

Article

Optimizing the Train Unloading Operations in a Maritime Container Terminal

Daniela Ambrosino ^{1,2,3,*} , Veronica Asta ^{3,4}, Khawar Bashir ^{2,4,*}  and Costanza Chiesa ⁴¹ Department of Economics and Business Studies, University of Genoa, 16126 Genoa, Italy² Research Center on Logistics, Transport and Infrastructures (CIELI), University of Genoa, 16126 Genoa, Italy³ OPTIMEasy–University of Genoa Spin Off, 16121 Genoa, Italy; veronica.asta@optimeasyway.com⁴ Circle Group SpA, Piazza Borgo Pila 40 (A/46), 16129 Genoa, Italy; chiesa@circletouch.eu

* Correspondence: daniela.ambrosino@unige.it (D.A.); khawar.bashir@edu.unige.it (K.B.)

Abstract

Background: Rail operations form a critical interface between maritime container terminals and their hinterlands, yet their operational planning remains underexplored; this paper introduces the Train Unloading Planning Problem (TUPP), which jointly determines the container unloading sequence, gantry crane movements, reach stacker routing, and the assignment of containers to compatible export yard bay-locations. **Methods:** We propose a mixed-integer linear programming (MILP) formulation that captures the tightly coupled interactions among these decisions, strengthened by valid inequalities that reduce CPU time; because exact optimization becomes intractable at operationally realistic sizes, we develop an Iterated Local Search (ILS) heuristic that refines an MILP warm-start using two complementary neighborhood operators, each embedding a greedy reach stacker re-routing procedure, together with a controlled perturbation mechanism. **Results:** On 16 groups of benchmark instances reflecting Italian terminal operations, the MILP proves optimality only for the smallest instances, with optimality gaps high on medium sized instances. Averaged across all groups, the ILS improves solution quality by 16.62% over the time-limited MILP while requiring about 360 s of total solution time, roughly a ten-fold reduction, with gains reaching 36.56% on the largest instances. **Conclusions:** The framework provides an effective decision support tool for operational planning in non-automated maritime container terminals.

Keywords: maritime container terminal; train unloading operations; mathematical programming formulation; heuristic approach.



Academic Editors: Amir Alizadeh-Masoodian and Jay Golden

Received: 12 May 2026

Revised: 11 July 2026

Accepted: 22 July 2026

Published: 28 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Maritime container terminals are key infrastructures in the global logistics network, facilitating the trans-shipment of goods between sea and inland transport modes. Recent evidence from the Review of Maritime Transport 2025 indicates that global maritime trade is increasingly shaped by volatility, re-routed flows, and heightened uncertainty. Geopolitical tensions and shifting trade policies are contributing to longer voyage distances and increased ton-mile demand. According to UNCTAD Review of Maritime Transport 2025, seaborne trade growth is projected at 0.5% in 2025, while containerized trade is forecasted to increase by 1.4%. Over the medium term (2026–2030), annual growth rates are projected at approximately 2% for total trade and 2.3% for containerized volumes [1]. The expansion of sea–rail intermodal systems represents a key strategic response to this

growth [2], yet the operational planning of rail terminal equipment remains a critical and underexplored bottleneck. The increasing volatility and complexity of maritime trade patterns reinforce the need for efficient and well-coordinated handling processes across terminal functions.

Terminal performance depends on the synchronization of multiple interdependent operations carried out across the seaside, yard, and landside areas. Within this integrated system, rail processes play a strategic role in linking maritime terminals with their hinterlands, supporting capacity extension and congestion mitigation. Their effective management requires careful coordination of physical layout, handling equipment, and resource utilization.

The performance of rail processes is strongly influenced by equipment scheduling and the synchronization of container movements. Port storage areas serve as temporary buffers during modal transitions [3], and inefficient yard space allocation due to extended container dwell times can significantly reduce available storage capacity and disrupt terminal flows [4]. Recent systematic reviews emphasize the importance of synchronizing maritime and inland operations through integrated scheduling [5], and highlight the broader role of operations research in supporting such coordination within container terminals [6]. From an operational perspective, inefficient coordination among these resources can lead to excessive equipment travel, increased handling times, and reduced terminal throughput, directly affecting service reliability.

Several studies demonstrate that improved coordination of container handling resources and inland transfer operations can substantially reduce delays and enhance the efficiency of rail–terminal interfaces [7,8]. Ref. [9] analyzes landside train operations and shows how coordinated scheduling of gantry cranes and straddle carriers can significantly influence terminal productivity and train delays. Among rail-related terminal operations, train unloading and loading present distinct challenges due to their direct impact on equipment utilization and yard space allocation. Recent studies underline the need for coordinated scheduling among cranes and horizontal transport systems to avoid interference and improve handling efficiency, with applications in both automated terminals using AGVs and conventional port operations using internal trucks [10,11], while the necessity of complying the rail processes with strict arrival and departure time windows is pointed out in [12]. The growing relevance of these coordination challenges is further reinforced by the rising share of rail in inland container transport and the continued growth of containerised maritime volumes, which intensify the pressure on rail-side handling at sea–rail terminals [13]. This pressure is compounded by the ongoing extension of on-dock rail access into port terminals: bringing trains directly to the terminal shortens horizontal transport distances but concentrates additional handling demand on rail-side equipment, further tightening the coupling between train operations and terminal resources and reinforcing the need for efficient rail-side planning [14].

Despite these contributions, the existing literature on non-automated maritime container terminals often addresses loading and unloading sequence decisions, equipment scheduling, and yard allocation as separate problems or relies on simplifying assumptions that limit applicability. In particular, an integrated treatment of sequence definition, crane movements, reach stacker trajectories, and yard block compatibility within a single operational framework is still largely missing. This fragmentation hampers the ability to accurately capture the operational interdependencies that characterize real-world rail operations.

This study addresses this gap by focusing on the train unloading process, which presents distinct operational challenges due to the need for coordinated yard space allocation and strict compliance with departure schedules.

Specifically, we address the Train Unloading Planning Problem (TUPP), a problem faced daily in some maritime container terminals located in Italy, operating as follows: trains arriving at the terminal are transferred to rail tracks, the terminal zone where they are unloaded and loaded by terminal handling equipment. Containers unloaded from the trains are generally loaded onto specific vessels expected at the terminal berths. These containers are therefore transferred to the export yard and stored in specific locations awaiting shipment. The export yard is organized to guarantee efficient vessel loading operations, and specific storage strategies are applied. Obviously, export yard and rail system management strongly depend on the terminal layout and its handling equipment. We are facing the TUPP in terminals with an European layout, with an export yard operated by reach stackers (RSs) and adopting a consignment storage strategy, and a rail park where gantry cranes (GCs) are used to operate trains. TUPP aims to jointly determine the unloading sequence of containers and their assignment to export yard bay-locations, while explicitly accounting for coordinated crane and reach stacker operations, containers' attributes, and yard storage constraints.

The main contributions of this work are summarized as follows:

- The development of a novel mixed-integer linear programming model (MILP) for the Train Unloading Planning Problem that jointly integrates unloading sequence decisions, crane movements, reach stacker operations, and yard block compatibility constraints, while minimizing the weighted sum of three terms that represent the movement of the handling equipment.
- The design of a structured heuristic optimization framework, combining deterministic local search and controlled perturbation, to overcome the scalability limitations of exact optimization.
- An extensive computational analysis evaluating the effects of weight configurations used for the three terms composing the objective function of the MILP model and heuristic strategies on solution quality and computational efficiency across a range of instance sizes.

The remainder of the paper is organized as follows. Section 2 reviews the relevant literature on train unloading and container terminal operations. Section 3 describes the problem setting and key assumptions. Section 4 presents the mixed-integer linear programming formulation and the heuristic solution framework proposed to solve the problem under investigation. Section 5 reports and discusses the computational results obtained with the exact and heuristic solution approaches. Finally, Section 6 concludes the study and outlines directions for future research.

2. Literature Review

Despite the recognized importance of rail operations for intermodal efficiency, hinterland connectivity, and port capacity extension, research on train loading and unloading processes within maritime container terminals remains relatively limited. Rail operations are widely acknowledged as a key mechanism for congestion mitigation and spatial decoupling between ports and hinterlands, with dry port developments representing a strategic response to capacity constraints and hinterland access challenges [15], yet their operational modelling has received significantly less attention than quay-side and yard-side processes.

At a strategic and tactical level, several studies emphasize the relevance of the rail–sea interface while adopting aggregated or flow-based perspectives. Early contributions highlight the lack of integrated models capable of jointly capturing shunting activities, yard storage decisions, and rail operations within maritime container terminals [16]. Discrete-time control approaches have been proposed to coordinate import rail cycles and yard storage decisions with train dispatching in order to reduce transfer delays [7]; however,

these models focus exclusively on import flows and do not represent equipment-level interactions or container handling sequences. Similarly, stochastic dynamic programming formulations have been developed for prestaging and flow-rate optimization under uncertainty, accounting for operational features such as discharge–load precedence and late train arrivals, but without explicitly modelling individual handling resources or the sequencing of container movements [12].

More recent studies investigate the coordinated operation of multiple handling systems in sea–rail contexts, particularly within automated terminal environments. Integrated bi-objective optimization models have been developed to jointly minimize completion time and energy consumption in automated sea–rail terminals, explicitly modelling the interactions among rail-mounted gantry cranes, yard cranes, and automated guided vehicles while accounting for multiple container flows between ships, trains, and yard storage areas [10]. These models address the coupling between railway and terminal operations by defining distinct container flow patterns (train–ship, train–yard, and yard–train) and corresponding vehicle routing strategies to support simultaneous loading and unloading operations.

A closely related stream focuses on coordinated rail-side scheduling under strict temporal constraints. The joint scheduling of yard cranes, external trucks, and rail-mounted gantry cranes in sea–rail systems operating under port–railway separation mode has been examined, explicitly incorporating train arrival and departure windows as well as vessel loading deadlines for export containers [2]. The collaborative scheduling of automated rail-mounted gantry cranes and automated guided vehicles is addressed using a two-layer scheduling rule combined with a self-adaptive chaotic genetic algorithm, demonstrating that optimal equipment ratios vary with vessel delay scenarios and container type distributions. Flexible yard crane scheduling under mixed rail and truck flows in shared storage yards has been investigated, accounting for train time windows, uncertain external truck arrivals, and crane interference constraints [17]. A two-stage matheuristic framework that integrates task reassignment and sequential decision-making phases to minimize expected makespan and truck waiting time across multiple uncertain scenarios is proposed. The most recent contributions to this stream jointly schedule multiple rail-side handling resources in on-dock railway operation areas. One formulates a binary integer program on a space time state network, intractable at realistic scale and solved via reformulation and a Lagrangian relaxation-based heuristic [13], while another jointly optimizes rail-mounted gantry crane deployment, task assignment, and scheduling under train arrival and departure timetables, strengthening the formulation with valid inequalities and solving it through a logic-based Benders decomposition [14]. These studies reinforce the case for problem specific heuristics while sharpening the gap addressed here: they operate at the level of crane scheduling, deployment, and timetabling with container or internal truck transfer, fixing container destinations in advance, so that two central elements of TUPP that are: (i) yard block assignment under size, weight-class, and destination compatibility, (ii) reach stacker routing among bay-locations, are not represented.

Collectively, these studies [2,10,13,17] highlight the tightly coupled temporal dependencies induced by rail operations and the resulting scalability limits of exact optimization approaches, with commercial solvers failing to produce quality solutions within reasonable time limits, thereby necessitating the development of specific heuristics, metaheuristics, and hybrid decomposition strategies capable of handling realistic instance sizes while maintaining solution quality.

In parallel, research on non-automated port operations has examined the synchronization of crane scheduling and internal truck routing under safety and collision avoidance constraints, demonstrating that coordinated task allocation and speed control strategies can

significantly reduce handling delays and improve equipment utilization [11]. While these approaches demonstrate the operational benefits of integrated multi-equipment scheduling, they remain largely confined to either fully automated terminals with AGV-based transport systems or focus exclusively on seaside-to-yard transfer operations. Notably, neither stream addresses container-unloading sequence decisions or yard block assignment rules that characterize rail-side operations in maritime container terminals. Moreover, operations involving reach stackers, which characterize the problem analyzed in this paper, are also excluded.

In a related inland rail terminal context, the sequencing of rail-mounted gantry crane operations has been examined under hybrid handling modes combining single and dual-cycle movements [18]. A mathematical formulation is proposed to determine the handling sequence of container moves within a fixed crane working area, with the objective of minimizing crane idle travel time. An ant colony optimization heuristic is employed to generate near-optimal operation sequences, and computational experiments report significant reductions in idle movements compared to rule-based strategies. However, the considered setting differs substantially from maritime rail-side export operations. In particular, containers are assumed to be handled entirely within predefined rail terminal areas, without assignment to export yard blocks, and handling locations are fixed and known in advance. Moreover, the model focuses exclusively on the movement sequencing of a single crane type and does not account for interactions with other handling resources or storage capacity constraints. As a result, while the study provides useful insights into crane movement sequencing in inland rail yards, its assumptions limit direct applicability to similar unloading planning problems in maritime container terminals.

Further studies have explored coordinated equipment scheduling in rail-side handling areas. The integrated scheduling of rail-mounted gantry cranes, internal trucks, and reach stackers under simultaneous loading and unloading operations has been investigated, formulating a mixed-integer programming model that minimizes both makespan and total equipment waiting time in [19]. A genetic algorithm incorporating a first-available-machine rule is developed to solve the problem, wherein rail-mounted gantry operation sequences are encoded chromosomally while internal trucks and reach stackers are dynamically assigned based on earliest availability. Their numerical experiments compare alternative equipment configurations and quantify unproductive times, particularly waiting and empty travel, highlighting how additional RSs or internal trucks affect overall system performance. Although the setting differs from maritime export yards due to predefined container destinations, the absence of yard block assignment decisions, simplified assumptions on container types and in-block RS movements, the study provides relevant insights into RS behavior and multi-equipment coordination and confirms that reach stacker-related decision problems in intermodal terminals remain relatively underexplored.

Closer to deep-sea terminal environments, some studies address train loading and unloading operations. A mixed-integer programming formulation combined with a simulated annealing heuristic is proposed in [9] to coordinate gantry cranes and straddle carriers for loading and unloading trains under given arrival and departure times, with the objective of minimizing total train delay. The model provides a detailed representation of crane and straddle carrier interactions and evaluates the impact of train characteristics and fleet size on system performance. However, container storage decisions are assumed to be predefined, and reach stacker operations as well as yard block assignment choices are not considered.

In fully automated railway yards embedded within container terminals, task allocation and operation modes for automated rail-mounted gantry cranes have been investigated under different loading and unloading strategies in [20]. Multiple mixed-integer linear

programming models are formulated to compare full loading–unloading and synchronous loading–unloading modes under pre-assigned and non-pre-assigned task distributions, with the objective of minimizing task completion time. While the results highlight the efficiency gains of synchronous operations in automated settings, the analysis is restricted to ARMG-based systems and predefined railway yard layouts, and does not account for human-operated equipment, reach stackers, or storage-block compatibility decisions that characterize the problem analyzed in the present paper.

Finally, train unloading operations in an inland railway terminal where reach stackers transfer containers from wagons to storage spans have been addressed in [21]. The authors formulate the problem as the minimization of unloading makespan and propose three alternative optimization approaches. The first follows a sequential strategy in which containers are initially assigned to storage spans and subsequently scheduled on parallel reach stackers. The second adopts an integrated formulation that simultaneously decides storage assignment and reach stacker task sequencing, while the third introduces a discrete-time model capable of explicitly tracking container handling times. Computational experiments on small and pseudo-real instances show that the integrated formulation yields improved makespan performance compared to the sequential approach. The study considers a stand-alone inland railway platform with fixed handling areas, homogeneous containers, and unloading operations performed exclusively by reach stackers, without interaction with other handling systems. As a result, while the work provides valuable insights into reach stacker-based unloading and storage assignment in inland railway terminals, its assumptions limit direct applicability to integrated unloading planning problems in maritime container terminals.

Overall, the literature highlights the growing importance of rail operations within container terminals, while revealing a lack of integrated operational models capable of jointly capturing unloading sequence decisions, coordinated crane and reach stacker movements, and yard block assignment under realistic terminal constraints. Existing studies either adopt aggregated planning perspectives, focus on fully automated environments, integrate multiple rail-side equipment types only under predefined container destinations and truck-based transfer without yard block assignment or reach stacker routing, or isolate individual handling systems, thereby limiting their applicability. Despite this growing relevance, no existing work jointly captures unloading sequence decisions, coordinated gantry crane and reach stacker movements, and yard block assignment within a single operational model for conventional maritime container terminals.

This study therefore addresses the TUPP, presenting an integrated optimization framework that jointly captures all these decisions within a unified operational model. The next section defines the problem in more detail, describing the operational setting and assumptions.

3. Problem Description

In this section, we introduce, in detail, TUPP, a problem presenting a complex planning challenge centered on assigning each container from the train to a suitable yard block in the export area, while defining the best unloading sequence. Beyond satisfying compatibility and capacity constraints for yard blocks, the efficiency of the unloading process depends on minimizing unnecessary equipment movements. Thus, two main operational areas in maritime container terminals are involved:

- The rail park (inside the land side area): A zone with rail tracks where rail-mounted gantry cranes (GCs) operate along fixed tracks above the trains to lift containers from the wagons. The gantry cranes are equipped with an adjustable spreader capable of

handling both 20' and 40' containers. After a container is picked up by a gantry crane, it is placed on a tow tag that transfers it to a specific position in the yard.

- The export yard served by reach stackers and organized according to the storage strategy defined by the terminal, which typically divides the yard into several zones based on the expected vessels. Within each vessel area, there are blocks composed of bay-locations (mini-blocks), characterized by different capacities ranging from 12 to 20 containers, depending on the number and height of stacks composing each mini-block. Each bay-location stores homogeneous containers in accordance with the rules defined by the terminal: containers stored in a bay-location share the same port of destination, size, and weight class. For more details on the yard storage strategy and export yard management, see [22]. RSs move in the yard among blocks and store containers in bay-locations. In particular, an RS waits at a specific bay-location for a tow tag carrying a container to be stored in a designated cell/slot.

Figure 1 illustrates the container terminal layout with the different operational areas: the land-side with the rail park, the gate, the internal storage areas (import and export yard) and the seaside area. The highlighted area (dashed box) represents the study focus, where train unloading operations, gantry crane movements, reach stacker routing, and container allocation and storage must be coordinated (the legend identifies the terminal elements). The tow tags act as intermediate transfer devices that move containers from the gantry crane to the reach stackers along a fixed path. They are not included in the optimization problem here analysed because they are available in sufficient number to serve the unloading flow without ever becoming a bottleneck: a container picked up by the crane is always handed over without waiting for a tow tag, so their presence does not constrain the unloading sequence, the yard block assignment, or the reach stacker routing. As they introduce neither a binding capacity limit nor an additional scheduling decision, the tow tags can be safely abstracted from the model without affecting the optimized decisions.

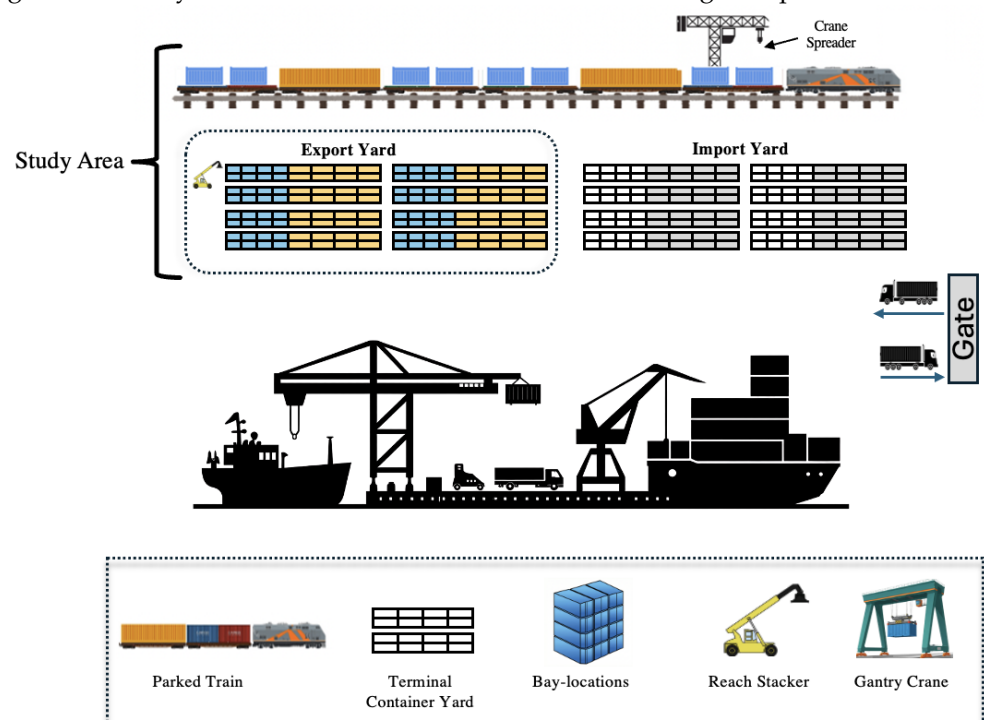


Figure 1. Container terminal layout with rail terminal area highlighted as the study focus.

Let us introduce the problem under investigation in more detail. Given a set of containers to unload (C), characterized by a size (λ), a weight (ω), a class of weight (ϕ),

a destination (d) and a position on a train (θ), and given an export yard comprising a set of bay-locations (B), each one characterized by a maximum capacity (q) and a residual capacity depending on the containers already stored, in addition to an assigned size (Δ), a weight class (k), and a port of destination (σ) of the containers that can be stored on it, the problem consists of determining the sequence of containers to unload while defining, for each container, the bay-locations and where to store it, as well as the RS that will store it, in such a way to minimize the following:

- The GC distance traveled and the number of spreader changes during the container unloading;
- The RS distance traveled for storing in the yard the sequence of containers received.

In the present work, the problem concerns the definition of an unloading plan for a single train loaded with 20' and 40' containers, operated by a single GC, with the aim of minimizing crane movements along the track, changes in spreader length, and the distance traveled by reach stackers in the yard. Moreover, the containers on the train have to be loaded on the same vessel (thus, only a specific area devoted to store containers for that vessel is considered). The following additional assumptions are adopted:

- The train is parked parallel to the gantry crane;
- Unloading begins from the first wagon onward, as the crane is initially positioned at the start of the train;
- The initial configuration of the export yard and the positions of RSs in the yard are known;
- No other container flows occur and the yard configuration remains stationary throughout the planning horizon;
- No overlapping rail operations are performed;
- No equipment breakdowns or operational disruptions are included in the analysis.

The proposed approaches to address this problem have been developed to explicitly encode the unloading sequence decisions, equipment movements, and yard block assignment and are introduced in the following section.

4. Methodology

This section presents the mixed-integer linear programming formulation and the heuristic approach proposed to solve TUPP, described above.

4.1. Mathematical Formulation

This subsection formally presents the mixed-integer linear programming formulation for TUPP. As already stated, the model simultaneously determines the container unloading sequence, the assignment of containers to storage blocks, and the movement of handling equipment. The objective function minimizes a weighted combination of three operational components: the travel distance of reach stackers between yard blocks, the horizontal crane travel along the train, and the number of spreader changes required during the unloading process. In particular, RS travel distance is obtained by summing Euclidean distances between consecutively assigned blocks for each reach stacker, crane distance is calculated as the cumulative wagon-position difference along the unloading sequence, and spreader changes are counted as discrete events when consecutive containers differ in size. The three components are aggregated through weight coefficients ($\alpha_1, \alpha_2, \alpha_3$), which control the relative importance assigned to each operational dimension. The sensitivity of the resulting solutions to alternative weight configurations is examined in Section 5.3.

The useful notation used in the model is summarized in Table 1, where sets and parameters are introduced together with the decision variables.

Table 1. Sets, parameters and variables used in the Train Unloading Planning Problem (TUPP).

Sets	
$\mathbb{C}$	Set of containers to be unloaded from the train.
$\mathbb{B}$	Set of yard bay-locations.
$\mathbb{D}$	Set of destinations associated with yard bay-locations.
$\mathbb{R}$	Set of reach stackers.
$\mathbb{T}$	Set of unloading operations, $\mathbb{T} = \{1, \dots, \mathbb{C} \}$.
Parameters	
λ_i	Size class of container i , $\lambda_i \in \{1, 2\}$: $\lambda_i = 1$ for 20-foot containers and $\lambda_i = 2$ for 40-foot containers, $\forall i \in \mathbb{C}$.
d_i	Destination associated with container i , $\forall i \in \mathbb{C}$.
ω_i	Weight of container i , $\forall i \in \mathbb{C}$.
ϕ_i	Weight class of container i , where $\phi_i \in \{\text{Light, Medium, Heavy}\}$, $\forall i \in \mathbb{C}$.
θ_i	Wagon index (position on the train) of container i , $\forall i \in \mathbb{C}$.
q_b	Maximum storage capacity of bay-locations b (in number of containers), $\forall b \in \mathbb{B}$.
e_b	Initial number of containers already stored in bay-locations b , $\forall b \in \mathbb{B}$.
k_b	Designated weight class for bay-locations b , where $k_b \in \{\text{Light, Medium, Heavy}\}$, $\forall b \in \mathbb{B}$.
Δ_b	Designated size class for bay-locations b , $\Delta_b \in \{1, 2\}$: $\Delta_b = 1$ for 20-foot, $\Delta_b = 2$ for 40-foot, $\forall b \in \mathbb{B}$.
σ_b	Destination of containers assigned to bay-locations b , $\forall b \in \mathbb{B}$.
$\delta_{j,l}$	Distance (in TEUs) an RS must travel when moving from bay-locations j to bay-locations l , $\forall j, l \in \mathbb{B}$.
ζ_0	Initial spreader configuration, $\zeta_0 \in \{1, 2\}$: $\zeta_0 = 1$ if the spreader is set for 20-foot containers, $\zeta_0 = 2$ if set for 40-foot.
p_0^C	Initial crane position (wagon index).
$p_0^{\text{RS},r}$	Initial bay-locations position of reach stacker r , $\forall r \in \mathbb{R}$.
$\text{sp}_{r,j}$	Binary parameter equal to 1 if $j = p_0^{\text{RS},r}$, indicating the starting bay-locations of reach stacker r ; 0 otherwise, $\forall r \in \mathbb{R}, j \in \mathbb{B}$.
$\alpha_1, \alpha_2, \alpha_3$	Weights used in the objective function for RS travel, crane travel, and spreader length changes.
Decision Variables	
$x_{i,b,t,r}$	Binary; = 1 if container i is assigned to bay-locations b by RS r at operation t , where $\phi_i = k_b$, $\lambda_i = \Delta_b$, and $d_i = \sigma_b$; 0 otherwise.
$y_{j,l,t,r}$	Binary; = 1 if RS r moves from bay-locations j to bay-locations l during operation t .
$p_{r,b,t}$	Binary; = 1 if RS r is operating in bay-locations b at operation t .
$u_{r,t}$	Binary; = 1 if RS r is idle during operation t .
z_t	Binary; = 1 if the crane changes spreader length at operation t .
s_t	Non-negative; crane travel distance required to execute operation t .

Set $\mathbb{T} = \{1, \dots, |\mathbb{C}|\}$ serves as an operation index, providing a discrete ordering of the unloading sequence rather than an explicit time representation. Although individual operations may differ in duration, the model operates at the planning level and is concerned with determining the sequencing of container unloading operations rather than computing exact execution times. The goal is to minimize the total equipment movements generated by the execution of the resulting operation sequence. In particular, the formulation does not represent equipment waiting or idle times, container completion times, or train arrival and departure windows; extending it to a standard scheduling formulation with explicit temporal constraints is identified as a direction for future work. To evaluate these movements, the initial state of each piece of handling equipment must be known: specifically, the initial crane position along the train (wagon index p_0^C), the initial spreader length (ζ_0), and the initial bay-locations position of each reach stacker r ($p_0^{\text{RS},r}$).

Let us briefly describe the decision variables. The main decision variables $x_{i,b,t,r}$ regard the assignment of each container to a bay-location and to an RS during a specific operation. These binary variables are generated only in case of compatibility between container i and bay-locations b . The remaining variables are necessary for computing the movements of the

handling equipment during the unloading process that last $|\mathbb{C}|$ operations. Binary variables $y_{j,l,t,r}$ are required to check if an RS moves from a bay-location to another one during an operation. $p_{r,b,t}$ check if an RS is operating in a bay-location during an operation and finally variables $u_{r,t}$ and z_t check for the RS status and if the crane changes the spreader length during an operation.

The proposed model, Mod1, is as follows:

$$\text{Mod1 : min} \left(\alpha_1 \cdot \sum_{j \in \mathbb{B}} \sum_{l \in \mathbb{B}} \sum_{t \in \mathbb{T}} \sum_{r \in \mathbb{R}} \delta_{j,l} \cdot y_{j,l,t,r} + \alpha_2 \cdot \sum_{t \in \mathbb{T}} s_t + \alpha_3 \cdot \sum_{t \in \mathbb{T}} z_t \right) \quad (1)$$

Subject to.

$$\sum_{i \in \mathbb{C}} \sum_{b \in \mathbb{B}} \sum_{r \in \mathbb{R}} x_{i,b,t,r} = 1 \quad \forall t \in \mathbb{T} \quad (2)$$

$$\sum_{b \in \mathbb{B}} \sum_{t \in \mathbb{T}} \sum_{r \in \mathbb{R}} x_{i,b,t,r} = 1 \quad \forall i \in \mathbb{C} \quad (3)$$

$$\sum_{i \in \mathbb{C}} \sum_{t \in \mathbb{T}} \sum_{r \in \mathbb{R}} x_{i,b,t,r} \leq q_b - e_b \quad \forall b \in \mathbb{B} \quad (4)$$

$$u_{r,t} = 1 - \sum_{i \in \mathbb{C}} \sum_{b \in \mathbb{B}} x_{i,b,t,r} \quad \forall t \in \mathbb{T}, r \in \mathbb{R} \quad (5)$$

$$p_{r,b,t} \geq \sum_{i \in \mathbb{C}} x_{i,b,t,r} \quad \forall r \in \mathbb{R}, b \in \mathbb{B}, t \in \mathbb{T} \quad (6)$$

$$\sum_{b \in \mathbb{B}} p_{r,b,t} = 1 \quad \forall r \in \mathbb{R}, t \in \mathbb{T} \quad (7)$$

$$p_{r,b,t} \geq p_{r,b,t-1} - (1 - u_{r,t}) \quad \forall b \in \mathbb{B}, \forall t \in \mathbb{T} \setminus \{1\}, \forall r \in \mathbb{R} \quad (8)$$

$$p_{r,b,1} \geq sp_{r,b} - (1 - u_{r,1}) \quad \forall b \in \mathbb{B}, \forall r \in \mathbb{R} \quad (9)$$

$$y_{j,l,t,r} \geq p_{r,j,t-1} + p_{r,l,t} - 1 \quad \forall j, l \in \mathbb{B}, j \neq l, \forall t \in \mathbb{T} \setminus \{1\}, \forall r \in \mathbb{R} \quad (10)$$

$$y_{j,l,1,r} \geq sp_{r,j} + p_{r,l,1} - 1 \quad \forall j, l \in \mathbb{B}, j \neq l, \forall r \in \mathbb{R} \quad (11)$$

$$s_t \geq \sum_{i \in \mathbb{C}} \sum_{b \in \mathbb{B}} \sum_{r \in \mathbb{R}} \theta_i \cdot x_{i,b,t,r} - \sum_{h \in \mathbb{C}} \sum_{b \in \mathbb{B}} \sum_{r \in \mathbb{R}} \theta_h \cdot x_{h,b,t-1,r} \quad \forall t \in \mathbb{T} \setminus \{1\} \quad (12)$$

$$s_t \geq \sum_{h \in \mathbb{C}} \sum_{b \in \mathbb{B}} \sum_{r \in \mathbb{R}} \theta_h \cdot x_{h,b,t-1,r} - \sum_{i \in \mathbb{C}} \sum_{b \in \mathbb{B}} \sum_{r \in \mathbb{R}} \theta_i \cdot x_{i,b,t,r} \quad \forall t \in \mathbb{T} \setminus \{1\} \quad (13)$$

$$s_1 \geq \sum_{i \in \mathbb{C}} \sum_{b \in \mathbb{B}} \sum_{r \in \mathbb{R}} \theta_i \cdot x_{i,b,1,r} - p_0^C \quad (14)$$

$$s_1 \geq p_0^C - \sum_{i \in \mathbb{C}} \sum_{b \in \mathbb{B}} \sum_{r \in \mathbb{R}} \theta_i \cdot x_{i,b,1,r} \quad (15)$$

$$z_t \geq \sum_{i \in \mathbb{C}} \sum_{b \in \mathbb{B}} \sum_{r \in \mathbb{R}} \lambda_i \cdot x_{i,b,t,r} - \sum_{h \in \mathbb{C}} \sum_{b \in \mathbb{B}} \sum_{r \in \mathbb{R}} \lambda_h \cdot x_{h,b,t-1,r} \quad \forall t \in \mathbb{T} \setminus \{1\} \quad (16)$$

$$z_t \geq \sum_{h \in \mathbb{C}} \sum_{b \in \mathbb{B}} \sum_{r \in \mathbb{R}} \lambda_h \cdot x_{h,b,t-1,r} - \sum_{i \in \mathbb{C}} \sum_{b \in \mathbb{B}} \sum_{r \in \mathbb{R}} \lambda_i \cdot x_{i,b,t,r} \quad \forall t \in \mathbb{T} \setminus \{1\} \quad (17)$$

$$z_1 \geq \sum_{i \in \mathbb{C}} \sum_{b \in \mathbb{B}} \sum_{r \in \mathbb{R}} \lambda_i \cdot x_{i,b,1,r} - \zeta_0 \quad (18)$$

$$z_1 \geq \zeta_0 - \sum_{i \in \mathbb{C}} \sum_{b \in \mathbb{B}} \sum_{r \in \mathbb{R}} \lambda_i \cdot x_{i,b,1,r} \quad (19)$$

$$s_t \geq 0 \quad \forall t \in \mathbb{T} \quad (20)$$

$$x_{i,b,t,r}, y_{j,l,t,r}, p_{r,b,t}, u_{r,t}, z_t \in \{0, 1\} \quad \forall i \in \mathbb{C}, b, j, l \in \mathbb{B}, t \in \mathbb{T}, r \in \mathbb{R}, j \neq l \quad (21)$$

The objective function (1) minimizes the weighted sum of reach stacker travel, crane travel along the train, and crane spreader length adjustments.

Constraints (2) ensure that exactly one container is unloaded at each operation t . Constraints (3) guarantee that each container is unloaded exactly once. Constraints (4) enforce bay-locations capacity limits, ensuring that assignments to bay-locations b do not exceed its residual capacity $q_b - e_b$. Constraints (5) define the idle indicator $u_{r,t}$ that is fixed equal to 1 when reach stacker r is unassigned during operation t . Constraints (6) and (7) govern positional tracking: constraints (6) link $p_{r,b,t}$ to the assignment decisions, while constraints (7) ensure each reach stacker uses exactly one bay-location at every operation. Constraints (8) preserve positional continuity for idle reach stackers, keeping them in their last worked bay-location between consecutive operations; constraints (9) provide the corresponding initialization for $t = 1$ via the binary parameter $sp_{r,b}$. Constraints (10) and (11) activate variables $y_{j,l,t,r}$ whenever reach stacker r transitions between bay-locations in consecutive operations, with constraints (11) handling the initialization from the starting position $sp_{r,j}$. Constraints (12) and (13) compute the crane travel distance s_t as the absolute difference between the wagon positions of consecutively handled containers; for $t = 1$, constraints (14) and (15) measure this distance from the crane's initial position p_0^C . Since s_t appears in the minimization objective and $s_t \geq 0$ is enforced by (20), the absolute value is equivalently represented at optimality without loss of linearity. Constraints (16) and (17) activate the binary variable z_t whenever consecutive containers differ in size, capturing spreader length changes; for $t = 1$, constraints (18) and (19) compare the size of the first handled container against the initial spreader configuration ζ_0 . Finally, (20) and (21) define the variable domains, imposing non-negativity on s_t and binary restrictions on all remaining decision variables.

Valid Inequalities

To strengthen the MILP formulation and improve solver performance, we propose the following valid inequalities (VIs). These inequalities enforce logical consistency in container assignment and reach stacker movement decisions. Although they are logically implied by the structure of the problem, their explicit inclusion helps tighten the formulation and reduces the size of the branch-and-bound search tree. Several alternative VI formulations were tested during the development of this work; while their impact on solution quality proved marginal, the selected inequalities were retained due to their contribution to reducing solver computation time, particularly for medium and large instances, with an average objective value improvement of approximately 5% observed across the tested instances.

$$\sum_{r \in \mathbb{R}} \sum_{b \in \mathbb{B}} x_{i,b,t,r} + \sum_{r \in \mathbb{R}} \sum_{b \in \mathbb{B}} x_{i,b,t',r} \leq 1 \quad \forall i \in \mathbb{C}, \forall t, t' \in \mathbb{T}, t < t' \quad (22)$$

$$\sum_{l \in \mathbb{B}, l \neq j} y_{j,l,t,r} \leq \sum_{i \in \mathbb{C}} \sum_{k=1}^{t-1} x_{i,j,k,r} + sp_{r,j} \quad \forall j \in \mathbb{B}, t \in \mathbb{T}, t > 1, r \in \mathbb{R} \quad (23)$$

$$\sum_{j \in \mathbb{B}, j \neq l} y_{j,l,t,r} \leq \sum_{i \in \mathbb{C}} x_{i,l,t,r} \quad \forall l \in \mathbb{B}, t \in \mathbb{T}, r \in \mathbb{R} \quad (24)$$

$$y_{j,l,t,r} \geq \sum_{i \in \mathbb{C}} x_{i,j,t-1,r} + \sum_{i \in \mathbb{C}} x_{i,l,t,r} - 1 \quad \forall j, l \in \mathbb{B}, j \neq l, t \in \mathbb{T} \setminus \{1\}, r \in \mathbb{R} \quad (25)$$

VI (22) ensures that each container i is assigned at most once over the entire unloading sequence, providing an explicit pairwise strengthening of the assignment uniqueness already enforced by (3), which tightens the LP relaxation by adding cuts across all operation pairs (t, t') . VI (23) enforces that a reach stacker can depart from bay-locations j at operation t only if it has previously operated there in some earlier operation $k < t$, or if

j is its designated starting position ($sp_{r,j} = 1$), preventing physically infeasible departure movements from bay-locations never visited. VI (24) prevents reach stacker movements toward bay-locations l when no container is assigned to that bay-locations at operation t .

Finally, VI (25) guarantees consistency between consecutive reach stacker positions by activating the transition variable $y_{j,l,t,r}$ whenever the reach stacker operates at different blocks in successive operations, directly linking the position tracking variables $p_{r,b,t}$ to the movement cost in the objective function.

4.2. Heuristic Solution Framework

Preliminary computational experiments on model *Mod1* presented above revealed that the combinatorial complexity of the problem grows rapidly with instance size, making exact solution methods computationally intractable for realistic operational scenarios. For medium- to large-scale instances, commercial optimization solvers consistently reach the imposed time limit of 3600 s without proving optimality (as detailed in Section 5.2). These required computational efforts are incompatible with the short response times needed for operational decision-making, especially when re-planning becomes necessary due to train delays, equipment breakdowns, or updated container information. To overcome these limitations, while preserving solution quality, we develop a structured heuristic optimization framework that exploits the operational characteristics of the unloading process and progressively refines feasible solution through complementary search mechanisms.

The proposed framework integrates two algorithmic components that balance computational efficiency with solution quality: (i) a warm-start procedure using time-limited MILP solving to generate initial solutions, and (ii) a local search schema combining two complementary neighborhood operators with controlled perturbation-based diversification.

The local search schema systematically improves the initial solution through alternating intensification and diversification phases, exploiting operator complementarity to achieve sustained quality improvement while maintaining computational tractability.

4.2.1. Solution Representation and Evaluation

A solution to the TUPP is represented as an ordered sequence of operations $S = (o_1, o_2, \dots, o_{|\mathbb{C}|})$, where each operation o_t for $t \in \mathbb{T}$, is defined as $o_t = (i_t, b_t, r_t)$, that corresponds to:

- The unloading of a specific container $i \in \mathbb{C}$;
- The assignment of container i to a yard bay-locations $b \in \mathbb{B}$;
- The identification of the reach stacker $r \in \mathbb{R}$ responsible for the storage in the yard bay-location b .

The sequence order implicitly determines the crane's temporal position along the train at each step, denoted by the wagon index θ_i of container i , while the crane's spreader configuration at operation t is defined by the container size λ_i . The reach stacker position is known in accordance with the last operation assigned to the reach stacker, assuming that in t if a reach stacker is idle, then it remains in the same position.

Each solution S is evaluated with respect to the three components as defined in the MILP formulation (Equation (1)). Let it be $f(S)$ for this evaluation.

The feasibility of the initial solution is guaranteed by construction, as it is the best solution found by the MILP (*Mod1*) within the imposed time limit, and feasibility is preserved throughout all heuristic steps; all neighborhood operators ensure that:

- Each container i is assigned to a bay-location b among those satisfying $\phi_i = k_b$ (weight compatibility), $\lambda_i = \Delta_b$ (size compatibility), and $d_i = \sigma_b$ (destination compatibility);

- Bay-location capacity constraints $\sum_i x_{i,b,t,r} \leq q_b - e_b$ are respected for all bay-locations $b \in \mathbb{B}$;
- At each operation $t \in \mathbb{T}$, each reach stacker performs at most one container handling activity, ensuring that no equipment conflicts arise and that idle reach stackers remain available for subsequent operations.

4.2.2. Neighborhood Operators

The local search framework employs two complementary neighborhood operators to explore the solution space systematically. Each operator targets a distinct aspect of the unloading plan, sequence reordering or positional restructuring, while equipment reallocation is handled by the embedded greedy RS re-routing sub-procedure, thereby providing diverse exploration capabilities. The operators differ in their granularity and disruptiveness, enabling the search to perform both fine grained refinement within promising solution regions and broader structural modifications to escape local optima. This multi granularity design follows established principles in local search heuristics, where perturbation mechanisms enable escape from local optima [23] and combining constructive and improvement phases with diversification strategies enhances solution quality [24].

Greedy RS Re-Routing

Prior to describing the neighborhood operators, we introduce the Greedy RS Re-routing (GRR) sub-procedure, which is embedded within each operator to determine the reach stacker assignment for a given container sequence. Given a solution, that is a fixed ordered sequence $S = ((i_1, b_1), \dots, (i_n, b_n))$ and the initial positions $p_0^{\text{RS},r}$ of each reach stacker $r \in \mathbb{R}$, GRR iterates over operations in sequence order and assigns each operation t to the reach stacker currently closest to the target bay-locations b_t , breaking ties in favour of the lower-indexed stacker. After each assignment, the position of the selected stacker is updated to b_t . This greedy nearest-RS rule is applied from scratch after every sequence modification, thereby decoupling the sequence optimization decisions from the equipment allocation decisions and guaranteeing that the RS assignment is always consistent with the current sequence structure. Algorithm 1 formalizes the procedure.

Algorithm 1: Greedy RS Re-routing

Input: Sequence $S = ((i_1, b_1), \dots, (i_n, b_n))$;
 initial RS positions $p_0^{\text{RS},r}$;
 distance function $\delta(\cdot, \cdot)$
Output: RS assignment $r^* = (r_1^*, \dots, r_n^*)$

for $r \in \mathbb{R}$ **do**
 $\text{pos}[r] \leftarrow p_0^{\text{RS},r}$

for $t = 1$ **to** n **do**
 $r_t^* \leftarrow \arg \min_{r \in \mathbb{R}} \delta(\text{pos}[r], b_t)$ // nearest RS wins; ties broken by lower index
 $\text{pos}[r_t^*] \leftarrow b_t$

return r^*

Operator 1: Swap-Based Reordering

The swap operator performs localized sequence adjustments by exchanging the temporal positions of two operations o_{t_1} and o_{t_2} in the sequence in S .

Given two distinct operations characterized by $t_1, t_2 \in \mathbb{T}$ with $t_1 < t_2$, the operator generates a neighbour solution S' ; let

$$S' = (o_1, \dots, o_{t_1-1}, o_{t_2}, o_{t_1+1}, \dots, o_{t_2-1}, o_{t_1}, o_{t_2+1}, \dots, o_{|\mathbb{C}|}). \quad (26)$$

After constructing S' , GRR is invoked on the modified sequence to recompute the RS assignment from scratch, yielding an updated assignment r' that is consistent with the new operation order. The objective is then evaluated on the pair (S', r') . Reordering affects all three objective components. Crane travel changes due to modified wagon position sequences, reach stacker travel may change because swapping alters the operation positions and the GRR may yield a different equipment allocation, and spreader changes are modified when the swap alters adjacency between containers of different sizes.

The swap neighborhood contains $\binom{|\mathbb{T}|}{2} = O(|\mathbb{T}|^2)$ distinct neighbors. This minimal modification may be particularly effective for correcting small sequencing inefficiencies, such as when two containers at adjacent wagon positions are scheduled far apart in the unloading sequence.

Algorithm 2 presents the complete Swap-Based Reordering move. The algorithm follows a randomized improvement strategy: at each iteration, a random pair (t_1, t_2) is selected uniformly from $\mathbb{T}$, GRR is applied to the modified sequence, the swap move is evaluated using incremental objective calculation, and the move is accepted immediately if it yields a strictly lower objective value. This differs from classical first-improvement search, which scans neighbors in a fixed order; here, a random neighbour is evaluated at each step, providing an unbiased stochastic exploration of the $\binom{|\mathbb{T}|}{2}$ swap neighborhood. To improve computational efficiency, only the cost components affected by positions from $\min(t_1, t_2)$ onward are recomputed. The search terminates after $N_{\max}$ consecutive non-improving iterations, indicating convergence to a local optimum with respect to the swap neighborhood.

Algorithm 2: Swap-Based Reordering move

Input: Initial solution S ; maximum non-improving iterations $N_{\max}$; RS start positions $p_0^{\text{RS},r}$

Output: Improved solution $\hat{S}$

// Initialization

$\hat{S} \leftarrow S$

$\hat{r} \leftarrow \text{GRR}(\hat{S})$

Compute $f(\hat{S}, \hat{r})$ using Equation (1)

$n_{\text{no-improve}} \leftarrow 0$

// Input validation

if $|\mathbb{T}| < 2$ **then**

return $\hat{S}$

// Iterative improvement loop

while $n_{\text{no-improve}} < N_{\max}$ **do**

 // Generate random swap move

 Randomly select two distinct positions $t_1, t_2 \in \mathbb{T}$ with $t_1 < t_2$

 // Construct neighbour solution

$S' \leftarrow \hat{S}$ with operations o_{t_1} and o_{t_2} exchanged

$r' \leftarrow \text{GRR}(S')$

 // full RS reassignment

 Recompute affected cost components from position $\min(t_1, t_2)$ onward

 Compute $f(S', r')$ using Equation (1)

 // Acceptance decision (first improvement)

if $f(S', r') < f(\hat{S}, \hat{r})$ **then**

$\hat{S} \leftarrow S'$

$\hat{r} \leftarrow r'$

$n_{\text{no-improve}} \leftarrow 0$

else

$n_{\text{no-improve}} \leftarrow n_{\text{no-improve}} + 1$

return $\hat{S}$

Operator 2: Repositioning

The repositioning operator performs more disruptive sequence modifications than swaps, enabling escape from local optima that trap pairwise exchange operators. This operator extracts one operation, here denoted $o_{t_{\text{remove}}}$, from its current position and reinserts it at a new position t_{insert} , shifting all intermediate operations to maintain sequence continuity. For forward insertion ($t_{\text{remove}} < t_{\text{insert}}$),

$$S' = (o_1, \dots, o_{t_{\text{remove}}-1}, o_{t_{\text{remove}}+1}, \dots, o_{t_{\text{insert}}}, o_{t_{\text{remove}}}, o_{t_{\text{insert}}+1}, \dots, o_{|\mathbb{C}|}), \quad (27)$$

and for backward insertion ($t_{\text{remove}} > t_{\text{insert}}$),

$$S' = (o_1, \dots, o_{t_{\text{insert}}-1}, o_{t_{\text{remove}}}, o_{t_{\text{insert}}}, \dots, o_{t_{\text{remove}}-1}, o_{t_{\text{remove}}+1}, \dots, o_{|\mathbb{C}|}). \quad (28)$$

Following the construction of S' , GRR is applied from scratch to recompute the RS assignment r' consistent with the repositioned sequence. The move is then evaluated on the pair (S', r') .

Repositioning moves modify the relative ordering of all operations between t_{remove} and t_{insert} . Since the container block assignment (i_t, b_t) of the relocated operation remains unchanged and GRR guarantees a feasible RS allocation for any sequence, feasibility is preserved by construction. All three objective components are potentially affected, as reordering alters crane travel distances, reach stacker routing costs as recomputed by GRR, and spreader change counts along the modified sequence segment.

The repositioning neighborhood contains $|\mathbb{T}| \cdot (|\mathbb{T}| - 1) = O(|\mathbb{T}|^2)$ distinct neighbors. Algorithm 3 presents the complete repositioning move procedure.

Algorithm 3: Repositioning move

Input: Initial solution S ; maximum non-improving iterations N_{max} ; RS start positions $p_0^{\text{RS},r}$
Output: Improved solution $\hat{S}$

```

// Initialization
 $\hat{S} \leftarrow S$ 
 $\hat{r} \leftarrow \text{GRR}(\hat{S})$ 
Compute  $f(\hat{S}, \hat{r})$  using Equation (1)
 $n_{\text{no-improve}} \leftarrow 0$ 

// Iterative improvement loop
while  $n_{\text{no-improve}} < N_{\text{max}}$  do
    // Select operation and insertion position
    Randomly select operation position  $t_{\text{remove}} \in \mathbb{T}$ 
    Randomly select insertion position  $t_{\text{insert}} \in \mathbb{T}$  with  $t_{\text{insert}} \neq t_{\text{remove}}$ 

    // Construct neighbour solution
     $S' \leftarrow \hat{S}$  with operation  $o_{t_{\text{remove}}}$  removed from position  $t_{\text{remove}}$ 
    Insert  $o_{t_{\text{remove}}}$  at position  $t_{\text{insert}}$  in  $S'$ 

     $r' \leftarrow \text{GRR}(S')$  // full RS reassignment
    Recompute all affected cost components from position  $\min(t_{\text{remove}}, t_{\text{insert}})$  onward
    Compute  $f(S', r')$  using Equation (1)

    // Acceptance decision
    if  $f(S', r') < f(\hat{S}, \hat{r})$  then
         $\hat{S} \leftarrow S'$ 
         $\hat{r} \leftarrow r'$ 
         $n_{\text{no-improve}} \leftarrow 0$ 
    else
         $n_{\text{no-improve}} \leftarrow n_{\text{no-improve}} + 1$ 

return  $\hat{S}$ 

```

The local search framework therefore operates with two primary sequence-level operators, swap and repositioning, whose combined neighborhood is augmented by GRR-based equipment re-routing at every candidate solution evaluation. The sequential application of these two operators within the local search schema, described in the following subsection,

leverages their complementary strengths to achieve sustained solution quality improvement while maintaining computational tractability. The GRR sub-procedure, embedded within both operators, ensures that RS allocation is always optimised with respect to the current sequence, removing the need for an independent RS reassignment search phase.

Table 2 provides a comparative overview of the two operators' characteristics.

Table 2. Comparative characteristics of neighborhood operators.

Characteristic	Swap (+ GRR)	Insertion (+ GRR)
Modification granularity	Fine	Coarse
Affected positions	2	$ t_{\text{remove}} - t_{\text{insert}} $
Sequence components affected	3	3
RS assignment	GRR (full re-route)	GRR (full re-route)
Neighborhood size	$O(\mathbb{T} ^2)$	$O(\mathbb{T} ^2)$
Evaluation complexity	$O(\mathbb{T})$	$O(\mathbb{T})$

4.2.3. Local Search Schema with Perturbation

The proposed local search schema executes cyclical iterations, each comprising two phases: (i) intensification through sequential local search, and (ii) diversification through perturbation. This approach follows the Iterated Local Search framework [25], a well-established heuristic that alternates between intensification phases (improving current solution) and diversification phases (exploring new solution regions). ILS has demonstrated effectiveness across diverse combinatorial optimization problems including container pre-marshalling [26] and parallel machine scheduling [24].

Starting from the initial solution generated by MILP, GRR is first applied to establish a consistent RS assignment for the warm-start sequence. The two neighborhood operators are then applied sequentially in order of increasing scope: swap reordering affects two positions locally with GRR the RS allocation after each accepted move, and repositioning restructures larger sequence segments with GRR similarly embedded within each evaluation. Each operators execute until N_{max} consecutive iterations fail to identify improvements, at which point the operator has converged to a local optimum with respect to its neighborhood. The output of each operator serves as input to the next, creating a cascading refinement process that exploits operator complementarity. Since RS allocation is resolved by GRR at every candidate evaluation, the search is conducted exclusively over the space of container sequences, with equipment assignment implicitly optimised throughout.

Upon completing both local search phases, the solution represents a local optimum with respect to the combined neighborhood structure. At this point, further improvement requires escaping the current basin of attraction. We employ a perturbation mechanism to achieve this diversification. The perturbation operator randomly selects three distinct operations from positions $\{t_1, t_2, t_3\} \subset \mathbb{T}$ and relocates them to new positions $\{p_1, p_2, p_3\} \subset \mathbb{T}$, with all six positions mutually distinct. Intermediate operations are shifted to maintain sequence continuity. Following perturbation, GRR is applied to the modified sequence before the next intensification cycle begins, ensuring that the RS assignment entering the subsequent local search phase is consistent with the perturbed sequence structure. This random three-swap perturbation deliberately degrades the current solution, enabling the subsequent local search iteration to explore a different region of the solution space.

Algorithm 4 formalizes the complete procedure. The perturbation strength quantified by the number of simultaneously relocated operations directly affects the balance between diversification and structure preservation. Weak perturbations (1–2 operations) risk insufficient diversification, potentially returning the search to previously visited local optima. Strong perturbations (5+ operations) excessively disrupt solution structure, requiring extended re-optimization time and potentially destroying valuable structural properties

accumulated through prior search phases. We selected three-swap perturbations based on preliminary experiments showing this configuration provides adequate escape capability while maintaining reasonable convergence speed. This choice aligns with findings in the scheduling literature where moderate-strength perturbations consistently outperform extreme alternatives [23].

Algorithm 4: Local Search Schema with Perturbation

Input: Initial solution S_0 from MILP; maximum non-improving iterations $N_{\max}$; number of cycles N_{cycles} ; RS start positions $p_0^{\text{RS},r}$

Output: Best solution S^{best} , RS assignment r^{best}

// Initialization

$r_0 \leftarrow \text{GRR}(S_0)$

// initial RS assignment

$S^{\text{best}} \leftarrow S_0$

$r^{\text{best}} \leftarrow r_0$

$S^{\text{current}} \leftarrow S_0$

Compute $f(S^{\text{best}}, r^{\text{best}})$ using Equation (1)

for $c = 1$ **to** N_{cycles} **do**

 // Intensification: sequential application of two operators

$S^{(1)} \leftarrow \text{Swap-LocalSearch}(S^{\text{current}}, N_{\max}, p_0^{\text{RS},r})$

// Algorithm 2

$S^{(2)} \leftarrow \text{Insertion-LocalSearch}(S^{(1)}, N_{\max}, p_0^{\text{RS},r})$

// Algorithm 3

$r^{(2)} \leftarrow \text{GRR}(S^{(2)})$

 Compute $f(S^{(2)}, r^{(2)})$

 // Update global best if improvement found (elitist)

if $f(S^{(2)}, r^{(2)}) < f(S^{\text{best}}, r^{\text{best}})$ **then**

$S^{\text{best}} \leftarrow S^{(2)}$

$r^{\text{best}} \leftarrow r^{(2)}$

 // Diversification: 3-swap perturbation (except after final cycle)

if $c < N_{\text{cycles}}$ **then**

 Randomly select three distinct positions $t_1, t_2, t_3 \in \mathbb{T}$

 Randomly select three distinct target positions $p_1, p_2, p_3 \in \mathbb{T} \setminus \{t_1, t_2, t_3\}$

 Construct S^{pert} by removing operations at t_1, t_2, t_3 and inserting at p_1, p_2, p_3

 Shift intermediate operations to maintain sequence continuity

$S^{\text{current}} \leftarrow S^{\text{pert}}$

// Continue search from perturbed solution

return $S^{\text{best}}, r^{\text{best}}$

Following perturbation, the two operator local search sequence repeats on the modified solution. The schema maintains an elitist selection strategy: the best solution encountered throughout all cycles (including both pre and post perturbation states) is retained as the global incumbent. This ensures that temporary solution degradation induced by perturbation does not compromise overall solution quality, guaranteeing monotonic improvement in the best solution found.

The schema's behavior is governed by two parameters. The parameter $N_{\max}$ controls when each individual operator terminates, balancing thoroughness of neighborhood exploration against computational cost. The parameter N_{cycles} determines how many complete intensification-diversification cycles execute, directly affecting total runtime and solution space coverage. We systematically calibrated both parameters through computational experiments on representative instance subsets, as detailed in Section 5.4. These experiments identified $N_{\max} = 1000$ and $N_{\text{cycles}} = 4$ as providing effective balance between solution quality and computational efficiency for the considered problem instances.

The proposed heuristic approach adapts the general ILS framework to the TUPP through four specific design choices. First, the local search phase employs two heterogeneous neighborhood operators each augmented with embedded GRR, rather than three operators with independent equipment allocation, enabling more comprehensive exploration of different solution aspects. Second, operators are applied sequentially rather than alternately, exploiting their complementary strengths through cascading refinement. Third, the acceptance criterion is purely elitist, always retaining the best solution found, rather than employing probabilistic or threshold-based acceptance as in classical ILS variants. This prioritizes solution quality over search diversity, appropriate for operational planning contexts where solution reliability matters more than exploration breadth. Fourth, the termination criterion uses a fixed number of cycles rather than time-based or convergence-based stopping, providing predictable computational requirements suitable for integration into operational planning systems. These design choices were validated through preliminary experiments comparing against standard ILS configurations, demonstrating superior performance on TUPP instances within typical operational time constraints.

Figure 2 illustrates the complete local search schema, showing the cyclic alternation between intensification through sequential operator application with embedded GRR and diversification through three-swap perturbation.

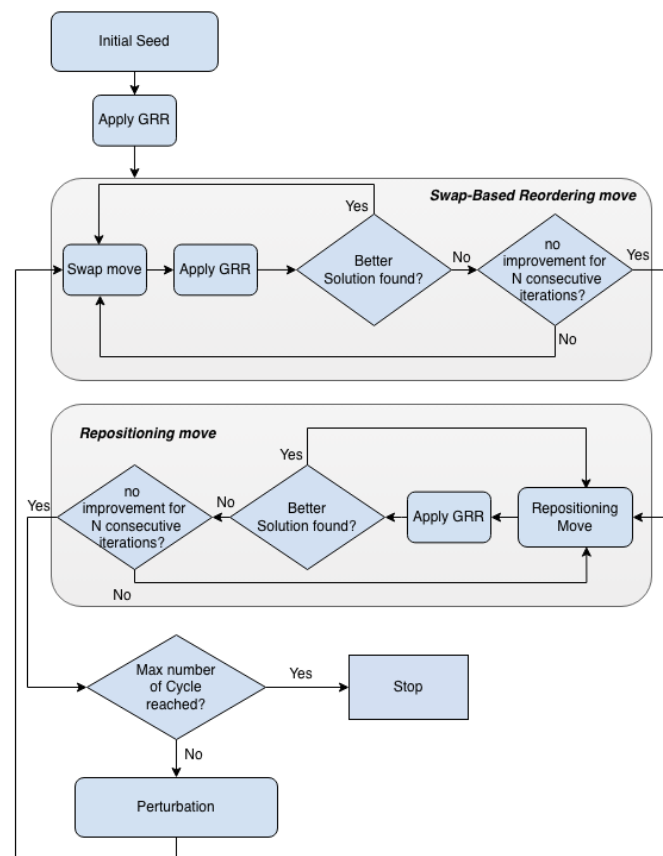


Figure 2. Local Search Schema with Perturbation and embedded Greedy RS Re-routing.

5. Computational Results

This section presents a systematic computational evaluation of the proposed solution framework using benchmark instances designed to replicate realistic train unloading scenarios at maritime container terminals operating in Italy. The analysis is organized as follows. Section 5.1 describes the experimental setup and benchmark design. Section 5.2 examines the computational behavior of the MILP model under a fixed time limit. Section 5.3 investigates how objective function weight configurations shape solu-

tion structure and operational trade-offs. Section 5.4 calibrates the local search parameters to identify the best balance between solution quality and computation time. Finally, Section 5.5 compares the ILS against the MILP model and quantifies the contribution of individual algorithmic components through ablation experiments.

5.1. Experimental Setup

All computational experiments were conducted on a desktop computer equipped with a 12th Gen Intel Core i5-1235U processor operating at 1.30 GHz with 8 GB of RAM, running Windows 11 (64-bit). The MILP formulation presented in Section 4.1 was solved using Gurobi 11.0.3 with default solver settings and a time limit of 3600 seconds. The ILS heuristic described in Section 4.2.3 was implemented in Python 3.11 within the PyCharm 2024.1 environment. Both approaches executed in single-threaded mode to ensure reproducibility and enable fair performance comparisons.

We designed a benchmark of 16 instance groups (G1–G16) that systematically vary problem complexity across two structural configurations. Groups G1–G8 represent single destination operations, where all train containers are destined for a single yard vessel area and distributed among six compatible storage bay-locations. Groups G9–G16 represent dual destination operations, where containers are addressed to two distinct yard vessel areas requiring distribution across twelve bay-locations, six per destination. Each group contains three instances sharing identical structural parameters but generated with different random seeds, yielding 16 groups of benchmark instances in total.

Problem size grows systematically across groups, with container volumes of 10, 15, 20, 30, 40, 50, 60, and 70 units, spanning small scale tactical scenarios to operationally realistic high volume settings. Container dimensions follow industry standard 20-foot and 40-foot TEU classifications, with approximately 55% of containers being 20-foot units and the remaining 45% of 40-foot type. Container weights are drawn uniformly at random from the integer range [3, 33] tonnes and classified into three weight classes according to the following thresholds: Light (weight < 15 t), Medium ($15 \leq \text{weight} < 24$ t), and Heavy (weight ≥ 24 t).

The export yard follows a consignment storage strategy in which each bay-locations is exclusively dedicated to containers sharing the same destination, size class, and weight class. Individual bay-locations capacities range from 12 to 20 containers depending on stack configuration, with residual capacities initialised to reflect a partially occupied yard at the start of each instance. Bay-locations are distributed across all three weight classes and both container sizes to ensure that every container in each instance has at least one compatible bay-locations available. The terminal equipment configuration consists of two reach stackers positioned at predetermined initial locations within the yard, consistent with typical operational practice at intermodal facilities.

Although synthetically generated, the benchmark instances are designed to reflect the operational reality of the conventional maritime container terminals in Italy that motivate this study, both in their structural and parametric characteristics and in the two predominant operational patterns reproduced, in which an arriving train feeds either one or two vessel areas. The range of train sizes, up to the 70-container fully loaded trains encountered in daily practice, ensures that the evaluation covers operationally realistic settings.

5.2. Performance of the MILP Model

Table 3 reports the computational performance of the MILP formulation across all 16 instance groups under a 3600 s time limit, while Figure 3 simultaneously visualizes optimality gap growth and solver time consumption across the full benchmark. Each

bubble encodes three dimensions: horizontal position reflects problem size, vertical position reflects the optimality gap, and bubble diameter reflects computation time.

Table 3. MILP performance under 3600s time limit.

Group	#C	Obj.	Gap (%)	Time (s)	RS Dist.	Crane Dist.	Sprd. Chg.
G1	10	11.05	0.00	101.3	9.50	18.33	3.67
G2	15	19.52	32.00	3600	16.67	34.83	3.33
G3	20	24.32	52.57	3600	12.50	56.67	5.33
G4	30	41.48	76.14	3600	22.17	96.00	8.00
G5	40	52.53	81.60	3600	26.33	125.00	9.00
G6	50	92.00	87.13	3600	31.17	250.50	6.33
G7	60	117.95	87.99	3600	50.83	297.33	16.33
G8	70	155.08	89.41	3600	47.67	426.17	17.00
G9	10	17.48	0.00	309.1	17.50	26.00	4.67
G10	15	23.57	35.00	3600	25.50	33.17	4.00
G11	20	26.98	55.00	3600	28.00	39.50	5.66
G12	30	50.68	78.14	3600	36.00	103.80	7.33
G13	40	72.93	85.50	3600	53.50	146.16	11.33
G14	50	124.45	89.34	3600	100.83	239.67	10.00
G15	60	150.92	90.00	3600	116.30	297.83	16.33
G16	70	163.83	89.57	3600	152.00	280.33	18.67

The model achieves proven optimality only for 10-container instances (G1 and G9), requiring 101.3 and 309.1 s respectively. The three-fold difference in solution time between these two groups despite identical container volumes reveals the combinatorial overhead introduced by the second destination, which doubles the number of candidate storage bay-locations and substantially enlarges the assignment space. All remaining 14 groups exhaust the 3600 s limit without closing the gap.

Gap escalation is rapid and monotone across both configurations. Within the small size (15 containers), gaps already reach 32–35%; they climb to 52–55% at 20 containers and exceed 76% at 30 containers. Beyond 50 containers, gaps plateau between 87% and 90%, as indicated by the dotted reference line in Figure 3. Single and dual destination configurations follow nearly parallel trajectories throughout the benchmark. Dual destination instances carry a systematic 2–3 percentage-point gap penalty within the small–medium range (G2–G5 versus G10–G13), consistent with the larger feasible assignment space arising from the second destination zone.

These findings establish that exact optimization is not computationally viable, except for instances with at most 10 containers, well below the volumes encountered in practice at the terminals. The large and rapidly growing optimality gaps for instances with 20 or more containers, combined with consistently exhausted time limits, provide the central motivation for the heuristic approach evaluated in Section 5.5.

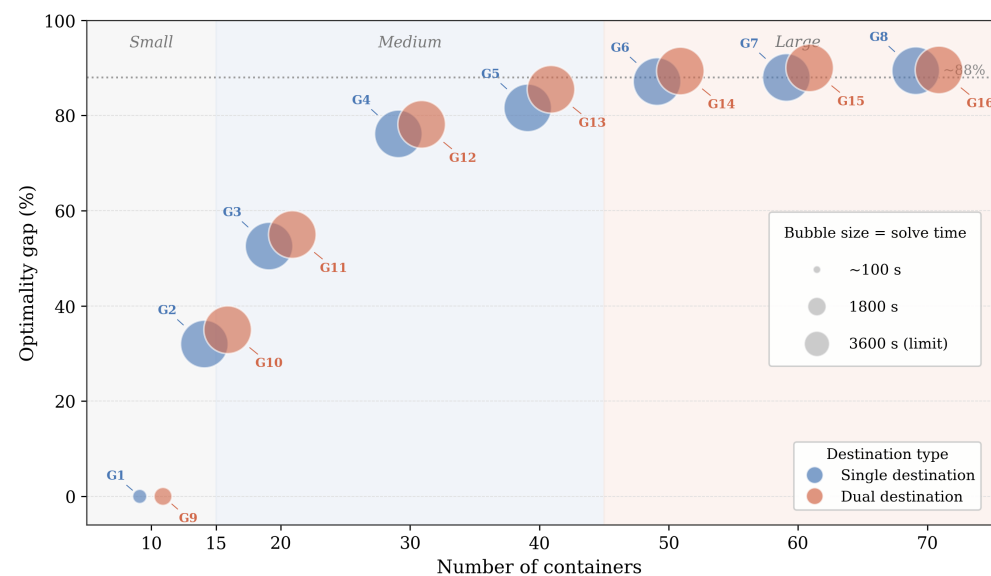


Figure 3. Optimality gap and solver time for the MILP model across all instance groups.

5.3. Impact of Weight Configuration on Solution Characteristics

The objective function combines three operational components reach stacker travel (α_1), crane horizontal travel (α_2), and spreader changes (α_3) whose relative weighting directly shapes the character of the resulting unloading plan. To examine how prioritizing each component affects both solution structure and solver tractability across instance sizes, the MILP is solved under the seven configurations defined in Table 4, spanning single-objective extremes (W2–W4), a balanced multi-objective setting (W1), and pairwise combinations (W5–W7).

Table 4. Weight configurations for objective function components.

Weight Config.	α_1 (RS)	α_2 (Crane)	α_3 (Spreader)
W1	0.5	0.3	0.2
W2	1.0	0.0	0.0
W3	0.0	1.0	0.0
W4	0.0	0.0	1.0
W5	0.5	0.0	0.5
W6	0.5	0.5	0.0
W7	0.0	0.5	0.5

The three components of the objective capture distinct operational concerns that carry different managerial weight rather than commensurable physical quantities. reach stacker travel and crane travel are the principal determinants of handling effort and equipment utilization, whereas spreader changes represent a secondary cost. The coefficients ($\alpha_1, \alpha_2, \alpha_3$) therefore encode a deliberate priority ordering $\alpha_1 = 0.5 > \alpha_2 = 0.3 > \alpha_3 = 0.2$ that reflects the relative operational importance of each concern, rather than serving as physical conversion factors between heterogeneous units. Because any such prioritization is ultimately a managerial choice, the seven configurations W1–W7 examined in this section constitute a systematic sensitivity analysis of the weights and quantifying how each choice reshapes the resulting operational trade-offs.

The radar chart in Figure 4 composes the three normalised component averages for each configuration, providing an immediate visual characterisation of the operational trade-offs induced by weight selection. The profiles are strikingly divergent. Single-objective configurations collapse toward one axis while driving the others to their maximum: W2

($\alpha_1 = 1$) suppresses RS travel to near zero but produces the highest crane distances across the benchmark, while W3 ($\alpha_2 = 1$) achieves the lowest crane distances at the cost of substantially elevated RS routing. W4 ($\alpha_3 = 1$) concentrates near the origin on both equipment axes, confirming that minimizing spreader changes alone imposes minimal travel demands on either crane or reach stackers. Pairwise configurations (W5–W7) retain the directional bias of their dominant component, producing asymmetric profiles that consistently disadvantage the unweighted dimension.

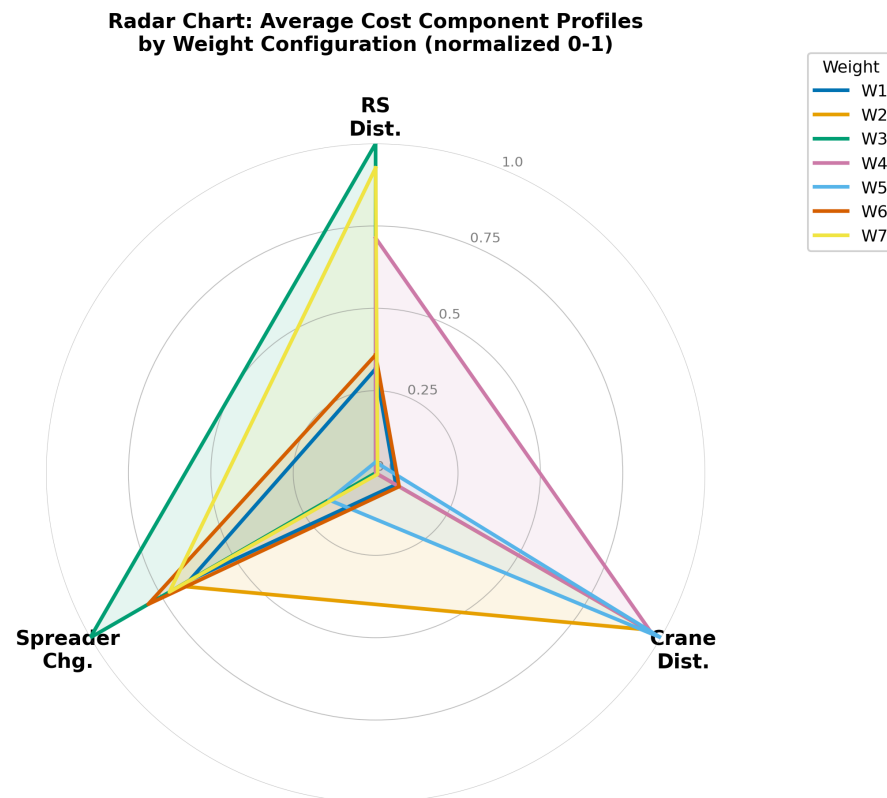


Figure 4. Normalised cost component profiles per weight configuration (W1–W7).

Objective values scale consistently with problem size under all configurations, as reported in Table 5. For single destination instances, the group averages under W1 rise from 8.13 (G1, 10 containers) to 117.85 (G8, 70 containers), a 14.5-fold increase, while under W3 the range extends from 13.50 to 266.50, a 19.7-fold increase, confirming that crane-dominated objectives degrade more severely with scale. Dual destination instances follow the same trend at a systematically higher level; the W1 average for G16 (70 containers, dual destination) reaches 163.83 compared with 155.08 for the comparable single destination group G8, a difference attributable to the increased routing complexity introduced by the second destination zone. Gap sensitivity to weight configuration becomes pronounced only from G14 onward ($n \geq 50$), where crane-dominant and pairwise configurations (W3 and W6) exhibit gaps exceeding 75%, while the spreader-only configuration (W4) retains a gap of 0.00% throughout, reflecting the negligible solver difficulty of that degenerate single-component landscape.

The balanced configuration W1 exhibits the most equidistant profile from the chart origin, distributing operational workload evenly across all three dimensions. This structural property, rather than any advantage in raw objective magnitude, motivates its adoption as the reference configuration for all heuristic experiments reported in Section 5.5.

Table 5. MILP performance under alternative weight configurations across instance groups.

Single Destination (6 Bay-Locations)							Dual Destination (12 Bay-Locations)						
Inst.	Wgt.	Obj.	Gap (%)	RS Dist.	Crane Dist.	Sprd. Chg.	Inst.	Wgt.	Obj.	Gap (%)	RS Dist.	Crane Dist.	Sprd. Chg.
G1	W1	11.05	0.00	9.50	18.33	3.67	G9	W1	17.48	0.00	17.50	26.00	4.67
	W2	5.50	0.00	5.50	58.83	2.33		W2	14.17	0.00	14.17	51.83	3.33
	W3	13.50	0.00	33.17	13.50	4.67		W3	13.50	0.00	56.83	13.50	4.67
	W4	1.00	0.00	25.67	52.83	1.00		W4	1.00	0.00	42.17	54.50	1.00
	W5	3.25	0.00	5.50	65.50	1.00		W5	8.00	0.00	14.67	63.00	1.33
	W6	13.50	0.00	13.00	14.00	4.33		W6	20.75	0.00	24.33	17.17	4.67
	W7	9.09	0.00	26.17	13.50	4.67		W7	9.25	0.00	55.00	13.50	4.67
	Avg.	8.13	0.00	16.93	33.78	3.10		Avg.	12.06	0.00	32.06	34.32	3.60
G2	W1	19.52	32.00	16.67	34.83	3.33	G10	W1	23.57	32.00	25.50	33.17	4.00
	W2	5.50	0.00	5.50	139.83	5.00		W2	13.17	0.00	13.17	121.67	5.00
	W3	21.67	0.00	46.50	21.67	5.00		W3	21.67	0.00	56.00	21.67	5.00
	W4	1.00	0.00	32.50	112.50	1.00		W4	1.00	0.00	58.67	116.50	1.00
	W5	3.25	0.00	5.50	124.17	1.00		W5	7.25	0.00	13.17	114.33	1.33
	W6	21.34	1.52	19.67	23.00	5.00		W6	29.67	24.00	30.00	29.33	3.67
	W7	13.34	0.00	40.67	21.67	5.00		W7	13.50	5.00	67.17	21.67	5.00
	Avg.	12.23	4.79	23.86	68.24	3.62		Avg.	15.69	8.71	37.67	65.48	3.43
G3	W1	24.32	52.57	12.50	56.67	5.33	G11	W1	26.98	55.00	28.00	39.50	5.60
	W2	5.00	0.00	5.00	229.17	6.00		W2	16.17	4.00	16.17	160.83	5.00
	W3	26.50	0.00	58.17	26.50	6.00		W3	29.50	27.00	83.17	29.50	6.67
	W4	1.00	0.00	34.50	175.50	1.00		W4	1.00	0.00	73.17	214.83	1.00
	W5	3.00	0.00	5.00	184.83	1.00		W5	8.83	0.00	16.33	164.67	1.33
	W6	29.00	44.57	16.67	41.33	7.00		W6	33.08	53.00	28.00	38.17	6.33
	W7	18.50	23.43	57.67	32.00	5.00		W7	23.92	57.00	64.83	41.50	6.00
	Avg.	15.33	17.22	27.07	106.57	4.48		Avg.	18.00	28.00	44.24	98.43	4.56
G4	W1	41.48	76.14	22.17	96.00	8.00	G12	W1	50.68	80.00	36.00	103.80	7.30
	W2	5.50	0.00	5.50	423.50	11.67		W2	17.33	51.00	17.33	374.67	7.33
	W3	56.33	44.38	77.50	56.33	9.67		W3	89.17	69.00	114.17	89.17	10.00
	W4	1.00	0.00	58.50	415.17	1.00		W4	1.00	0.00	110.50	404.50	1.00
	W5	3.25	0.00	5.50	376.50	1.00		W5	10.17	55.00	18.33	342.33	2.00
	W6	59.00	74.54	16.67	101.33	9.67		W6	69.75	79.00	45.50	94.00	7.67
	W7	32.58	55.41	76.00	57.83	7.33		W7	42.50	68.00	117.17	75.67	9.00
	Avg.	28.45	35.78	37.41	218.09	6.91		Avg.	40.09	57.43	65.57	211.88	6.33
G5	W1	52.53	81.60	26.33	125.00	9.00	G13	W1	72.93	85.00	53.50	146.16	11.33
	W2	5.50	0.00	5.50	748.83	12.67		W2	16.17	57.00	16.17	650.50	11.33
	W3	91.83	57.20	90.00	91.83	15.00		W3	123.50	71.00	138.00	123.50	13.67
	W4	1.00	0.00	56.50	750.50	1.00		W4	1.00	0.00	136.50	702.50	1.00
	W5	3.25	0.00	5.50	744.17	1.00		W5	11.17	64.00	19.33	575.50	3.00
	W6	99.75	82.80	29.50	170.00	13.67		W6	92.25	83.00	59.17	125.33	13.67
	W7	57.00	70.71	80.50	104.67	9.33		W7	41.00	58.00	138.33	69.33	12.67
	Avg.	44.41	41.76	41.98	390.71	8.81		Avg.	51.14	59.71	80.14	341.84	9.52
G6	W1	92.00	87.13	31.17	250.50	6.33	G14	W1	124.45	89.34	100.83	239.67	10.00
	W2	5.50	15.15	5.50	1271.17	13.67		W2	23.00	77.44	23.00	1086.17	12.33
	W3	130.83	64.73	144.33	130.83	19.00		W3	171.33	75.62	206.67	171.33	17.67
	W4	1.00	0.00	63.17	1189.83	1.00		W4	1.00	0.00	165.67	1094.50	1.00
	W5	5.42	12.82	9.17	1287.83	1.67		W5	13.91	77.54	24.50	1114.00	3.33
	W6	153.84	85.86	19.50	288.17	9.67		W6	154.42	86.53	122.67	186.17	18.00
	W7	66.58	67.96	119.50	118.83	14.33		W7	93.33	80.14	201.33	172.00	14.67
	Avg.	65.02	47.66	56.05	648.17	9.38		Avg.	83.04	69.52	120.67	580.55	11.00
G7	W1	117.95	87.99	50.83	297.33	16.33	G15	W1	150.92	90.00	116.30	297.83	16.30
	W2	5.50	30.30	5.50	1923.50	16.00		W2	18.50	87.00	18.50	1442.33	12.67
	W3	166.83	68.23	155.50	166.83	21.00		W3	166.17	66.00	234.17	166.17	23.67
	W4	1.00	0.00	83.17	1793.17	1.00		W4	1.00	0.00	237.83	1717.83	1.00
	W5	27.83	0.00	35.33	2598.83	20.33		W5	17.58	85.00	30.83	1463.50	4.00
	W6	193.91	88.02	37.33	350.50	14.33		W6	298.58	92.00	126.17	471.00	16.33
	W7	124.92	80.60	141.67	236.83	13.00		W7	107.75	77.00	201.00	203.17	12.33
	Avg.	91.13	50.73	72.76	1052.43	14.57		Avg.	108.64	71.00	137.83	822.26	12.33
G8	W1	155.08	89.41	47.67	426.17	17.00	G16	W1	163.83	89.57	152.00	280.33	18.67
	W2	5.50	24.24	5.50	2430.50	10.33		W2	29.17	89.58	29.17	1923.67	20.00
	W3	266.50	76.41	185.17	266.50	30.33		W3	231.33	77.59	266.00	231.33	26.00
	W4	1.00	0.00	87.17	2486.83	1.00		W4	1.00	0.00	177.33	1897.83	1.00
	W5	3.25	0.00	5.50	2551.83	1.00		W5	19.84	91.50	35.00	1841.33	4.67
	W6	241.09	88.57	63.00	419.17	19.33		W6	204.00	86.63	168.83	239.17	23.67
	W7	152.50	81.18	153.00	289.33	15.67		W7	127.33	79.61	275.83	231.33	23.33
	Avg.	117.85	51.40	78.14	1267.19	13.52		Avg.	110.93	73.50	157.74	949.28	16.76

Shaded rows report the average across all instance groups.

5.4. Parameter Calibration

The local search schema presented in Figure 2 is governed by three parameters: $N_{\max}$, which controls the termination criterion for each neighborhood operator; N_{cycles} , which determines the number of complete intensification–diversification cycles; and the perturbation strength k , which sets the number of operations simultaneously relocated at each diversification step. This subsection reports systematic calibration experiments designed to identify the configuration that best balances solution quality against computation time across the 16 groups of benchmark instances.

5.4.1. Calibration of $N_{\max}$

Table 6 reports the calibration results for seven values of $N_{\max}$ spanning the range [500, 5000], with $N_{\text{cycles}} = 4$ held fixed. The first column lists the tested value of $N_{\max}$. The second column, “Avg. Improvement over MILP (%)”, gives the average percentage gain in objective value achieved by the ILS over the MILP warm-start, computed across all 16 benchmark instance groups. The third column, “Avg. Time per Cycle (s)”, reports the mean time required to complete a single intensification cycle (one full sequential application of the swap and repositioning operators). The fourth column, “Cycles with Late Improvements (%)”, measures the proportion of cycles in which at least one improving move is identified in the second half of the iteration budget (between $N_{\max}/2$ and $N_{\max}$), indicating whether the chosen threshold is large enough to avoid premature termination. The last column, “Efficiency Ratio”, is computed as Improvement/Time and quantifies the marginal solution quality gain obtained per second of computation, providing a direct measure of cost effectiveness across configurations.

Table 6. Impact of $N_{\max}$ on solution quality and computational efficiency.

$N_{\max}$	Avg. Improvement over MILP (%)	Avg. Time per Cycle (s)	Cycles with Late Improv. (%)	Efficiency Ratio
500	21.03	17.24	90.97	1.22
1000	22.18	29.20	91.67	0.76
1500	23.46	42.20	91.67	0.56
2000	22.76	52.87	91.67	0.43
2500	23.96	64.49	91.67	0.37
3000	23.91	72.46	91.67	0.33
5000	24.55	104.23	91.67	0.24

The shaded row indicates the selected configuration ($N_{\max} = 1000$), adopted in all subsequent experiments.

Results exhibit clear diminishing returns as $N_{\max}$ increases. Expanding from 500 to 1000 yields 1.15 percentage points of additional improvement at a 69% increase in computation time, while the further step from 1000 to 5000 adds only 2.37 percentage points at a cost of more than tripling the per-cycle time to 104.23 s. The stability of the late-improvement rate at approximately 91.67% for all $N_{\max} \geq 1000$ confirms that operators avoid premature convergence at this threshold, yet improvements beyond iteration 1000 occur sporadically rather than systematically. The value $N_{\max} = 1000$ captures 90% of the solution quality attained by $N_{\max} = 5000$ while requiring only 28% of the per-cycle computation time, and is therefore adopted for all subsequent experiments.

5.4.2. Calibration of N_{cycles}

Table 7 presents the calibration results for $N_{\text{cycles}} \in \{2, 4, 6, 8, 10\}$, using the calibrated value $N_{\max} = 1000$. The first column lists the number of complete intensification–diversification cycles executed. The second column, “Cumulative Improvement (%)”, gives the average percentage gain in objective value achieved over the MILP warm-start once the specified number of cycles has been completed, computed across all 16 benchmark instance groups. The third column, “Marginal Gain (%)”, quantifies the additional improvement

contributed by the last two cycles relative to the previous cycle pair (e.g., the marginal gain at $N_{\text{cycles}} = 4$ is the difference between the cumulative improvements at $N_{\text{cycles}} = 4$ and $N_{\text{cycles}} = 2$), revealing the diminishing returns behavior of additional cycles. The last column, “Cumulative Time (s)”, reports the total average computation time required to complete the indicated number of cycles, allowing direct assessment of the time–quality trade-off across configurations.

Table 7. Marginal improvement contribution by cycle number.

Cycles	Cumulative Improvement (%)	Marginal Gain (%)	Cumulative Time (s)
2	17.80	2.50	36.70
4	21.42	1.42	71.25
6	23.59	0.87	107.51
8	25.29	0.63	143.10
10	26.18	0.18	177.48

The shaded row indicates the selected configuration ($N_{\text{cycles}} = 4$), adopted in all subsequent experiments.

Marginal gains decay rapidly across successive cycle pairs. The first two cycles deliver 17.80% cumulative improvement in 36.70 s, establishing a strong optimization baseline. Each subsequent pair contributes progressively less: 1.42 percentage points for cycles 3–4, 0.87 for cycles 5–6, and only 0.18 for cycles 9–10 less than one-thirteenth of the gain produced by the first pair. This pattern indicates that the perturbation mechanism effectively escapes local optima during early cycles but encounters increasingly familiar solution regions as the search progresses.

In terms of computational efficiency, the rate of solution quality improvement per unit time can be derived directly from Table 7: cycles 1–4 achieve 21.42% improvement in 71.25 s, corresponding to an average rate of $21.42/71.25 \approx 0.30\%$ per second, whereas cycles 1–10 reach 26.18% improvement in 177.48 s, corresponding to an average rate of $26.18/177.48 \approx 0.15\%$ per second. The progression from $N_{\text{cycles}} = 4$ to $N_{\text{cycles}} = 10$ therefore entails a 50% reduction in efficiency per unit time. Conversely, $N_{\text{cycles}} = 4$ captures approximately $21.42/26.18 \approx 82\%$ of the quality achieved by $N_{\text{cycles}} = 10$ while reducing total computation time by $(177.48 - 71.25)/177.48 \approx 60\%$, and is therefore adopted as the standard configuration.

5.4.3. Calibration of Perturbation Strength

Table 8 reports the calibration results for $k \in \{1, 2, 3, 4, 5\}$, using the previously calibrated values $N_{\text{max}} = 1000$ and $N_{\text{cycles}} = 4$. The first column lists the perturbation strength, expressed as the number of operations simultaneously relocated at each diversification step. The second column, “Escape Rate (%)”, reports the proportion of perturbation events that successfully lead the search to a different local optimum than the one identified before perturbation, providing a direct measure of diversification effectiveness. The third column, “Avg. Degradation (%)”, quantifies the average percentage increase in objective value induced by the perturbation step, computed as $(f^{\text{pert}} - f^{\text{pre}})/f^{\text{pre}} \times 100$, where f^{pre} is the objective value of the local optimum before perturbation and f^{pert} is the objective value immediately after the random relocation of k operations. The fourth column, “Re-opt Time (s)”, gives the average computation time required by the local search operators to reconstruct a local optimum after perturbation, measured from the perturbation event to the next convergence point. The last column, “Net Quality Gain (%)”, captures the overall percentage improvement obtained at the end of the post-perturbation re-optimization relative to the pre-perturbation local optimum, providing the most relevant indicator of the perturbation strategy’s effectiveness in escaping local optima while preserving solution quality. All values are averaged across the 16 benchmark instance groups.

Table 8. Impact of perturbation strength on escape capability and solution quality.

Perturbation Strength	Escape Rate (%)	Avg. Degradation (%)	Re-Opt Time (s)	Net Quality Gain (%)
1-swap	97.92	16.49	13.89	19.22
2-swap	100.00	32.28	17.51	19.20
3-swap	100.00	47.45	16.83	20.63
4-swap	100.00	59.73	18.10	20.35
5-swap	100.00	69.14	18.46	20.01

The shaded row indicates the selected configuration (3-swap perturbation), adopted in all subsequent experiments.

Single operation perturbation (one-swap) provides insufficient diversification: in approximately 2% of cases the search reconverges to the same local optimum, confirming that a single relocation fails to disrupt solution structure adequately. All stronger perturbations ($k \geq 2$) achieve a 100% escape rate. Solution degradation increases monotonically with k from 16.49% for 1-swap to 69.14% for five-swap yet re-optimization time remains stable across all strengths (13.89–18.46 s), indicating that the local search operators efficiently reconstruct solution quality regardless of the initial degree of disruption. The three-swap configuration achieves the highest net quality gain of 20.63%, outperforming both weaker perturbations, which provide insufficient diversification, and stronger ones, which destroy beneficial solution structure faster than the subsequent intensification phase can recover it. Accordingly, three-swap perturbation is adopted as the standard diversification mechanism for all experiments reported in Section 5.5.

5.5. Heuristic Performance and Comparison with MILP

This subsection evaluates the performance of the Iterated Local Search (ILS) against the time-limited MILP model across all 16 instance groups. All experiments use the calibrated settings identified in Section 5.4: $N_{\max} = 1000$, $N_{\text{cycles}} = 4$, and three-swap perturbation, with weight configuration W1 applied uniformly throughout.

5.5.1. ILS Solution Quality

Table 9 reports the ILS objective values alongside the warm-start solution used to initialize each run. The warm-start corresponds to the best feasible solution returned by Gurobi when the MILP model is solved under a reduced time budget of 300 s, rather than the full 3600 s budget used in the comparative experiments of Section 5.2. This shorter budget is deliberately chosen to provide the heuristic with a feasible starting solution within an operationally compatible time scale: in practice, the heuristic must be invoked rapidly enough to support daily planning and re-planning activities (for example, in response to train delays or container reassignments), and a 300 s warm-start ensures that the overall response time of the integrated MILP+ILS pipeline remains well below the time required to solve the MILP to a comparable quality on its own. The warm-start therefore reflects the typical input that the ILS would receive in practice, before any local search refinement is applied. For each instance, the warm-start is computed once and the ILS is then executed ten times to account for the stochastic components of the local search (random swap pairs, random reposition picks, and random three-swap perturbation). Table 9 reports the best and the mean objective values over these ten runs together with the standard deviation; the time column reports the total solution time, computed as the sum of the 300 s warm-start budget and the average local search execution time. For G1 and G9, where the MILP proved optimality within the full 3600 s budget, the warm-start is already optimal and the ILS confirms this value without further modification.

The improvement between warm-start and ILS objective grows substantially with instance size, underscoring the importance of the local search refinement phase. For small instances the warm-start already provides a competitive starting point and ILS improvement

is modest. For medium and large instances, however, the warm-start deteriorates significantly reaching 197.43 for G8 and 250.90 for G16 while the ILS recovers to 98.38 and 127.35 respectively, demonstrating that the intensification–diversification cycles navigate complex solution landscapes effectively even from poor initial configurations. The standard deviation reported in Table 9 remains below 8% of the mean for every group, confirming that the ILS produces stable solutions across random seeds; a paired Wilcoxon signed-rank test on the 16 instance groups with feasible warm-starts further confirms that the improvement of the mean ILS objective over the warm-start is statistically significant ($p < 0.001$).

Table 9. ILS results ($N_{\max} = 1000$, $N_{\text{cycles}} = 4$, W1 weights). Best, mean, and standard deviation over ten ILS runs per instance; component values and total time are reported as group averages.

Group	C	Warm-Start	Best	Mean	Std	RS Dist.	Crane Dist.	Sprd. Chg.	Total Time (s)
G1	10	11.05	11.05	11.05	0.00	9.50	18.33	3.67	316.94
G2	15	18.83	17.42	17.85	0.30	15.00	32.67	4.00	318.83
G3	20	24.97	19.42	20.85	0.95	12.83	42.00	5.33	322.84
G4	30	49.55	36.07	37.10	0.80	17.33	86.00	8.00	336.67
G5	40	79.05	44.85	50.20	4.05	17.00	111.17	15.00	360.32
G6	50	104.87	66.57	70.10	3.50	25.83	170.17	13.00	372.70
G7	60	146.40	84.78	91.20	4.40	29.00	223.83	15.67	390.85
G8	70	197.43	98.38	111.40	8.10	28.67	267.50	19.00	412.03
G9	10	17.48	17.48	17.48	0.00	17.50	26.00	4.67	312.87
G10	15	25.00	23.88	23.92	0.10	22.83	39.33	3.33	316.25
G11	20	32.12	28.02	28.20	0.40	23.67	50.17	5.67	321.38
G12	30	63.77	44.67	47.85	2.75	30.17	94.83	5.67	333.86
G13	40	108.55	56.62	60.30	3.45	43.17	109.67	10.67	348.87
G14	50	206.02	90.00	97.50	5.70	49.00	209.00	14.00	368.36
G15	60	250.58	96.50	108.20	5.50	57.67	214.00	17.33	393.51
G16	70	250.90	127.35	146.80	10.70	59.50	314.00	17.00	405.82

5.5.2. Comparison with the MILP Model

Table 10 presents a direct performance comparison between the ILS and the MILP solved to its full 3600 s limit. The columns of the table are organised as follows. The first two columns identify the instance group and its container volume |C|. The third column, “MILP Obj.”, reports the average objective value obtained by the MILP within the 3600 s time limit. The fourth column, “Gap (%)”, indicates the average optimality gap reported by Gurobi at the end of the run. The fifth column, “ILS Obj.”, reports the average objective value produced by the proposed Iterated Local Search. The last two columns quantify the comparative performance: “Improvement (%)” measures the relative quality gain of the ILS over the MILP and is computed as $(f_{\text{MILP}} - f_{\text{ILS}}) / f_{\text{MILP}} \times 100$, where positive values indicate ILS superiority; “Speedup” reports the ratio between the MILP time limit (3600 s) and the total ILS solution time, computed as the 300-s warm-start budget plus the average local-search execution time.

Figure 5 illustrates the three-level comparison MILP, warm-start, and ILS across all instance groups. The blue arrows connect the MILP bar to the ILS bar, with the annotated percentage quantifying the improvement (red arrows mark the two cases where the ILS is outperformed). Positive improvements are displayed in blue; the two negative cases (G10 and G11) are highlighted in red.

Averaged across all 16 instance groups, the ILS achieves a 16.62% improvement over the time-limited MILP while requiring only about 360 s of total solution time (the 300-s warm-start plus an average of 54.7 s of local search), corresponding to roughly a ten-fold reduction in computation time relative to the 3600 s solver limit. Single destination instances (G1–G8) yield an 18.86% average improvement, while G9–G16 demonstrate a 14.39% average improvement, with speedups remaining close to $10\times$ across both configurations. The slightly lower figure for G9–G16 is attributable entirely to two underperforming instances

(G10, G11) discussed below, not to any systematic difficulty with dual destination configurations: groups G13–G16 achieve improvements of 22.27–36.06%, among the strongest results in the benchmark.

Table 10. ILS versus MILP: solution quality and computation time.

Group	C	MILP Obj.	Gap (%)	ILS Obj.	Improvement (%)	Speedup
G1	10	11.05	0.00	11.05	+0.00	11.4×
G2	15	19.52	32.00	17.42	+10.76	11.3×
G3	20	24.32	52.57	19.42	+20.15	11.2×
G4	30	41.48	76.14	36.07	+13.05	10.7×
G5	40	52.53	81.60	44.85	+14.62	10.0×
G6	50	92.00	87.13	66.57	+27.64	9.7×
G7	60	117.95	87.99	84.78	+28.12	9.2×
G8	70	155.08	89.41	98.38	+36.56	8.7×
G9	10	17.48	0.00	17.48	+0.00	11.5×
G10	15	23.57	32.00	23.88	−1.33	11.4×
G11	20	26.98	55.00	28.02	−3.84	11.2×
G12	30	50.68	78.14	44.67	+11.87	10.8×
G13	40	72.93	85.00	56.62	+22.37	10.3×
G14	50	124.45	89.34	90.00	+27.68	9.8×
G15	60	150.92	90.00	96.50	+36.06	9.1×
G16	70	163.83	89.57	127.35	+22.27	8.9×
Average (All)		71.61	64.34	53.94	+16.62	10.3×
Average (G1–G8)		64.37	63.36	47.32	+18.86	10.3×
Average (G9–G16)		78.86	65.32	60.57	+14.39	10.4×

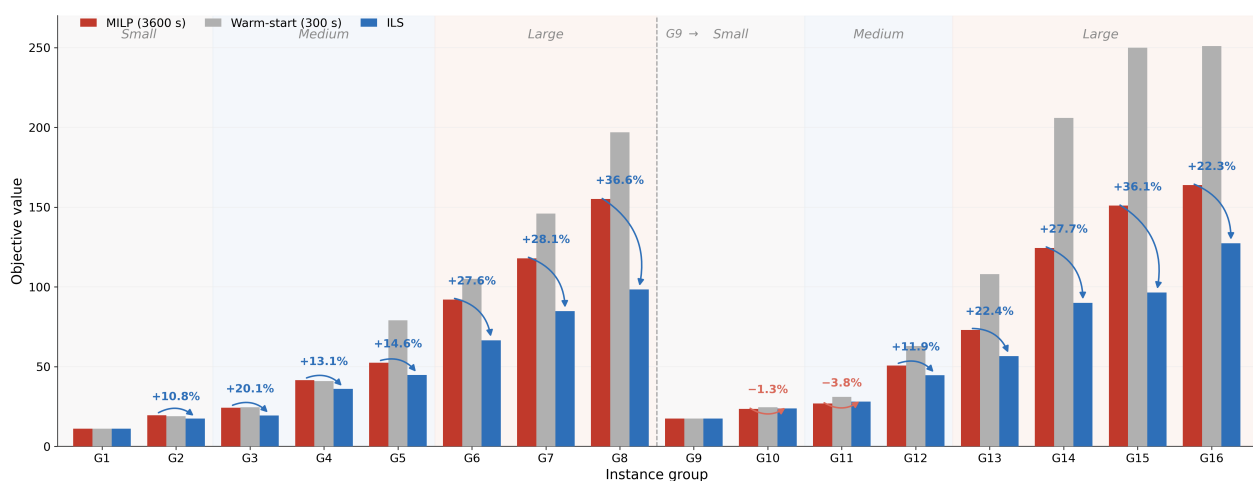


Figure 5. Objective value comparison across MILP, warm-start, and ILS.

The ILS is outperformed by the MILP on G10 (−1.33%) and G11 (−3.84%), both involving 15 and 20 containers respectively. These groups share a common characteristic: their moderate scale allows Gurobi to conduct meaningful branch-and-bound exploration within the full 3600-s budget, yielding structured solutions that the 300-s warm-start cannot match and that the local search finds difficult to surpass through sequence-level modifications alone. Crucially, the magnitude of underperformance is small in both cases, and the ILS still delivers a roughly 11× speedup on both groups, confirming that the degradation reflects a deliberate quality–speed trade-off rather than an algorithmic deficiency.

The ILS advantage grows consistently with instance size. For instances with 50 or more containers where the MILP is most severely constrained by its time limit, exhibiting gaps of 87–90% the ILS achieves improvements ranging from 27.64% to 36.56%. The most pronounced gains are observed at G8 (+36.56%) and G15 (+36.06%), both 70-container instances where MILP solutions are nearly uninformative. This pattern confirms the ILS as the right approach to solve real-size instances.

5.5.3. Component-Level Analysis

Figure 6 decomposes the ILS improvement into its three constituent cost components RS travel, crane travel, and spreader changes across the small, medium, and large instance size categories. Each box summarizes the distribution of component-level percentage reduction relative to the MILP solution across all instances within the category; positive values indicate that the ILS reduces the component below the MILP value, and negative values indicate an increase (the dashed line marks the zero level, and grey dots denote outliers).

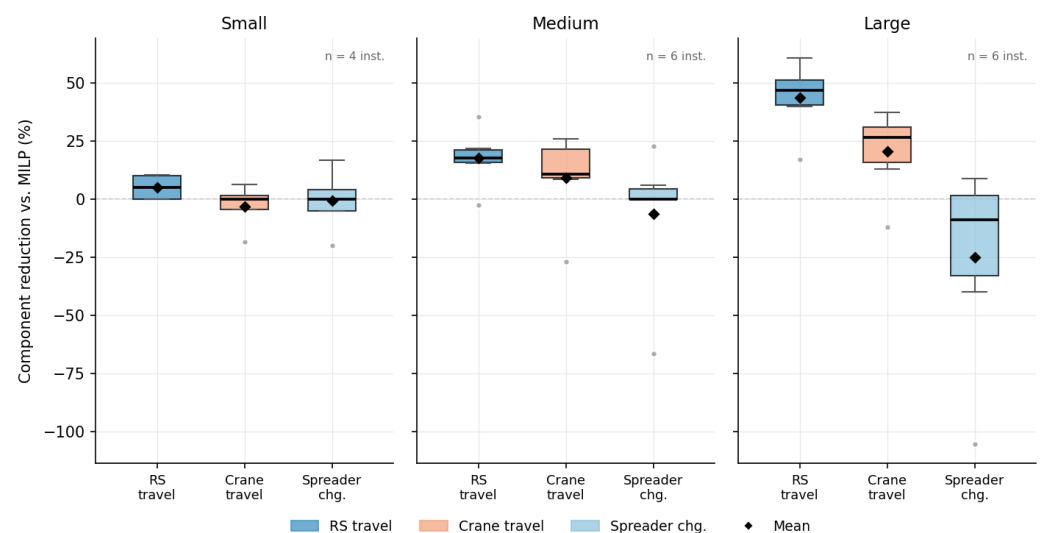


Figure 6. Distribution of component-level reduction achieved by ILS relative to the MILP solution, stratified by instance size category.

The results reveal a clear and interpretable pattern across size categories. For small instances, all three components remain close to zero with low variance, consistent with the limited improvement potential when the MILP already produces near-optimal solutions. For medium instances, RS travel and crane travel both improve, with median reductions of approximately 18% and 13% respectively, while spreader changes exhibit high variance including negative outliers reflecting instances where the ILS accepts increased spreader transitions in exchange for lower equipment travel costs. This substitution pattern intensifies in the large category: RS travel achieves the strongest median reduction of approximately 45% and the tightest distribution, while spreader changes show a clearly negative mean, indicating that the ILS consistently trades spreader efficiency for gains in the two dominant travel components. The stability of RS travel reduction across large instances reflects the effectiveness of the Greedy RS Re-routing sub-procedure, which re-optimizes equipment allocation at every move evaluation and therefore systematically identifies improvements in reach stacker routing that the MILP operating under a saturated time budget fails to realise.

It is worth noting that a fast greedy constructive rule, which unloads containers in near wagon order and assigns each to its nearest compatible bay-locations, can substantially reduce a single objective component but only by displacing cost onto the others. On the benchmark instances, such a rule lowers crane travel by roughly 40% relative to the ILS, yet inflates reach stacker travel by about 95% and spreader changes by about 26%. This confirms that the three objective components are genuinely conflicting and that a coordinated search balancing all of them, rather than a myopic single-component rule, is required to obtain operationally sound plans.

Computation time scales approximately linearly with problem size, ranging from 12.87 s for G9 to 112.03 s for G8, as reported in Table 9. This predictable scaling behavior, combined with a maximum runtime under two minutes across the entire benchmark, confirms that the ILS is well suited to operational planning contexts also for solving situations requiring rapid or repeated solution generation, for example, in response to dynamic terminal conditions such as train delays, container reassignments, or equipment availability changes.

5.5.4. Ablation Study

To assess the individual contribution of each algorithmic component, we conducted an ablation analysis on the 16 groups of benchmark instances. Five variants of the ILS were compared, all starting from the same 300 s MILP warm-start and using the calibrated parameters $N_{\max} = 1000$, $N_{\text{cycles}} = 4$, and three-swap perturbation: Full, the complete configuration; no-GRR, without the Greedy RS Re-routing phase; no-Rep, without the repositioning operator; no-Swap, without the swap operator; and no-Pert, without the three-swap perturbation (i.e. a single intensification cycle). Table 11 reports the average objective value per instance group together with the percentage gap of each ablated variant relative to the full configuration.

Table 11. Ablation study: component contributions to ILS performance.

Grp.	C	Full	no-GRR	no-Rep	no-Sw.	no-Pert	ΔGRR (%)	ΔRep (%)	$\Delta\text{Sw.}$ (%)	ΔPert (%)
G1	10	11.05	11.05	11.05	11.05	11.05	+0.00	+0.00	+0.00	+0.00
G2	15	17.42	17.50	17.50	17.50	17.50	+0.46	+0.46	+0.46	+0.46
G3	20	19.42	20.65	21.77	20.30	23.03	+6.33	+12.10	+4.53	+18.59
G4	30	36.07	37.15	44.92	36.50	44.07	+2.99	+24.54	+1.19	+22.18
G5	40	44.85	46.40	67.88	45.65	66.60	+3.46	+51.35	+1.78	+48.49
G6	50	66.57	69.20	90.32	67.40	91.70	+3.95	+35.68	+1.25	+37.75
G7	60	84.78	89.40	123.10	86.20	111.15	+5.45	+45.20	+1.68	+31.10
G8	70	98.38	104.10	153.90	100.25	151.85	+5.81	+56.43	+1.90	+54.35
G9	10	17.48	17.48	17.48	17.48	17.48	+0.00	+0.00	+0.00	+0.00
G10	15	23.88	24.10	24.97	24.00	24.77	+0.92	+4.57	+0.50	+3.73
G11	20	28.02	28.60	29.83	28.30	30.03	+2.07	+6.46	+1.00	+7.17
G12	30	44.67	46.85	55.97	45.30	53.17	+4.88	+25.30	+1.41	+19.03
G13	40	56.62	60.20	77.70	57.45	72.32	+6.32	+37.23	+1.47	+27.73
G14	50	90.00	95.50	143.02	91.85	134.78	+6.11	+58.91	+2.06	+49.76
G15	60	96.50	102.30	154.82	98.20	147.08	+6.01	+60.43	+1.76	+52.41
G16	70	127.35	134.20	171.82	129.40	173.48	+5.38	+34.92	+1.61	+36.22
Mean gap							+3.76	+28.35	+1.41	+25.56

The repositioning operator and the three-swap perturbation are the principal drivers of performance: removing either (no-Rep, no-Pert) degrades the average objective by 28.35% and 25.56% respectively, with the largest losses on the biggest instances, confirming that broad neighborhood exploration and diversification are essential. The Greedy RS Re-routing phase (no-GRR, +3.76%) and the swap operator (no-Swap, +1.41%) provide smaller but consistent refinements, the former growing with instance size as the reach stacker allocation space expands. The full configuration is never outperformed by any ablated variant, confirming that all four components contribute to the framework's effectiveness.

6. Conclusions

This paper addressed the Train Unloading Planning Problem (TUPP), a combinatorial optimization problem arising in the rail process operations of maritime container terminals. The problem requires the joint determination of the container unloading sequence, the assignment of containers to compatible export yard bay-locations, and the coordinated

movement of gantry cranes and reach stackers, while satisfying equipment, capacity, and storage compatibility constraints.

The main contributions of this work are three-fold. First, we introduced the first mixed-integer linear programming formulation that jointly captures unloading sequence decisions, gantry crane movements, reach stacker routing, and yard block compatibility within a single operational model for non-automated maritime container terminals; to the best of our knowledge, no prior work integrates these decisions in a unified framework. Second, we strengthened this formulation through a set of valid inequalities that reduce computation time across medium and large scale instances. Third, to overcome the scalability limits of exact optimization, we developed an Iterated Local Search heuristic combining two complementary neighborhood operators, swap reordering and insertion-based repositioning, each augmented with an embedded greedy RS re-routing sub-procedure, together with a controlled three-swap perturbation mechanism for diversification.

Computational experiments on 16 groups of benchmark instances confirmed both the limits of the exact approach and the effectiveness of the heuristic. The MILP achieves proven optimality only for 10-container instances within a 3600 s time limit, with optimality gaps escalating rapidly beyond this scale and plateauing between 87% and 90% for instances with 50 or more containers. Under the calibrated configuration ($N_{\max} = 1000$, $N_{\text{cycles}} = 4$), the ILS improves solution quality by 16.62% over the time limited MILP while requiring only about 360 s of total solution time (a 300 s warm-start plus 54.7 s of local search on average), a roughly ten fold reduction relative to the 3600 s solver limit, with gains reaching 36.56% on the largest, operationally realistic instances. A component level analysis further shows that the ILS systematically reduces reach stacker travel through the embedded re-routing procedure, as well as crane travel, occasionally at the cost of increased spreader changes, a substitution pattern that reflects the relative magnitudes of the three cost components across different instance sizes.

From a managerial perspective, the framework offers terminal operators a practical decision-support tool that produces a complete operational plan, the unloading sequence, the yard block assignment, and the reach stacker allocation together, in a small fraction of the time required by exact optimization, making it applicable both to daily planning and to re-planning when conditions change, as in the case of train delays, container reassignments, or equipment unavailability. The objective weights ($\alpha_1, \alpha_2, \alpha_3$) further give operators a direct lever for aligning the plan with prevailing priorities: raising α_1 prioritizes in-yard travel when reach stackers are scarce, α_2 favours shorter crane movements when crane availability constrains turnaround, and α_3 limits spreader reconfigurations when these cause appreciable delays. These weights can be calibrated to local cost structures or selected from a small set of pre-tuned profiles according to the operational scenario.

Several directions remain open for future research. From a modeling perspective, the most natural extension is to consider multiple simultaneous train operations, where resource sharing and interference between trains introduce additional coordination challenges. The integration of train loading and unloading within a unified planning horizon also represents a practically relevant generalisation. A further relevant extension concerns the explicit modeling of equipment malfunctions and operational interruptions, such as a temporary crane or reach stacker breakdown during unloading. Incorporating these events would require moving from the current sequence-based representation to a time-indexed or event-based formulation able to capture the duration of an interruption and its propagation to subsequent operations, since a breakdown affects not the order of moves but their timing and resource availability. This added temporal dimension, together with the stochastic nature of failure occurrences, would motivate dedicated reactive or robust re-planning strategies. From a methodological perspective, future work could explore more advanced

operator selection mechanisms, such as reinforcement learning-based adaptive weights, to dynamically balance intensification and diversification based on search history. Extending the strengthened MILP bounds to guide heuristic repair operators is a further avenue worth exploring, in line with the research direction proposed by [27] for related container terminal scheduling problems.

Author Contributions: Conceptualization, D.A., K.B. and C.C.; methodology, D.A., K.B. and C.C.; software, K.B. and C.C.; validation, D.A. and K.B.; formal analysis, K.B.; investigation, D.A., K.B. and C.C.; data curation, K.B.; writing—original draft preparation, K.B.; writing—review and editing, D.A. and K.B.; visualization, K.B.; supervision, D.A., V.A. and C.C.; project administration, V.A. and C.C.; funding acquisition, D.A. and V.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the European Union—NextGenerationEU and by the Ministry of University and Research (MUR), National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.5, project “RAISE—Robotics and AI for Socio-economic Empowerment” (ECS00000035).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the reported results will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

TUPP	Train Unloading Planning Problem
MILP	Mixed-Integer Linear Programming
ILS	Iterated Local Search
GC	Gantry Crane
RS	Reach Stacker
TEU	Twenty-foot Equivalent Unit
VI	Valid Inequality
AGV	Automated Guided Vehicle
RMG	Rail-Mounted Gantry Crane

References

1. United Nations Trade and Development (UNCTAD). *Review of Maritime Transport 2025: Staying the Course in Turbulent Waters*; Stylus Publishing, LLC: Sterling, VA, USA, 2025.
2. Yu, X.; He, C. Collaborative Scheduling of Yard Cranes, External Trucks, and Rail-Mounted Gantry Cranes for Sea–Rail Intermodal Containers Under Port–Railway Separation Mode. *J. Mar. Sci. Eng.* **2025**, *13*, 1109. <https://doi.org/10.3390/jmse13061109>.
3. Dinu, O.; Dragu, V.; Ruscă, F.; Ilie, A.; Oprea, C. Intermodal transport and distribution patterns in ports relationship to hinterland. *IOP Conf. Ser. Mater. Sci. Eng.* **2017**, *227*, 012038. <https://doi.org/10.1088/1757-899X/227/1/012038>.
4. Ambrosino, D.; Xie, H. Optimization approaches for defining storage strategies in maritime container terminals. *Soft Comput.* **2023**, *27*, 4125–4137. <https://doi.org/10.1007/s00500-022-06769-7>.
5. Abu-Aisha, T.; Audy, J.F.; Ouhimmou, M. Toward an efficient sea-rail intermodal transportation system: A systematic literature review. *J. Shipp. Trd.* **2024**, *9*, 23. <https://doi.org/10.1186/s41072-024-00182-z>.
6. Weerasinghe, B.A.; Perera, H.N.; Bai, X. Optimizing container terminal operations: A systematic review of operations research applications. *Marit. Econ. Logist.* **2024**, *26*, 307–341. <https://doi.org/10.1057/s41278-023-00254-0>.
7. Caballini, C.; Pasquale, C.; Sacone, S.; Siri, S. A discrete-time model for optimizing the rail port cycle. *IFAC Proc. Vol.* **2012**, *45*, 83–88. <https://doi.org/10.3182/20120912-3-BG-2031.00016>.

8. Gharehgozli, A.H.; De Koster, R.; Jansen, R. Collaborative solutions for inter terminal transport. *Int. J. Prod. Res.* **2017**, *55*, 6527–6546. <https://doi.org/10.1080/00207543.2016.1262564>.
9. Gharehgozli, A.; Roy, D.; Saini, S.; Van Ommeren, J.K. Loading and unloading trains at the landside of container terminals. *Marit. Econ. Logist.* **2023**, *25*, 549–575. <https://doi.org/10.1057/s41278-022-00219-9>.
10. Yang, Y.; Sun, S.; He, S.; Jiang, Y.; Wang, X.; Yin, H.; Zhu, J. Research on the Multi-Equipment Cooperative Scheduling Method of Sea-Rail Automated Container Terminals under the Loading and Unloading Mode. *J. Mar. Sci. Eng.* **2023**, *11*, 1975. <https://doi.org/10.3390/jmse11101975>.
11. Ezzahra, S.F.; Abdelouahad, B.; Khalifa, M.; Mohammed, Q. Optimizing Port Operations: Synchronization, Collision Avoidance, and Efficient Loading and Unloading Processes. *Int. J. Adv. Comput. Sci. Appl.* **2023**, *14*, 613–620. <https://doi.org/10.14569/IJACSA.2023.0140768>.
12. Xie, Y.; Song, D.P. Optimal planning for container prestaging, discharging, and loading processes at seaport rail terminals with uncertainty. *Transp. Res. Part E Logist. Transp. Rev.* **2018**, *119*, 88–109. <https://doi.org/10.1016/j.tre.2018.09.008>.
13. Xia, T.; Wang, L.; Zhang, Q.; Dong, J.X.; Song, D.P.; Zhu, X. Integrated scheduling of gantry cranes, container trucks and yard cranes in on-dock railway operation areas at multimodal container ports. *IET Intell. Transp. Syst.* **2025**, *19*, e12600. <https://doi.org/10.1049/itr2.12600>.
14. Liu, W.; Wang, L.; Yan, B.; Zhu, X.; Liu, Z. Integrated optimization of dynamic deployment and scheduling for rail-mounted gantry cranes in sea-rail intermodal port with on-dock rails. *Transp. Res. Part E Logist. Transp. Rev.* **2025**, *202*, 104312. <https://doi.org/10.1016/j.tre.2025.104312>.
15. Cullinane, K.; Wilmsmeier, G. The Contribution of the Dry Port Concept to the Extension of Port Life Cycles. In *Handbook of Terminal Planning*; Böse, J.W., Ed.; Operations Research/Computer Science Interfaces Series; Springer: New York, NY, USA, 2011; Volume 49, pp. 359–379. https://doi.org/10.1007/978-1-4419-8408-1_18.
16. Ambrosino, D.; Asta, V.; Crainic, T.G. Optimization challenges and literature overview in the intermodal rail-sea terminal. *Transp. Res. Procedia* **2021**, *52*, 163–170. <https://doi.org/10.1016/j.trpro.2021.01.089>.
17. Liu, W.; Zhu, X.; Wang, L.; Li, S. Flexible yard crane scheduling for mixed railway and road container operations in sea-rail intermodal ports with the sharing storage yard. *Transp. Res. Part E Logist. Transp. Rev.* **2024**, *190*, 103714. <https://doi.org/10.1016/j.tre.2024.103714>.
18. Wang, L.; Zhu, X. Rail Mounted Gantry Crane Scheduling Optimization in Railway Container Terminal Based on Hybrid Handling Mode. *Comput. Intell. Neurosci.* **2014**, *2014*, 682486. <https://doi.org/10.1155/2014/682486>.
19. Yan, B.; Zhu, X.; Wang, L.; Chang, Y. Integrated Scheduling of Rail-Mounted Gantry Cranes, Internal Trucks and Reach Stackers in Railway Operation Area of Container Terminal. *Transp. Res. Rec. J. Transp. Res. Board* **2018**, *2672*, 47–58. <https://doi.org/10.1177/0361198118782754>.
20. Chen, H.; Liu, W.; Oldache, M.; Pervez, A. Research on Train Loading and Unloading Mode and Scheduling Optimization in Automated Container Terminals. *J. Mar. Sci. Eng.* **2024**, *12*, 1415. <https://doi.org/10.3390/jmse12081415>.
21. Kouismi, S.; Benabbou, L.; Sbihi, N.; Penz, B. Scheduling unloading operations in a railway terminal. In Proceedings of the 2016 11th International Conference on Intelligent Systems: Theories and Applications (SITA), Mohammedia, Morocco, 19–20 October 2016; pp. 1–6. <https://doi.org/10.1109/SITA.2016.7772276>.
22. Ambrosino, D.; Xie, H. Machine Learning-Based Optimization Models for Defining Storage Rules in Maritime Container Yards. *Modelling* **2024**, *5*, 1618–1641. <https://doi.org/10.3390/modelling5040085>.
23. Herrmann, F. Perturbation In Local Search For Scheduling. In *Proceedings of the ECMS 2015 Proceedings*; Mladenov, V.M., Georgieva, P., Spasov, G., Petrova, G., Eds.; ECMS: Albena, Bulgaria, 2015; pp. 446–452. <https://doi.org/10.7148/2015-0446>.
24. de C. M. Nogueira, J.P.; Arroyo, J.E.C.; Villadiego, H.M.M.; Gonçalves, L.B. Hybrid GRASP heuristics to solve an unrelated parallel machine scheduling problem with earliness and tardiness penalties. *Electron. Notes Theor. Comput. Sci.* **2014**, *302*, 53–72. <https://doi.org/10.1016/j.entcs.2014.01.020>.
25. Lourenço, H.R.; Martin, O.C.; Stützle, T. Iterated local search. In *Handbook of Metaheuristics*; Glover, F., Kochenberger, G.A., Eds.; Springer: Boston, MA, USA, 2003; pp. 320–353. https://doi.org/10.1007/0-306-48056-5_11.
26. Farrelly, J.; Grimes, D. An Iterated Local Search Approach to the Container Pre- Marshalling Problem. In Proceedings of the 2022 33rd Irish Signals and Systems Conference (ISSC), Cork, Ireland, 9–10 June 2022; pp. 1–6. <https://doi.org/10.1109/ISSC55427.2022.9826140>.
27. Iris, Ç.; Pacino, D.; Ropke, S. Improved formulations and an Adaptive Large Neighborhood Search heuristic for the integrated berth allocation and quay crane assignment problem. *Transp. Res. Part E Logist. Transp. Rev.* **2017**, *105*, 123–147. <https://doi.org/10.1016/j.tre.2017.06.013>.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.