y - B_{\text{opt}}^y - \delta_y, \text{ and} \tag{28}$$

$$s_y^- \geq B_{\text{opt}}^y - B_y - \delta_y, \tag{29}$$

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

$$\min \sum_{i \in I_t^{\text{free}}} \sum_{k \in I_t^{\text{free}},\, i \leq k} \pi_{ik}(\rho_{ik}^x + \rho_{ik}^y) + \lambda_t \cdot (s_x^+ + s_x^- + s_y^+ + s_y^-). \tag{30}$$

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. (2025a) that optimizes only the prioritization objective, while larger values increasingly emphasize center of gravity alignment.

4.4.2. Lambda Scaling with GB-LB

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 lower bound to normalize the load balancing penalties. For iteration t, we compute the GB-LB for the current subproblem's free items and scale lambda as

$$\lambda_t = \lambda_{\text{factor}} \cdot \frac{\text{GB-LB}_t}{2(\delta_x + \delta_y)}, \tag{31}$$

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 (2013), gradually increasing the load balancing emphasis with

$$\lambda_{\text{factor}}(t) = \lambda_{\max} \cdot \frac{e^{a(t/(N-b))}}{e^{a(1-b)+c}}, \tag{32}$$

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 (30). 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

$$\min \sum_{i,k} \pi_{ik}(\rho^x_{ik} + \rho^y_{ik}) + \lambda^{\text{out}}_t(s^+_x + s^-_x + s^+_y + s^-_y) + \lambda^{\text{in}}_t(u_x + u_y), \tag{33}$$

where u_x and u_y are auxiliary variables representing the absolute deviations, constrained by

$$u_x \geq B_x - B^x_{\text{opt}}, \tag{34}$$

$$u_x \geq B^x_{\text{opt}} - B_x, \tag{35}$$

$$u_y \geq B_y - B^y_{\text{opt}}, \text{ and} \tag{36}$$

$$u_y \geq B^y_{\text{opt}} - B_y. \tag{37}$$

The parameter λ^{out}_t follows the scaling from equation (31) 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 $(x^{(1)}_i, y^{(1)}_i)$ 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.

Algorithm 1 Iterative Balance Adjustment

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

Output: Load-balanced coordinates $\{(x_i^*, y_i^*)\}_{i \in I}$ or infeasibility declaration

Generate unfixing sequence (ordered list of item indices) using techniques from Section 4.5
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 infeasible ▷ All items exhausted without achieving balance

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 approach orders items by decreasing right-edge coordinate (x_i^r), prioritizing items farthest from the access point. For our experimental setup using a center-left edge access point, items with larger x_i^r values are farthest from the access point. This technique adapts to different access point locations: for a center-bottom access point, we would use decreasing y_i^r values, while for a bottom-left corner access point, we would combine both x_i^r and y_i^r coordinates. The intuition is that items farther from the access point contribute less to the prioritization objective if moved, as their placement is already distant from optimal. Unfixing

these items first preserves the near-optimal placements of high-priority items near the access point, allowing post-processing to achieve load balance with minimal degradation to overall solution quality.

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,

$$\text{score}_i = \alpha \cdot \frac{x_i^r}{\max_j x_j^r} + (1-\alpha) \cdot \frac{m_i}{\max_j m_j} \cdot \text{moveable}_i, \tag{38}$$

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 $U \neq \emptyset$ **do**
    for each item $i \in U$ **do**
        Calculate maximum moveable distance d_i for item i in primary correction direction, considering bin boundaries and other items
        $\text{Impact}_i = \frac{m_i}{M^E} \cdot d_i$ ▷ CG correction potential
        $\text{Cost}_i = \pi_{ii} \cdot d_i + \sum_{k \neq i} \pi_{ik} \cdot d_i$ ▷ Prioritization disruption
        $\text{Score}_i = \frac{\text{Impact}_i}{\text{Cost}_i + \varepsilon}$ ▷ $\varepsilon > 0$ prevents division by zero
    end for
    Select $i^* = \arg\max_{i \in U} \text{Score}_i$
    Append i^* to sequence
    Update CG assuming i^* moves full distance d_{i^*}
    Remove i^* from U
end while

This greedy approach balances the potential for CG correction against the disruption to the prioritized layout, selecting items that offer the best trade-off at each step. The movement

distance calculation considers both physical constraints (bin boundaries, other items) and the distance needed to correct the CG imbalance.

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 (Tapestry Solutions, Inc., 2024; Headquarters, Department of the Army, 2019). 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. (2025b).

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 (Headquarters, Department of the Army, 2019).

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 1. Vessel dimensions come from official U.S. Navy (U.S. Navy, 2025a,b) and U.S. Army (Headquarters, Department of the Army, 2015) specifications. Detailed dimension derivations are in Kirschenman et al. (2025b). 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 1: Vessel specifications used in instance generation. Dimensions obtained from official U.S. Navy and Army sources (U.S. Navy, 2025a,b; Headquarters, Department of the Army, 2015).

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. The CG deviation

tolerances (δ_x, δ_y) are defined as percentages of the vessel dimensions. This percentage-based approach 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. (2025b).

Table 2 summarizes the characteristics of the generated instances, showing the diversity achieved through this process.

Table 2: 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 π 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. (2025a).

5.3. Experimental Design

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. (2025a)). 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. Based on findings from Kirschenman et al. (2025a), this configuration provides an effective balance between solution quality and computational efficiency among several Pareto-efficient configurations. The in-stride balancing strategy augments this framework by adding penalty terms to the Stage 1 objective function to provide load balancing pressure, while maintaining the same matheuristic parameters. When Stage 1 solutions from either approach are not both feasible and load balanced, we apply the post-processing strategy described in Section 4.5.1 with 5-second time limits per Stage 2 iteration, beginning with $k = 6$ items.

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 (1%, 5%, 10%, and 15% of vessel dimensions), creating 280 instance-tolerance combinations. The tolerance box parameters δ_x and δ_y are set as percentages of vessel width and height 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.

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. These tests, detailed in Appendix A, validate the model's ability to handle additional operational constraints using the formulations presented therein.

As discussed in Section 2.3, these constraints integrate directly into the non-overlap logic. The main experimental campaign focuses on load balancing, which represents the most computationally challenging operational constraint due to its global coupling of all placement decisions. This experimental design enables comprehensive evaluation of how different solution strategies perform under varying operational constraints.

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 analysis focuses on three core research questions: which methods most reliably achieve load balance, how solution quality compares across techniques, and what computational trade-offs exist between competing approaches.

6.1. 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.

6.1.1. Overall Success Rate Comparison

Table 3 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 3: 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 yields one of two outcomes: either a fully feasible solution that already satisfies both packing and load balancing constraints, or a completely infeasible solution with no decision variable values to adjust. Unlike the matheuristic methods that can produce solutions that are feasibly packed but not yet load balanced, the Single MILP provides no intermediate solutions suitable for post-processing 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. This demonstrates the effectiveness of the post-processing adjustment strategy, which recovers load balance in 97.5% of instances that fail Stage 1. 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.

6.1.2. 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.

6.1.3. 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 3 presents the distribution of Stage 2 iterations and solve times for the four unfixing strategies across all successful load balancing attempts.

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 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. This indicates that the choice of unfixing strategy primarily impacts

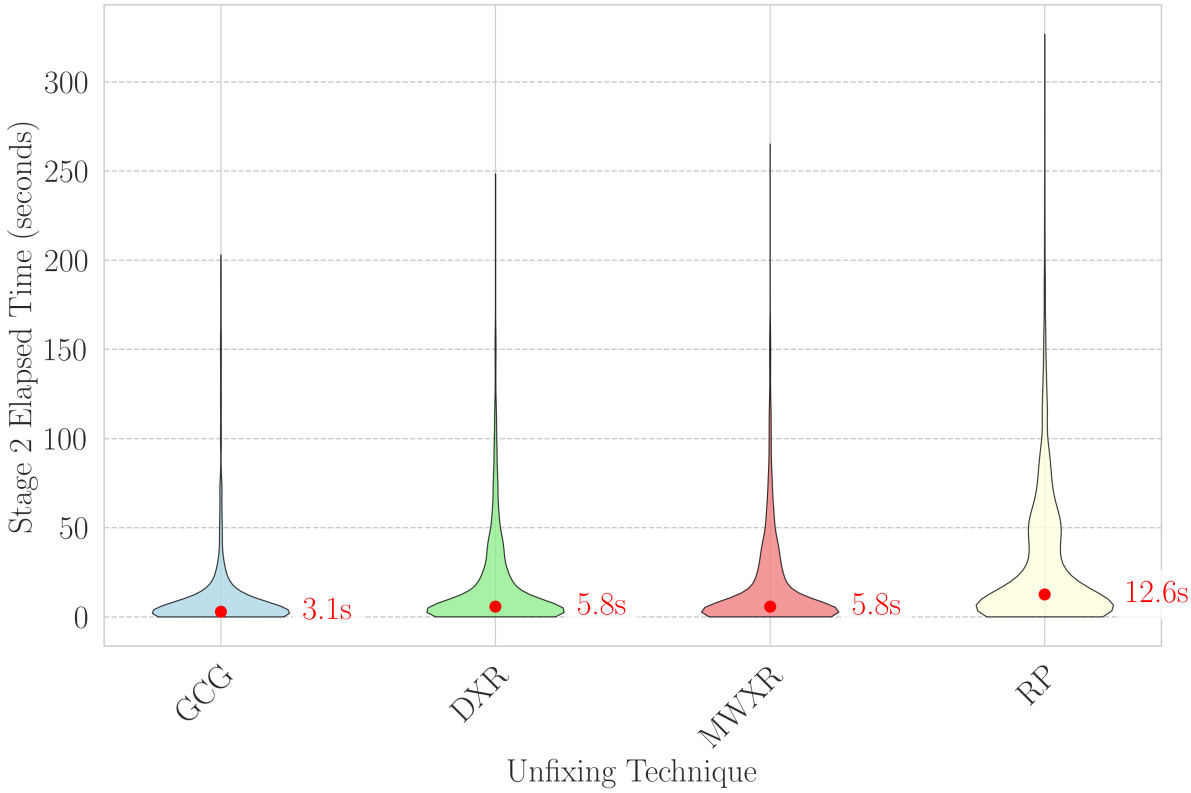

Figure 3: Distribution of Stage 2 iterations and solve times by unfixing technique for successful load balancing attempts. Marked points indicate median values. Unfixing technique success rates: Greedy CG Impact (GCG) 71.1%, Decreasing xr (DXR) 65.9%, Max-Weight xr (MWXR) 65.1%, Reverse Priority (RP) 65.7%.

computational efficiency rather than solution feasibility. The Greedy CG Impact technique's performance-to-effort ratio makes it a practical choice for operational deployment.

6.1.4. 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 4 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.

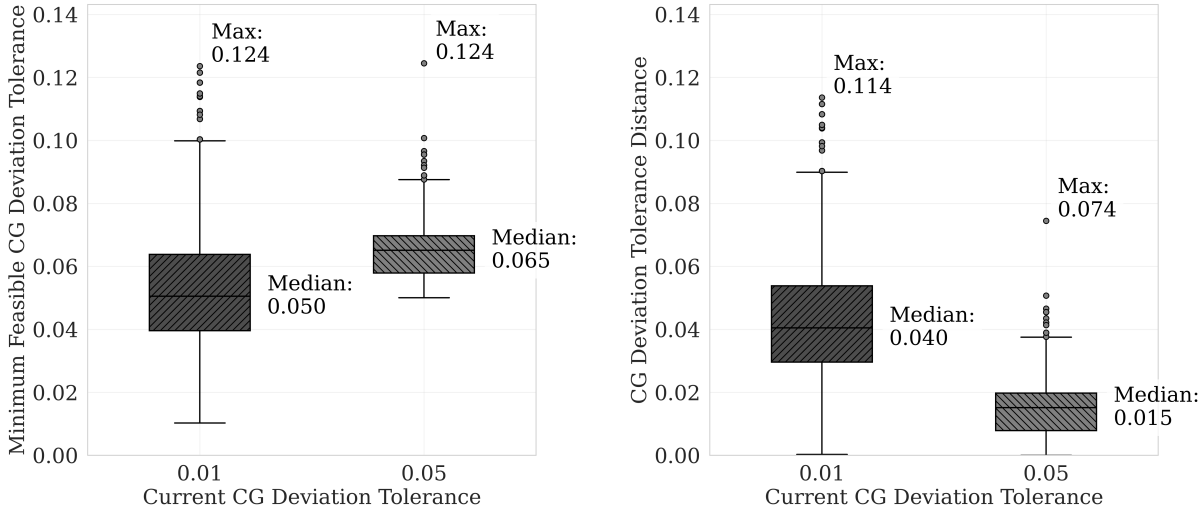

Figure 4: Distribution of CG deviation tolerance requirements for failed solutions. The left panel shows the minimum CG deviation tolerance thresholds at which failed solutions would have achieved load balance. The right panel shows the distance between actual CG deviations and the allowable tolerance thresholds for failed solutions at 1% and 5% CG deviation tolerance levels.

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 grid-based lower bounds developed for this problem structure.

Table 4 presents the lower bound quality for the 10 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 grid-based

bounds provide an independent quality assessment across all instances, enabling comprehensive comparison of bounding approaches.

Table 4: Comparison of lower bound quality for the 10 largest problem instances by item count. Values for Obj, LB, and GB-LB are scaled by 10^5 (e.g., 11.3 represents 1.13e6). % Gap columns show $((\text{Obj} - \text{Bound})/\text{Obj}) \times 100\%$.

Items	**Groups**	**CG Dev**	**Obj**	**LB**	**GB-LB**	**LB Gap**	**GB-LB Gap**
111	26	0.01	118.44	71.72	90.69	39.44%	23.43%
111	26	0.05	112.22	61.37	90.69	45.32%	19.19%
111	26	0.10	110.05	49.91	90.69	54.65%	17.59%
111	26	0.15	109.87	39.19	90.69	64.33%	17.46%
110	22	0.01	125.38	65.84	95.04	47.49%	24.20%
110	22	0.05	117.41	57.50	95.04	51.03%	19.05%
110	22	0.10	112.97	48.74	95.04	56.86%	15.87%
110	22	0.15	110.07	41.50	95.04	62.29%	13.66%
94	11	0.01	75.23	37.09	61.10	50.70%	18.78%
94	11	0.05	74.19	33.10	61.10	55.38%	17.64%
94	11	0.10	74.29	28.17	61.10	62.08%	17.75%
94	11	0.15	73.82	23.35	61.10	68.37%	17.23%
84	26	0.01	48.45	23.56	37.76	51.38%	22.08%
84	26	0.05	46.37	19.58	37.76	57.77%	18.57%
84	26	0.10	45.56	16.27	37.76	64.29%	17.12%
84	26	0.15	44.85	13.12	37.76	70.75%	15.82%
84	21	0.01	62.78	33.43	48.73	46.75%	22.37%
84	21	0.05	60.30	29.41	48.73	51.23%	19.18%
84	21	0.10	58.16	24.39	48.73	58.06%	16.20%
84	21	0.15	57.86	20.11	48.73	65.24%	15.77%
82	17	0.01	57.04	37.88	44.25	33.59%	22.43%
82	17	0.05	54.72	32.38	44.25	40.82%	19.14%
82	17	0.10	54.47	26.42	44.25	51.49%	18.76%
82	17	0.15	54.37	21.58	44.25	60.32%	18.62%
75	9	0.01	40.76	25.72	30.69	36.90%	24.72%
75	9	0.05	38.78	22.87	30.69	41.03%	20.87%
75	9	0.10	37.97	19.38	30.69	48.97%	19.19%
75	9	0.15	37.72	15.63	30.69	58.55%	18.64%
71	5	0.01	48.68	27.33	35.65	43.87%	26.77%
71	5	0.05	46.60	21.97	35.65	52.84%	23.49%
71	5	0.10	46.30	15.14	35.65	67.31%	23.01%
71	5	0.15	46.39	9.34	35.65	79.88%	23.16%
70	17	0.01	22.30	10.72	17.86	51.90%	19.91%
70	17	0.05	22.28	9.10	17.86	59.17%	19.85%
70	17	0.10	22.22	7.28	17.86	67.24%	19.63%
70	17	0.15	22.22	5.92	17.86	73.36%	19.64%
70	14	0.01	42.53	25.62	31.60	39.76%	25.70%
70	14	0.05	40.51	22.52	31.60	44.42%	22.00%
70	14	0.10	38.19	19.58	31.60	48.72%	17.25%
70	14	0.15	37.84	16.69	31.60	55.88%	16.50%

The results demonstrate the critical importance of the grid-based bounds for large-scale instances, serving as the primary quality assessment tool when Single MILP frequently fails to find feasible solutions within time limits. The grid-based bounds remain consistently available across all instances, with their superiority stemming from explicit incorporation of the problem's prioritization structure, accounting for both item-to-access-point distances and item-to-item adjacency requirements 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, while potentially making the problem more tractable by reducing the feasible region. The grid-based gap percentages remain consistent across different CG deviation levels for each instance, providing stable quality assessment regardless of constraint tightness. This consistency enables reliable solution quality evaluation across varying operational requirements, particularly when traditional solver bounds are unavailable.

To complement these deterministic bounds, we employ Wilson's statistical lower bound approach based on extreme value theory (Wilson et al., 2004; Kirschenman et al., 2025a), 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. (2025a), 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 5 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 1.92% to 6.15%, suggesting proximity to theoretical optima despite the challenging multi-objective nature of the problem.

Table 5: 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	4.76	4.88	(4.76, 4.88)	2.46%
84 items	0.05	4.57	4.68	(4.57, 4.67)	2.14%
84 items	0.10	4.42	4.56	(4.42, 4.58)	3.49%
84 items	0.15	4.38	4.54	(4.38, 4.55)	3.74%
82 items	0.01	5.63	5.73	(5.63, 5.74)	1.92%
82 items	0.05	5.36	5.46	(5.36, 5.47)	2.01%
82 items	0.10	5.29	5.43	(5.29, 5.45)	2.94%
82 items	0.15	5.31	5.42	(5.31, 5.45)	2.57%
75 items	0.01	3.93	4.10	(3.93, 4.11)	4.38%
75 items	0.05	3.77	3.91	(3.77, 3.92)	3.83%
75 items	0.10	3.65	3.80	(3.65, 3.83)	4.70%
75 items	0.15	3.66	3.80	(3.66, 3.90)	6.15%

6.2.2. Distribution of Solution Quality

Figure 5 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.

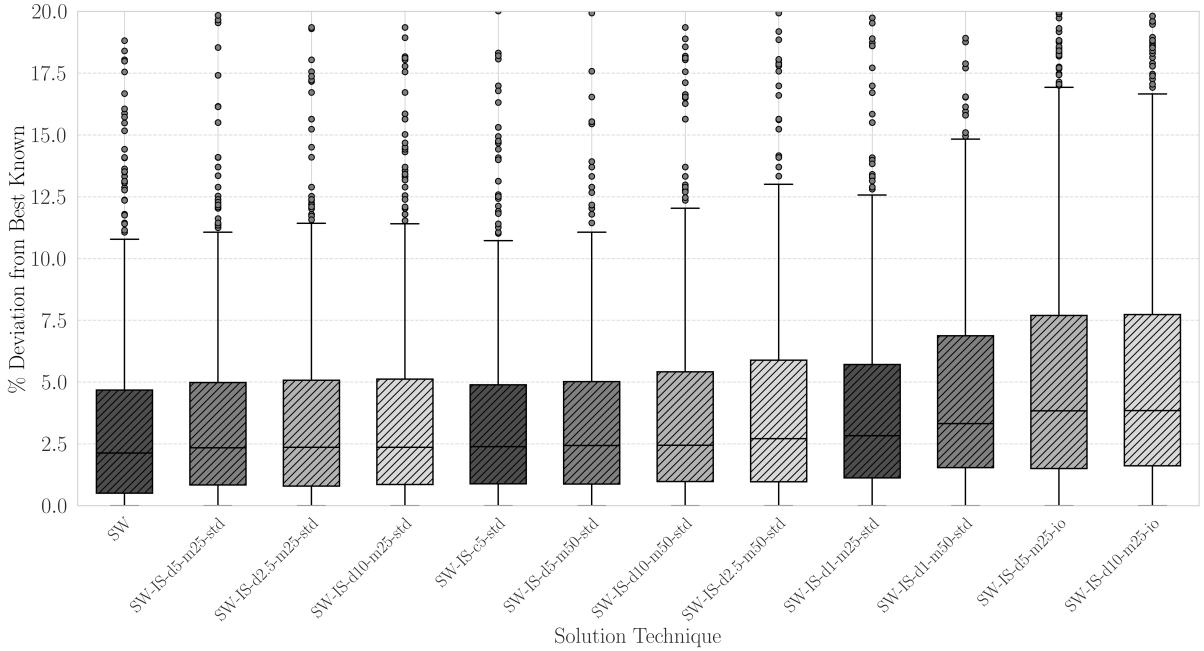

Figure 5: Distribution of solution quality (percentage deviation from best-known objective) for top-performing techniques. Abbreviations: SW = Sliding-Window Iterative (all unfixing techniques); SW-IS = Sliding-Window In-Stride with parameter notation c[X] for constant lambda schedules, d[X] for dynamic 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. 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.

6.3.1. Trade-off Between Success Rate and Solution Quality

Figure 6 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.

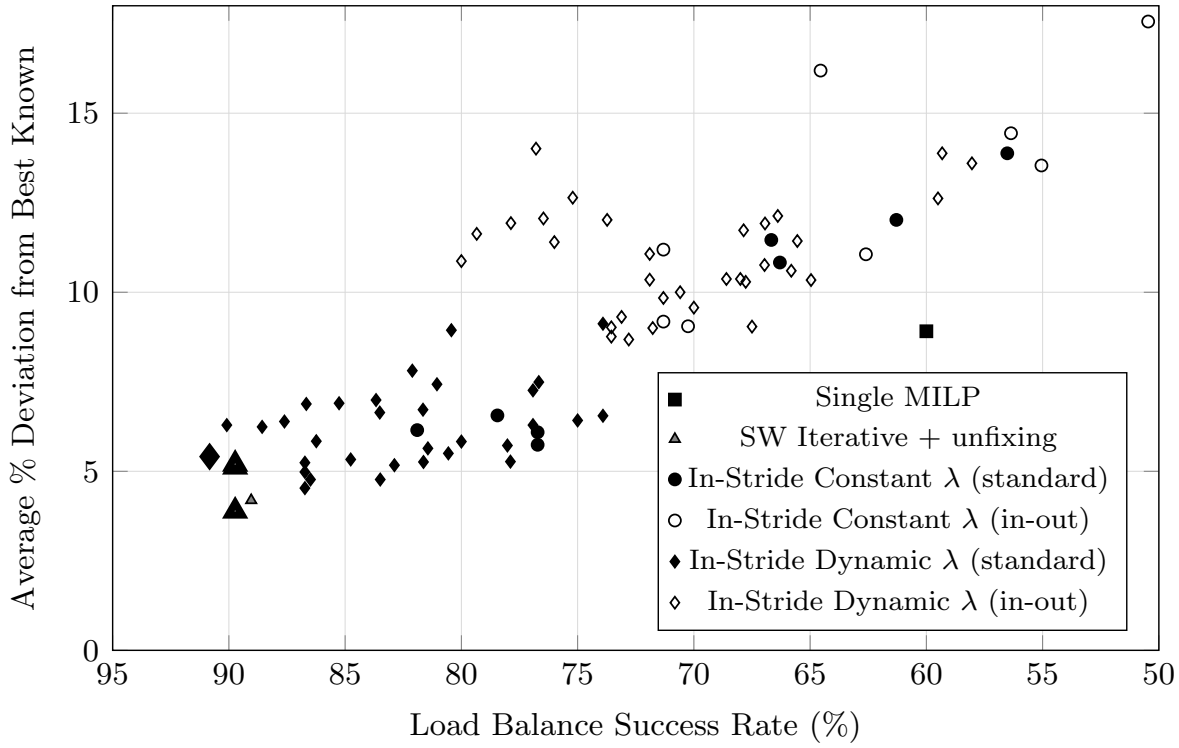

Figure 6: Pareto frontier analysis: Load balancing success rate versus solution quality. Points with thicker borders represent Pareto efficient configurations. Note the horizontal axis is reversed.

Four configurations achieve Pareto efficiency: three SW Iterative variants (with DXR, GCG, and RP unfixing techniques) all achieve 89.7% 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 6.1.1). 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 ($H = 1.73$, $p = 0.422$). The techniques are statistically equivalent in solution quality performance. 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.

The SW Iterative method achieves high success rates (97.5%), solution quality (2.1% median deviation), and computational efficiency (118s average), with post-processing typically converging in a single iteration when using the Greedy CG Impact unfixing technique.

Table 6 provides a comprehensive comparison of all three performance metrics across solution methods.

Table 6: Performance comparison across all solution methods. Avg. Quality Gap calculated only for instances where feasible solutions were found.

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%

The computational analysis demonstrates the SW Iterative approach achieves the highest performance across all three evaluation criteria. The method's performance across success rate, solution quality, and computational efficiency 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.

7. Conclusion

This paper extends the prioritized two-dimensional packing framework to address the specific operational requirements of military combat loading, with particular emphasis on integrating global load balancing constraints. Our approach maintains the flexibility of the prioritization matrix from prior work while incorporating critical feasibility requirements that ensure vessel stability and safety. Through extensive computational experiments on realistic military scenarios, we demonstrate that matheuristic solution methods fundamentally outperform Single MILP optimization approaches across all evaluation criteria: load balancing success rate, solution quality, and computational efficiency. Critically, the Single MILP approach fails to find even a single feasible solution in 40% of test instances within reasonable time limits, while our matheuristic configurations achieve substantially higher success rates—a stark illustration that this represents a fundamental difference in problem-solving capability, not merely solution quality or speed.

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. 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 SW Iterative method family dominates the Pareto frontier, with multiple configurations achieving excellent trade-offs between load balance success rate and solution quality. Among the Pareto efficient points, the SW In-Stride configuration with dynamic lambda schedules ($\gamma = 10.0$, $\lambda_{\max} = 0.25$, standard penalties with GCG post-processing) achieves the highest load balance success rate at 90.8% with 5.4% deviation from best-known values, while SW Iterative with DXR unfixing provides the best solution quality (3.9% deviation) at 89.7% load balance success rate. For practitioners prioritizing reliability above all else, the carefully tuned SW In-Stride approach with dynamic lambda schedules offers marginally better load balance success rates; for those emphasizing solution quality, SW Iterative with DXR provides superior performance. These dominant configurations demonstrate superior computational efficiency compared to the Single MILP approach, with the SW Iterative method averaging 121.8 seconds per instance compared to 340.0 seconds for the Single MILP approach, generating better solutions in less than half the time.

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. 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.

Acknowledgments

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

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.

References

Akyüz, M.H., Lee, C.Y., 2014. A mathematical formulation and efficient heuristics for the dynamic container relocation problem. Naval Research Logistics 61, 101–118. doi:10.1002/nav.21569.

Bazaraa, M.S., Sherali, H.D., 1980. Benders' partitioning scheme applied to a new formulation of the quadratic assignment problem. Naval Research Logistics Quarterly 27, 29–41. doi:10.1002/nav.3800270104.

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

Brown, G.G., DeGrange, W.C., Price, W.L., Rowe, A.A., 2017. Scheduling combat logistics force replenishments at sea for the US Navy. Naval Research Logistics 64, 677–693. doi:10.1002/nav.21780.

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

Chazelle, 1983. The bottomn-left bin-packing heuristic: An efficient implementation. IEEE Transactions on Computers 100, 697–707. doi:10.1109/TC.1983.1676307.

Commander in Chief, U.S. Pacific Fleet, 1950. 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. Report No. 14ND-CINCPACFLT-P-35. Covers the period 25 June to 15 November 1950. Archival document sourced from the National Archives and Records Administration (NARA), College Park.

Davies, A.P., Bischoff, E.E., 1999. Weight distribution considerations in container loading. European Journal of Operational Research 114, 509–527. doi:10.1016/S0377-2217(98)00139-8.

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

DorMohammadi, S., Rais-Rohani, M., 2013. Exponential penalty function formulation for multilevel optimization using the analytical target cascading framework. Structural and Multidisciplinary Optimization 47, 599–612. doi:10.1007/s00158-012-0861-x.

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

Eley, M., 2002. Solving container loading problems by block arrangement. European Journal of Operational Research 141, 393–409. doi:10.1016/S0377-2217(02)00133-9.

Fischetti, M., Fischetti, M., 2018. Matheuristics, in: Handbook of Heuristics. Springer, pp. 121–153. doi:10.1007/978-3-319-07124-4_14.

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

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

Headquarters, Department of the Army, 2015. Army Watercraft Operations. U.S. Army Training and Doctrine Command. URL: https://armypubs.army.mil/epubs/DR_pubs/DR_a/pdf/web/atp4_15.pdf. accessed: August 12, 2025. Official source for Army watercraft specifications, including the General Frank S. Besson-class and Runnymede-class watercraft.

Headquarters, Department of the Army, 2019. Department of the Army Pamphlet 71–32 Force Development and Documentation Consolidated Procedures. Headquarters, Department of the Army. URL: https://armypubs.army.mil/epubs/DR_pubs/DR_a/ARN44160-PAM_71-32-002-WEB-4.pdf.

Heidelberg, K.R., Parnell, G.S., Ames IV, J.E., 1998. Automated air load planning. Naval Research Logistics 45, 751–768. doi:10.1002/(SICI)1520-6750(199812)45:8<751::AID-NAV1>3.0.CO;2-3.

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

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

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

Kirschenman, W.K., Heese, H.S., Kay, M.G., King, R.E., McConnell, B.M., 2025b. Military maritime load planning instances for prioritized two dimensional orthogonal packing. Dryad Repository. Dataset, https://doi.org/10.5061/dryad.vt4b8gv5z.

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

Kline, A.G., Ahner, D.K., Lunday, B.J., 2020. A heuristic and metaheuristic approach to the static weapon target assignment problem. Journal of Global Optimization 78, 791–812. doi:10.1007/s10898-020-00938-4.

Klug, J., 2023. Establishing the realm of the possible: Logistics and military strategy. Military Strategy Magazine 9, 28–34. URL: https://www.militarystrategymagazine.com/article/establishing-the-realm-of-the-possible-logistics-and-military-strategy/.

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

Lewis, J., Parker, R., 1982. On a generalized bin-packing problem. Naval Research Logistics Quarterly 29, 119–145. doi:10.1002/nav.3800290112.

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

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

Lurkin, V., Schyns, M., 2015. The airline container loading problem with pickup and delivery. European Journal of Operational Research 244, 955–965. doi:10.1016/j.ejor.2015.02.027.

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

Moon, I., Nguyen, T.V.L., 2014. Container packing problem with balance constraints. OR Spectrum 36, 837–878. doi:10.1007/s00291-013-0356-1.

Patsiatzis, D., Knight, G., Papageorgiou, L., 2004. An MILP approach to safe process plant layout. Chemical Engineering Research and Design 82, 579–586. doi:10.1205/026387604323142612.

Pierce, J.F., Crowston, W., 1971. Tree-search algorithms for quadratic assignment problems. Naval Research Logistics Quarterly 18, 1–36. doi:10.1002/nav.3800180102.

Pohl, K.J., Morosoff, P., 2011. 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. pp. 55–? ISBN: 978-1-897233-77-1, Paper ID: IS11-Kym-Peter-Paper.

Ramos, A.G., Silva, E., Oliveira, J.F., 2018. A new load balance methodology for container loading problem in road transportation. European Journal of Operational Research 266, 1140–1152. doi:10.1016/j.ejor.2017.10.050.

Tapestry Solutions, Inc., 2024. JECD Codes: Definitions and Descriptions for ICODES Cargo Category and Shipping Configuration Codes. Internal documentation PDF for ICODES JECD.

Townsend, C.E., 2017. A comprehensive view of deployment readiness challenges. Army Sustainment URL: https://www.army.mil/article/193244/a_comprehensive_view_of_deployment_readiness_challenges. digital Exclusive.

Trivella, A., Pisinger, D., 2016. The load-balanced multi-dimensional bin-packing problem. Computers & Operations Research 74, 152–164. doi:10.1016/j.cor.2016.04.020.

U.S. Navy, 2025a. Fact file: Amphibious assault ships - LHD/LHA(R). https://www.navy.mil/Resources/Fact-Files/Display-FactFiles/Article/2169814/amphibious-assault-ships-lhdlhar/. Accessed: Aug 12, 2025. Source for Wasp-class well deck. Also confirms America-class (Flight 1) has a well deck for two LCACs, from which dimensions are estimated.

U.S. Navy, 2025b. Fact file: Dock landing ship - LSD. https://www.navy.mil/Resources/Fact-Files/Display-FactFiles/Article/2169901/dock-landing-ship-lsd/. Accessed: Aug 12, 2025. Official source for Whidbey Island-class and Harpers Ferry-class well deck dimensions.

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

Wilson, A.D., King, R.E., Wilson, J.R., 2004. Case study on statistically estimating minimum makespan for flow line scheduling problems. European Journal of Operational Research 155, 439–454. doi:10.1016/S0377-2217(02)00910-4.

X Corps Headquarters, 1951. 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. Pagination approximately 40, relative to start of Part II. Year inferred from related map dated 8 June 1951. Approved by LTG E. M. Almond and COL J. S. Guthrie. Archival document accessed at the MacArthur Memorial Museum.

Appendix A. Modeling and Validation of Additional Operational Constraints

While the primary focus of this research centers on developing effective solution methods for load balancing constraints—the most computationally challenging aspect of the prioritized packing problem—our formulation can readily accommodate other operational constraints. This appendix presents the mathematical formulations for obstacle avoidance and material separation constraints, then provides proof-of-concept validation that our model successfully enforces these requirements when activated.

Vessel decks may contain fixed obstacles such as structural pillars. We model these by partitioning the set of items I into a set of cargo items to be placed, I^E, and a set of fixed obstacle items, I^F. For each obstacle $f \in I^F$, its coordinates are known parameters $(x_f^*, y_f^*, x_f^{r*}, y_f^{r*})$. We add the following constraints to fix their positions:

$$x_f = x_f^*, \quad y_f = y_f^*, \quad x_f^r = x_f^{r*}, \quad y_f^r = y_f^{r*} \qquad \forall f \in I^F \tag{A.1}$$

The standard non-overlap constraints (4)–(10), which apply to all pairs of items in $I = I^E \cup I^F$, then naturally prevent any cargo item from overlapping with these fixed obstacle regions. Because the coordinates of these obstacles are fixed parameters, the corresponding orientation variables $t_{f,c,d}$ are implicitly determined by the geometric dimensions and can be pre-set or handled

automatically during solver preprocessing. This is a standard technique in the packing and FLP literature (Lodi et al., 2002; Meller et al., 1998).

To ensure safety, certain pairs of items may require a minimum separation distance. Let $H \subseteq I^E \times I^E$ be the set of pairs requiring a minimum edge-to-edge separation of d^H. We can integrate this requirement by modifying the geometric non-overlap constraints for those pairs. Let h_{ik} be a binary parameter that is 1 if $(i,k) \in H$ and 0 otherwise. The non-overlap constraint (7) is modified as

$$x_k^r \leq x_i - h_{ik} \cdot d^H + (1 - a_{ik}^x)L \qquad \forall i,k \in I^E, i < k, \tag{A.2}$$

and similarly for the other three directional separation constraints. If separation is required ($h_{ik} = 1$), an additional gap of d^H is enforced between the items' edges. This approach embeds separation rules directly into the non-overlap logic, similar to methods used to model safety distances in process plant layout design (Patsiatzis et al., 2004).

The main experimental campaign excluded these additional constraints to maintain focus on the load balancing challenge, which represents the core computational bottleneck. However, to demonstrate the practical applicability of our approach for real-world military logistics scenarios, we conducted limited validation experiments on small instances specifically designed to visually demonstrate constraint enforcement. All validation solutions use the Single MILP method with time limits as described in Section 5.3.

Appendix A.1. Obstacle Avoidance Validation

To validate our obstacle avoidance implementation, we created two special test instances based on existing problems from our dataset: instance_24 (15 cargo items, 3 priority groups) and instance_47 (18 cargo items, 6 priority groups). Each test instance was augmented with two fixed obstacles strategically positioned to challenge the packing algorithm while maintaining visual clarity. The obstacles were designed to occupy less than 10% of the total deck area to ensure feasible solutions exist.

For instance_24, obstacles were placed as: (1) a wall protrusion from the left edge (120×48 units), and (2) a central obstacle in the upper deck region (200×150 units). Similar obstacle configurations were applied to instance_47. Both instances used the Runnymede Class LCU vessel configuration with load balancing tolerance $\delta = 0.15$.

Figure A.7 demonstrates the successful enforcement of obstacle avoidance constraints. The visualization shows cargo placement solutions (gray rectangles) that successfully avoid fixed obstacles (black rectangles) while maintaining load balancing requirements, contrasted with baseline solutions that do not consider obstacle constraints.

Appendix A.2. Material Separation Validation

For material separation validation, we modified the same test instances to include separation requirements. Three cargo items per instance (items 2, 6, and 11) were designated as requiring a minimum 60-inch edge-to-edge separation distance—representative of typical safety distances for hazardous materials in military logistics operations. This separation distance was integrated directly into the non-overlap constraints as formulated above.

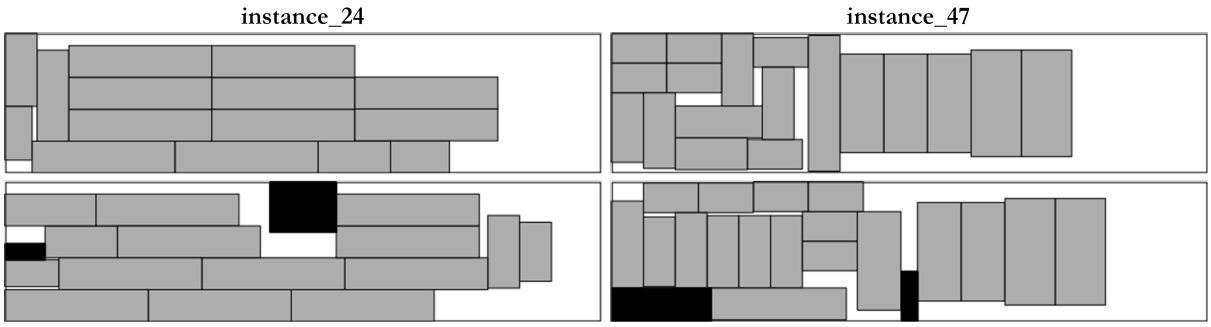

Figure A.7: Obstacle avoidance validation for instances 24 (left panel) and 47 (right panel). Top row shows baseline Single MILP solutions without obstacle constraints, with all cargo items in gray fill. Bottom row shows solutions incorporating obstacle avoidance constraints, where cargo items (gray fill) successfully avoid fixed obstacles (black fill) while maintaining load balancing requirements.

The constraint enforcement strategy ensures that any pair of designated items maintains the required separation, even when only one item in the pair requires separation (a more conservative approach than strictly pair-based separation). This design choice enhances safety while remaining computationally tractable within the existing constraint framework.

Figure A.8 demonstrates successful enforcement of material separation constraints. The visualization shows that designated cargo items requiring separation (white rectangles) maintain the required 60-inch minimum edge-to-edge distances from each other and from regular cargo items (gray rectangles) while the solution process minimizes prioritization objectives and satisfies load balancing requirements, contrasted with baseline solutions that do not enforce separation constraints.

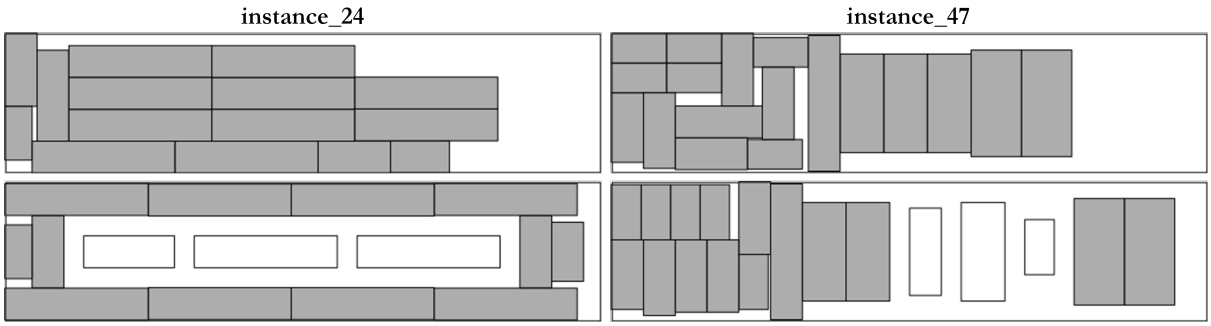

Figure A.8: Material separation validation for instances 24 (left panel) and 47 (right panel). Top row shows baseline Single MILP solutions without separation constraints, with all cargo items in gray fill. Bottom row shows solutions incorporating material separation constraints, where items requiring 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.

Appendix A.3. Implementation Notes

These proof-of-concept validations confirm that our mathematical formulation correctly handles additional operational constraints without requiring fundamental changes to the solution architecture. The obstacle avoidance constraints integrate seamlessly through the existing non-overlap logic by treating obstacles as fixed items with predetermined coordinates. Mate-

rial separation constraints are embedded directly into the geometric non-overlap constraints, maintaining the model's computational structure while enforcing safety requirements.

Both constraint types can be activated independently or simultaneously, and the matheuristic-based solution methods described in Section 4 remain applicable. The small instance sizes used for validation (15–18 items) were selected specifically to enable clear visual verification of constraint enforcement rather than to demonstrate computational scalability, which was the focus of the main experimental campaign.