

Article

Modeling and Solution Algorithm for Green Lock Scheduling Problem on Inland Waterways

Ziyun Wu ¹, Bin Ji ^{1,*} and Samson S. Yu ² 

¹ School of Traffic and Transportation Engineering, Central South University, Changsha 410075, China; chole_wzy@csu.edu.cn

² School of Engineering, Deakin University, Waurn Ponds, VIC 3216, Australia; s.yu@ieee.org

* Correspondence: chcijibin@csu.edu.cn; Tel.: +86-195-7312-9461

Abstract: Inland navigation serves as a vital component of transportation, boasting benefits such as ample capacity and minimal energy consumption. However, it also poses challenges related to achieving navigation efficiency and environmental friendliness. Locks, which are essential for inland waterways, often cause ship passage bottlenecks. This paper focuses on a green lock scheduling problem (GLSP), aiming to minimize fuel emissions and maximize navigation efficiency. Considering the realistic constraints, a mixed-integer linear programming model and a large neighborhood search solution algorithm are proposed. From a job shop scheduling perspective, the problem is decomposed into three main components: ship-lockage assignment, ship placement subproblem, and lockage scheduling subproblem coupled with ship speed optimization. A large neighborhood search algorithm based on a decomposition framework (LNSDF) is proposed to tackle the GLSP. In this, the complex lockage scheduling problem is addressed efficiently by mapping it to a network planning problem and applying the critical path method. Numerical experiments substantiate the effectiveness of our proposed model and a heuristic approach was used in solving the GLSPs. In the sensitivity analysis, under three different objective weight assignments, the resulting solutions achieved average effective ship fuel savings of 4.51%, 8.86%, and 2.46%, respectively. This indicates that our green lock scheduling problem considering ship speed optimization can enhance ship passage efficiency while reducing carbon emissions.



Citation: Wu, Z.; Ji, B.; Yu, S.S. Modeling and Solution Algorithm for Green Lock Scheduling Problem on Inland Waterways. *Mathematics* **2024**, *12*, 1192. <https://doi.org/10.3390/math12081192>

Academic Editor: Elias Olivares-Benitez

Received: 8 March 2024

Revised: 8 April 2024

Accepted: 13 April 2024

Published: 16 April 2024



Copyright: © 2024 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) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: sustainable navigation; green lock scheduling; mixed-integer linear programming; speed optimization; discrete optimization

MSC: 90B06

1. Introduction

In the process of ship navigation, there are many bottlenecks, such as channel navigation problems before entering container ports [1,2], dams [3,4], and canals [5,6] in inland waterway navigation. Locks, as important components in inland waterway transportation, are also one of the major bottlenecks in the navigation process of inland waterway transportation. Developed countries such as China, Europe, and North America have commonly used inland waterways, such as the 29 single-chamber locks upstream of the Mississippi River [7] and the 6 locks along Belgium's Albert Canal [8]. China's inland waterways host over 300 locks of various sizes, including large- and medium-sized ones [9]. With the development of inland waterways and the consequent increase in traffic volume, some locks are experiencing capacity constraints, leading to disruptions in waterway operations. Due to the high cost and long construction cycles of inland waterway improvement and new channel crossings, optimizing ship scheduling processes stands out as an effective measure to enhance the efficiency of inland waterway navigation.

To ensure the safe and orderly flow of ships through locks, it is essential to allocate lock chambers for bidirectional ship traffic within the navigation channel. This process, known

as lock scheduling, involves determining ship berthing positions within lock chambers and scheduling the start times for each lock. Traditional ship lock scheduling commonly relies on heuristic rules based on engineers' experience to prioritize ship queues [4,10]. However, such methods lack scientific rigor and comprehensive consideration, especially in situations of heavy ship traffic, making effective scheduling difficult to achieve. Research on lock scheduling can be categorized into single lock scheduling and serial locks scheduling. The study by Passchyn et al. [11] found that the single lock of the parallel chambers scheduling problem for optimizing ship waiting times is strongly NP-hard even without considering the ship placement problem. Verstichel et al. [12] systematically considered the necessary constraints of the ship placement problem and proposed an improved best-fit heuristic two-dimensional packing algorithm to solve the ship placement problem, and the numerical simulation results showed that the heuristic algorithm has a high efficiency. Subsequently, the team utilized branch-and-bound [13] logic-based Benders decomposition [14] to solve the small-scale single-lock scheduling problem with parallel chambers accurately. But, these two accurate algorithms still struggle to solve the large-scale single lock of the parallel chambers scheduling problem effectively. Ji et al. [15] regarded this problem as a coupled problem of the vehicle routing problem with ship placement scheduling and proposed an adaptive large neighborhood search and an improved best-fit algorithm. His study marked the first successful attempt to efficiently solve the large-scale single-lock scheduling problem with parallel chambers.

Inland waterways in China often leverage stepped channelization to enhance navigation conditions, resulting in the wide-spread distribution of serial locks. Studies have demonstrated that the joint scheduling of serial locks can maximize the overall capacity and service levels [16,17]. Compared with the single lock scheduling problem, the ship flow routes in the serial-lock system are more diverse, and the scheduling schemes of adjacent locks affect each other, so the serial-lock scheduling problem has a higher complexity. In China, there are more abundant scheduling studies for the Three Gorges-Gezhou Dam (TGD). Yuan et al. [18] divided the TGD lock scheduling into three subproblems: lock allocation, schedule optimization, and the ship scheduling problem, and proposed a chaotic embedded particle swarm optimization algorithm to solve it. Zhang et al. [3] proposed a multi-objective mathematical model for optimizing lock utilization, the average waiting time, and ship energy consumption in the joint scheduling problem of the TGD. They introduced a multi-objective metaheuristic algorithm (MOMA) based on fuzzy correlation entropy for solving it. Subsequently, Zheng et al. [19] further considered the impact of the Three Gorges ship lift and approach channel on the scheduling of the TGD, and proposed a collaborative adaptive multi-objective algorithm (CAMOA) for solving it based on this foundation. All of the above studies conducted in-depth research on the navigation efficiency and green carbon emission objectives of the scheduling problem of the TGD locks, but only the special staircase lock structure of the TGD was not generalized.

In terms of the general serial-lock scheduling problem (SLSP), Smith et al. [7] developed a discrete event simulation model for the passage of ships through five locks on the upper reaches of the Mississippi River. Prandtstetter et al. [20] introduced a variable neighborhood search approach for optimizing the scheduling of the nine-lock system along the Austrian segment of the Danube River. Passchyn et al. [21] proposed a mixed-integer linear programming model (MILP) with time-indexing and lockage-indexing for the SLSP. They evaluated the efficacy of the CPLEX solution model on a small-scale instance, confirming its efficiency. They analyzed the impact of scheduling rules such as first-come-first-served (FCFS) and shortest lockage service time priority, as well as the expansion and capacity enhancement of locks, on ship passage efficiency. However, the above studies did not consider the ship placement subproblem, instead simplifying this subproblem into a knapsack problem and a one-dimensional packing problem.

Ji et al. [22] addressed this subproblem in SLSP, developing an MILP for the general SLSP from the perspectives of flexible job shop scheduling. They employed Gurobi for the exact solution of the model, and the results indicated that the model constructed

from the job shop scheduling perspective exhibited a higher efficiency. Furthermore, Ji et al. [23] proposed a decomposition algorithm framework for solving the general serial-lock scheduling problem. They introduced a heuristic solving algorithm combining a large neighborhood search and dynamic programming, effectively addressing large-scale serial-lock scheduling problems. Additionally, they analyzed the impact of factors such as ship priorities and uneven lockage capacities.

All of the previously mentioned studies aimed to improve ship crossing efficiency or lock chamber utilization without considering the environmental factors of lock scheduling. In recent years, with the continuous growth of the waterway transport volume and the increasing advocacy of the concept of low-carbon transportation, more and more scholars have focused on the problem of ship speed optimization in shipping. Unlike previous studies that usually regarded the navigation time of ships between water hubs as a parameter, these scholars considered ship speed as a variable and explored the interplay between the efficiency of navigation and the cost of “green” and low-carbon transportation [6,24]. Defryn et al. [25] focused on the passage of a single ship through a single lock and investigated the impact of optimizing the speed of a single ship on other ships from a game theory perspective. Tan et al. [26] and Buchem et al. [27] investigated the joint schedule design and speed optimization problem for an inland waterway service using different approaches, but both focused on only one ship. Passchyn et al. [21] studied the passage of ships through a serial-lock system with a single chamber. They developed a ship-lockage assignment model based on time discretization and analyzed the impact of optimizing ship speeds on the total ship staying time and carbon emissions. Golak et al. [28] focused on ships passing through a serial-lock system, assuming the uninterrupted continuous operation of the locks. They constructed an MILP to optimize the total ship staying time and carbon emissions. It is worth noting that this study assumed the continuous and uninterrupted operation of the locks, with fixed start times for lockage, which may not align with real-world scheduling scenarios. Yang et al. [5] proposed a nonlinear model with two linear approximation methods to optimize the overall cost of ship navigation from the perspective of system optimization, and the experimental results showed the necessity and effectiveness of considering the green objectives for ship navigation strategies. However, the aforementioned studies utilized overly simplified descriptions of chamber capacity, failing to address the actual ship placement issues.

This paper investigates the green lock scheduling problem (GLSP) of ships traveling bidirectionally through serial locks, considering features such as ship placement and ship speed optimization. The objective was to minimize both total fuel emissions and total ship staying time. To address this, an MILP was constructed based on job shop scheduling theory. To efficiently solve large-scale GLSPs, the problem was decomposed into subproblems including lockage scheduling, ship placement, and ship-lockage assignment. A large neighborhood search algorithm based on a decomposition framework (LNSDF) is proposed. An innovative approach was taken to map the complex lockage scheduling subproblem, which considers ship speed optimization, into a network planning problem. The critical path method (CPM) was then employed to solve this mapped problem efficiently. Numerical experiments demonstrated that the MILP proposed in this paper can be accurately solved using CPLEX for small-scale GLSPs. Additionally, large-scale problems can be effectively resolved by the LNSDF. The sensitivity analysis indicates that the GLSP model can ensure ship passage efficiency while effectively reducing ship fuel emissions.

2. Problem Description

The green lock scheduling process studied in this paper is illustrated in Figure 1. Several locks are distributed along the inland waterway, with each chamber being able to both provide passage services for ships traveling upstream and downstream. The properties of each chamber include its length, width, draft, and the operational time for a lockage. An additional empty lockage is required between two consecutive lockages in the same direction within the same chamber. On the waterway, there is a bidirectional flow of

ship traffic distributed across different times and spaces. Ships each have fixed routes to enter or leave the series-lock systems via the main channel or tributaries. The declared information of the ship includes the length, width, locks it needs to pass, draft, tonnage, speed range, and distance to the first lock. In the scheduling process of the GLSP, the lock operator needs to make decisions on three key aspects: the allocation of ships and lockages, the determination of the optimal arrival speed of ships and the optimal opening time of lockages, and the two-dimensional placement of ships in the lock chambers. These decision-making processes are referred to as the ship-lockage allocation problem, the lock scheduling problem, and the ship placement problem, respectively. Factors to be considered throughout the decision-making process include the characteristics of each lock chamber, the declaration information provided by the ships, the two-way ship traffic flow in the channel, and operational constraints. The goal is to develop a lock operation schedule that maximizes lock efficiency and minimizes ship fuel consumption.

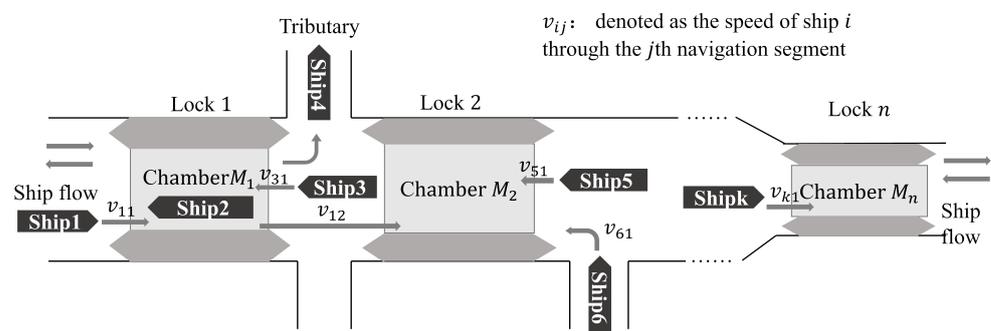


Figure 1. Green lock scheduling diagram.

Therefore, based on these factors, the spatio-temporal constraints, such as ship attributes, lock characteristics, and operational limitations, need to be considered when addressing the GLSP. Among these, the ship placement problem requires the two-dimensional arrangement of ships within the lock chamber to be resolved. Apart from the typical two-dimensional packing constraints, safety mooring constraints for ships within the lock chamber also need to be taken into account. Specifically, as shown in Figure 2a, besides meeting the two-dimensional packing rigidity constraint, any ship must be moored against the inner wall of the chamber or be completely moored alongside a longer ship [12], which will be described by the model constraints in Section 3.

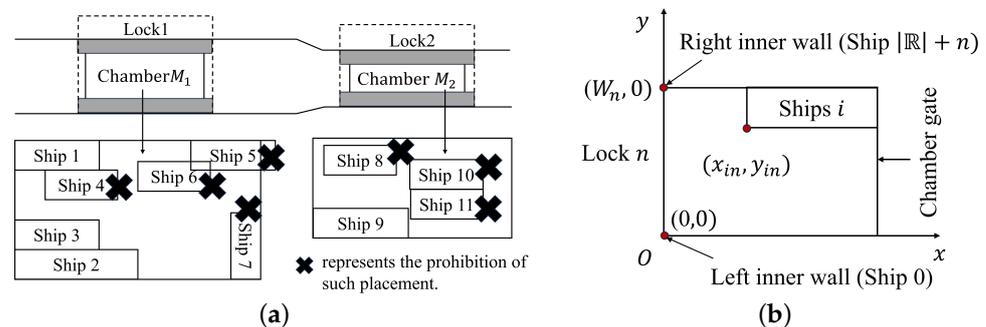


Figure 2. Ship placement diagram. (a) shows the diagram of examples of prohibitions and permissions of ship placement, and (b) shows the diagram of the coordinate.

3. Proposed GLSP Model

In the GLSP, the scheduling of locks is influenced by both adjacent locks and ship speeds. Additionally, ship routes exhibit diverse characteristics, making the problem description complex. This paper maps the problem into a job shop scheduling problem, introducing ship speed optimization features and fuel emission objectives to construct the

GLSP model. Specifically, ships are treated as jobs, locks as machines, and the process of ships passing through multiple locks sequentially is viewed as the processing of jobs across different machines. The ability of a lock to serve multiple ships simultaneously is considered as the machine having the batch processing capability.

The basic assumptions of the model are as follows:

- Safety margins are incorporated within the dimensions of the ships;
- The depth of the chambers meets the draft requirements of all ships;
- The ships navigate at a constant speed within each voyage segment.

Table 1. Symbol description of MILP.

Sets	
\mathbb{N}	Set of locks, indexed by n
\mathbb{R}	Set of ships, indexed by i, j
\mathbb{N}_i	Subset of \mathbb{N} , which represents the locks that ship i passes through. $ \mathbb{N}_i $ represents the number of locks that ship i needs to pass through.
$\mathbb{R}^u, \mathbb{R}^d$	Subset of \mathbb{R} , which represents the upstream and downstream ships, respectively. $\mathbb{R}^u \cup \mathbb{R}^d = \mathbb{R}, \mathbb{R}^u \cap \mathbb{R}^d = \emptyset$
\mathbb{R}_n	Subset of \mathbb{R} , which represents the ships passing through lock n
\mathcal{L}	Set of lockages, indexed by p, q
\mathcal{L}_n	Subset of \mathcal{L} , which represents the lockage processed in lock n
$\mathcal{L}^u, \mathcal{L}^d$	Subset of \mathcal{L} , which represents the upstream and downstream lockages, respectively. $\mathcal{L}^u \cup \mathcal{L}^d = \mathcal{L}, \mathcal{L}^u \cap \mathcal{L}^d = \emptyset$
\vec{I}_i	$\vec{I}_i = (I_{i1}, \dots, I_{ik}, \dots, I_{i \mathbb{N}_i })$, which represents the sequence of locks that ship i needs to pass through, where I_{i1} and $I_{i \mathbb{N}_i }$ is the origin and destination lock of ship i
\mathfrak{A}_{in}	Set of ships that ship i can be allowed to moor at lock n
V_i	The discrete speed set of ship $i, V_i = v_{min}^i, \dots, v_{max}^i$
Parameters	
L_n, W_n	The length and width of chamber in lock n
L, W	The maximum length and width among all chambers
l_i, w_i	The length and width of ship i
r_i	The time when ship i arrives at the serial-lock system
p_n	The time for processing one lockage (lockage processing time) of chamber in lock n
λ_{npq}	The minimum interval time between lockage p and q in lock n when they are processed in the same chamber. If the directions of p and q are opposite, it is 0; otherwise, it is the empty lock processing time of chamber in lock n
P_{max}	The maximum processing time among all lock chambers
C_{max}	The upper bound of the completion time of all ships
D_{in}	Distance that ship i travels from the previous lock (or origin) to lock n
v_{min}^i, v_{max}^i	The minimum and maximum speed of ship i
π_i	The importance weight of ship i
α_i	The fuel consumption coefficient of ship i
Variables	
δ_{inq}	Binary variables, indicating whether ship i is processed by lockage q in lock n (1) or not (0)
a_{in}	Continuous variable, indicating the arrival time of ship i at lock n
C_{in}	Continuous variable, indicating the completion time of lockage q in lock n
P_{nq}	Continuous variable, indicating the processing time of lockage q in lock n
β_{npq}	Binary variables, indicating whether lockage p in lock n is processed before lockage q (1) or not (0)
z_{nq}	Binary variables, indicating whether lockage q in lock n is used (1) or not (0)
c_{in}	Continuous variable, indicating the time when ship i departs lock n
ξ_{ijn}, μ_{ijn}	Binary variables indicating whether ship i is moored to the left and right of ship j in lock n (1) or not (0)
e_{ijn}	Binary variable indicating whether ship i is completely located to the left of ship j in lock n (1) or not (0)
b_{ijn}	Binary variable indicating whether ship i is completely located behind ship j in lock n (1) or not (0)
ω_{ijn}	Binary variable, indicating whether ship i arrives at lock n later than ship j (1) or not (0)
η_{ijn}	Binary variable, indicating whether ship i and ship j are processed by the same lockage in lock n (1) or not (0)
x_{in}, y_{in}	Integer variable, indicating the coordinates (horizontal and vertical) of ship i in lock n , Figure 2b illustrates the location of the specific representation
τ_{inv}	Binary variable, indicating whether ship i arrives at lock n with speed v (1) or not (0)

3.1. Symbol Description

The significance of the symbols involved in the mixed-integer linear programming model (MILP) is as shown in Table 1.

3.2. Optimization Objectives

For the operator of the locks, improving the efficiency of lock operations to ensure ships pass through as quickly as possible is its primary goal. For the ships, they aim to minimize fuel consumption (carbon emissions) during their journey. Therefore, the GLSP considers both ship passage efficiency and ship fuel emissions as two objectives. Specifically, maximizing ship passage efficiency is described by minimizing the total ship staying time in the serial-lock system, as shown in Equation (1).

$$f_1 = \sum_{i \in \mathbb{R}} \pi_i (c_{i|\mathbb{N}_i|} - r_i) \tag{1}$$

In elucidating the correlation between ship speed and fuel consumption, we draw on the derivations and assumptions outlined in [27]. When sailing, a ship encounters resistance R proportional to the square of its speed v^2 . Accordingly, the power required (denoted by P) can be expressed as the product of the resistance, i.e., $P(v) = R(v)v$. Subsequently, from the fact that fuel consumption per unit of time is proportional to the power required, the formula for fuel consumption per unit of ship can be derived as $F(v) = \alpha v^3$, where α represents the ship’s fuel consumption coefficient. Furthermore, considering that the time required for a ship to sail a distance D is $t = D/v$, we can derive the fuel consumption for sailing a distance D as $F(v, D) = \alpha Dv^2$. This result is consistent with the findings of [21] and confirms our analysis. Due to the nonlinear constraints introduced by the speed variable, we discretize the ship’s speed and introduce a binary variable τ_{inv} to linearize the objective as Equation .

$$f_2 = \sum_{i \in \mathbb{R}} \sum_{n \in \mathbb{N}_i} (\alpha_i D_{in} \sum_{v \in V_i} (v^2 \tau_{inv})) \tag{2}$$

In this paper, the problem is transformed into a single-objective optimization problem using weighted coefficients, and the objectives are normalized. This is represented by Equation .

$$\min k_1 \frac{f_1}{F_{min}^1} + k_2 \frac{f_2}{F_{min}^2} \tag{3}$$

3.3. Constraints

The model constraints consist of three components: ship-lockage assignment (Constraints (4)–(7)), lockage scheduling (Constraints (8)–(15)), and ship placement (Constraints (17)–(39)).

Constraints (4)–(7) are the relevant constraints for ship-lockage assignment.

$$\sum_{q \in \mathcal{L}_n^u} \delta_{inq} = 1, \forall n \in \mathbb{N}, i \in \mathbb{R}_n^u \tag{4}$$

$$\sum_{q \in \mathcal{L}_n^d} \delta_{inq} = 1, \forall n \in \mathbb{N}, i \in \mathbb{R}_n^d \tag{5}$$

$$z_{nq} \geq \delta_{inq}, \forall n \in \mathbb{N}, i \in \mathbb{R}_n, q \in \mathcal{L}_n \tag{6}$$

$$z_{nq} \leq \sum_{i \in \mathbb{R}_n} \delta_{inq}, \forall n \in \mathbb{N}, q \in \mathcal{L}_n \tag{7}$$

Constraints (4) and (5) indicate that each ship must be serviced by exactly one lockage at each lock, and the direction of the lockage must be consistent with that of the ship, respectively. Constraints (6) and (7) ensure that each activated lockage serves at least one ship.

Constraints (8)–(15) are constraints related to lockage scheduling.

$$C_{nq} - P_{nq} \geq a_{in} + C_{max} (\delta_{inq} - 1), \forall n \in \mathbb{N}, i \in \mathbb{R}_n, q \in \mathcal{L}_n \tag{8}$$

$$c_{in} \geq C_{max}(\delta_{inq} - 1) + C_{nq}, \forall i \in \mathbb{R}, n \in \mathbb{N}_i, q \in \mathcal{L}_n \tag{9}$$

$$c_{in} \leq C_{max}(1 - \delta_{inq}) + C_{nq}, \forall i \in \mathbb{R}, n \in \mathbb{N}_i, q \in \mathcal{L}_n \tag{10}$$

Constraint (8) indicates that each lockage can only start operating after the ship it serves arrives. Constraints (9) and (10) specify that ships must depart from the lock after the corresponding lockage has finished operating.

$$a_{iI_{i1}} = r_i + D_{iI_{i1}} \sum_{v \in V_i} (\bar{v} \tau_{i,I_{i1},v}), \forall i \in \mathbb{R} \tag{11}$$

$$a_{iI_{ij}} = c_{iI_{i,j-1}} + D_{iI_{ij}} * \sum_{v \in V_i} (\bar{v} \tau_{i,I_{ij},v}), \forall i \in \mathbb{R}, j < |\mathbb{N}_i| \tag{12}$$

$$\sum_{v \in V_i} \tau_{inv} = 1, \forall i \in \mathbb{R}, n \in \mathbb{N}_i \tag{13}$$

Constraints (11) and (12) represent that the arrival time of a ship at the lock is determined by the distance between the ship and the lock and the ship’s speed on that segment, where \bar{v} denotes the reciprocal of the ship’s speed, and $D_{iI_{i1}}$ represents the distance traveled by the ship until reaching the first ship on the route. Constraint (13) states that each ship selects only one average speed for sailing on a segment.

$$C_{nq} - C_{np} + 2C_{max}\beta_{npq} \geq P_{np} + \lambda_{npq}, \forall n \in \mathbb{N}, p, q \in \mathcal{L}_n \tag{14}$$

$$C_{np} - C_{nq} + 2C_{max}(1 - \beta_{npq}) \geq P_{nq} + \lambda_{npq}, \forall n \in \mathbb{N}, p, q \in \mathcal{L}_n \tag{15}$$

$$P_{nq} \geq p_n z_{nq}, \forall n \in \mathbb{N}, q \in \mathcal{L}_n \tag{16}$$

Constraints (14) and (15) represent the temporal sequence relationship between the adjacent lockage within the same lock. The subsequent lockage must wait for the completion of the preceding lockage before it can begin. If the consecutive lockages have the same direction, an empty lockage is required between them. Constraint (16) specifies that the processing time of each lockage is determined by the properties of the chamber where it is assigned.

Constraints (17)–(38) are the constraints related to ship placement, wherein Constraints (17)–(21) are standard two-dimensional constraints used to prevent ship placement from exceeding the dimensions of the lock chamber, and Constraints (22)–(38) are special ship mooring constraints for lock scheduling.

$$\begin{cases} e_{ijn} + e_{jin} + b_{ijn} + b_{jin} + (1 - \delta_{inq}) + (1 - \delta_{jnq}) \geq 1 \\ \forall n \in \mathbb{N}, i, j \in \mathbb{R}_n, q \in \mathcal{L}_n \end{cases} \tag{17}$$

$$x_{in} - x_{jn} + We_{ijn} \leq W - w_i, \forall n \in \mathbb{N}, i, j \in \mathbb{R}_n \tag{18}$$

$$y_{in} - y_{jn} + Le_{ijn} \leq L - l_i, \forall n \in \mathbb{N}, i, j \in \mathbb{R}_n \tag{19}$$

$$x_{in} + w_i \leq W_n + (1 - \delta_{inq})W, \forall n \in \mathbb{N}, q \in \mathcal{L}_n, i \in \mathbb{R}_n \tag{20}$$

$$y_{in} + l_i \leq L_n + (1 - \delta_{inq})W, \forall n \in \mathbb{N}, q \in \mathcal{L}_n, i \in \mathbb{R}_n \tag{21}$$

In the ship placement problem, we first need to consider the constraints of the standard 2D crate, i.e., (1) the ship cannot overlap inside the lock chamber, which is modeled by constraints (17)–(19), and (2) the ship cannot be placed beyond the 2D dimensions of the lock chamber, which is modeled by (20) and (21).

$$y_{jn} - y_{in} \leq (1 - \xi_{ijn})L, \forall n \in \mathbb{N}, i \in \mathbb{R}_n, j \in \mathfrak{X}_{in} \tag{22}$$

$$y_{in} - y_{jn} \leq l_j - l_i + (1 - \xi_{ijn})L, \forall n \in \mathbb{N}, i \in \mathbb{R}_n, j \in \mathfrak{X}_{in} \tag{23}$$

Constraints (22) and (23) indicate that ship i can only be moored to the left of ship j , whose length is greater than ships i in \mathfrak{X}_{in} . The \mathfrak{X}_{in} is the set of ships that can be moored

by ship i in lock n . The ships in \mathfrak{R}_{in} must simultaneously satisfy the following conditions: they are in the same direction as ship i and need to pass through the lock n that ship i needs to pass through.

$$x_{jn} - x_{in} \leq w_i + (1 - \xi_{ijn})W, \forall n \in \mathbb{N}, i \in \mathbb{R}_n, j \in \mathfrak{R}_{in} \tag{24}$$

$$x_{jn} - x_{in} \geq w_i - (1 - \xi_{ijn})W, \forall n \in \mathbb{N}, i \in \mathbb{R}_n, j \in \mathfrak{R}_{in} \tag{25}$$

Constraints (24) and (25) denote another condition for the mooring of ships i and j , i.e., ship j must be adjacent to ship i in lock n .

$$x_{|\mathbb{R}|+n,n} - x_{in} \leq w_i + (1 - \xi_{i,|\mathbb{R}|+n,n})W, \forall n \in \mathbb{N}, i \in \mathbb{R}_n \tag{26}$$

$$x_{|\mathbb{R}|+n,n} - x_{in} \geq w_i - (1 - \xi_{i,|\mathbb{R}|+n,n})W, \forall n \in \mathbb{N}, i \in \mathbb{R}_n \tag{27}$$

A ship may be moored to the left side of the right wall of the lock chamber in addition to the left side of other ships, as indicated by constraints (26) and (27). Specifically, the right inner wall of the lock chamber of lock n is represented by a virtual ship $|\mathbb{R}| + n$, whose position is fixed as $(W_n, 0)$, where $|\mathbb{R}|$ represents the number of ships planned during the planning period.

$$y_{jn} - y_{in} \leq (1 - \mu_{ijn})L, \forall n \in \mathbb{N}, i \in \mathbb{R}_n, j \in \mathfrak{R}_{in} \tag{28}$$

$$y_{in} - y_{jn} \leq l_j - l_i(1 - \mu_{ijn})L, \forall n \in \mathbb{N}, i \in \mathbb{R}_n, j \in \mathfrak{R}_{in} \tag{29}$$

$$x_{jn} - x_{in} \leq -w_i + (1 - \mu_{ijn})W, \forall n \in \mathbb{N}, i \in \mathbb{R}_n, j \in \mathfrak{R}_{in} \tag{30}$$

$$x_{jn} - x_{in} \geq -w_i - (1 - \mu_{ijn})W, \forall n \in \mathbb{N}, i \in \mathbb{R}_n, j \in \mathfrak{R}_{in} \tag{31}$$

$$x_{0n} - x_{in} \leq -w_i + (1 - \mu_{i0n})W, \forall n \in \mathbb{N}, i \in \mathbb{R}_n \tag{32}$$

$$x_{0n} - x_{in} \geq -w_i - (1 - \mu_{i0n})W, \forall n \in \mathbb{N}, i \in \mathbb{R}_n \tag{33}$$

Similarly, when ship i needs to be moored to the right of ship j , the spatial relationship is ensured by Constraints (28)–(31). Constraints (32) and (33) indicate that, besides other ships within the lockage, ship i can also be moored to the right of the left wall of the lock chamber, represented by a virtual ship 0, with its coordinates fixed at $(0,0)$.

$$\sum_{i \in \mathbb{R}_n, j \in \mathfrak{R}_{in}} (\xi_{ijn} + \mu_{ijn}) + \xi_{i,|\mathbb{R}|+n,n} + \mu_{i0n} \geq 1, \forall n \in \mathbb{N}, i \in \mathbb{R}_n \tag{34}$$

$$\xi_{ijn} + \mu_{jin} \leq 1, \forall n \in \mathbb{N}, i \neq j, i, j \in \mathbb{R}_n \tag{35}$$

Constraint (34) ensures that each ship must be moored on one side of either another ship or the lock chamber. When two ships of equal length are present, they are likely to be moored to each other (e.g., ships 10 & 11 in Figure 2a). This violates the safety mooring requirement in practical scheduling, and Constraint (35) is set to prevent this situation.

$$\delta_{jnq} - \delta_{inq} \leq \eta_{ij}, \forall n \in \mathbb{N}, i \neq j, i, j \in \mathbb{R}_n, q \in \mathcal{L}_n \tag{36}$$

$$\delta_{inq} - \delta_{jnq} \leq \eta_{ij}, \forall n \in \mathbb{N}, i \neq j, i, j \in \mathbb{R}_n, q \in \mathcal{L}_n \tag{37}$$

$$\xi_{ijn} + \mu_{ijn} \leq \eta_{ij}, \forall i \in \mathbb{R}_n, n \in \mathbb{N}, j \in \mathfrak{R}_{in} \tag{38}$$

Constraints (36)–(38) indicate that two ships that do not transfer through the same lockage cannot moor to each other.

$$a_{in} - a_{jn} \leq \omega_{ijn}C_{max}, \forall n \in \mathbb{N}, i \neq j, i, j \in \mathbb{R}_n \tag{39}$$

$$a_{in} - a_{jn} \geq (\omega_{ijn} - 1)C_{max}, \forall n \in \mathbb{N}, i \neq j, i, j \in \mathbb{R}_n \tag{40}$$

$$c_{in} - c_{jn} \leq \omega_{ijn}C_{max}, \forall n \in \mathbb{N}, i \neq j, i, j \in \mathbb{R}_n \tag{41}$$

$$c_{in} - c_{jn} \geq (\omega_{ijn} - 1)C_{max}, \forall n \in \mathbb{N}, i \neq j, i, j \in \mathbb{R}_n \tag{42}$$

In practical lock operation scheduling, the FCFS principle may be employed. This means that within the same lock, ships that arrive earlier should depart before ships that arrive later. This is represented by Constraints (39)–(42).

$$0 \leq a_{in} \leq C_{max}, \forall i \in \mathbb{R}, n \in \mathbb{N}_i \tag{43}$$

$$0 \leq c_{in} \leq C_{max}, \forall i \in \mathbb{R}, n \in \mathbb{N}_i \tag{44}$$

$$0 \leq C_{nq} \leq C_{max}, \forall n \in \mathbb{N}, q \in \mathcal{L}_n \tag{45}$$

$$0 \leq P_{nq} \leq P_{max}, \forall n \in \mathbb{N}, q \in \mathcal{L}_n \tag{46}$$

$$x_{in} \in \{0, \dots, W\}, y_{in} \in \{0, \dots, L\}, \forall n \in \mathbb{N}, i \in \mathbb{R}_n \tag{47}$$

$$\delta_{inq} \in \{0, 1\}, \forall i \in \mathbb{R}, n \in \mathbb{N}, q \in \mathcal{L}_n \tag{48}$$

$$\xi_{ijn}, \mu_{ijn}, e_{ijn}, b_{ijn} \in \{0, 1\}, \forall n \in \mathbb{N}, i, j \in \mathbb{R}_n \tag{49}$$

$$\beta_{npq} \in \{0, 1\}, \forall n \in \mathbb{N}, p, q \in \mathcal{L}_n \tag{50}$$

$$\tau_{inv} \in \{0, 1\}, \forall i \in \mathbb{R}, n \in \mathbb{N}_i, v \in V_i \tag{51}$$

Constraints (43)–(51) are type and range constraints for the variables.

$$z_{n,q+1} \leq z_{nq}, \forall n \in \mathbb{N}, q, q + 1 \in \mathcal{L}_n \tag{52}$$

$$C_{n,q+1} \geq C_{nq}, \forall n \in \mathbb{N}, q, q + 1 \in \mathcal{L}_n \tag{53}$$

To speed up the model, Constraints (52) and (53) are added to enforce the order of use and start times for lockages within the same chamber.

4. Solution Method

4.1. Solution Framework

In order to solve large-scale GLSP problems efficiently, we propose a large neighborhood search algorithm based on the decomposition framework (LNSDF). Considering the high complexity of the GLSP, we decomposed the GLSP into an allocation main problem with two subproblems (i.e., the ship-lockage allocation main problem, the ship placement subproblem, and the lockage scheduling problem) based on the problem characteristics. Subsequently, effective solution methods were applied for each problem to realize the interactive solution of the three problems.

As shown in Figure 3, LNSDF is a method that applies a decomposition framework on the basis of LNS, with interactions between various subproblems. The ship-lockage allocation problem, as the main problem, influences the determination of ship placement and lockage scheduling schemes. Therefore, initially, the removal and insertion operators of LNS are employed to reorganize the allocation relationship between ships and lockages, a heuristic that obtains new allocation schemes. Subsequently, the two-dimensional placement of ships in each lockage is carried out, using the well-established Multi-Order Best-Fit (MOBF) method to make decisions on ship placement schemes. At this stage, if MOBF fails to find any feasible placement schemes, the objective function value of the current solution is set to infinity, and the solution will not be adopted. Next, for feasible solutions, decisions are made on ship speed schemes and lockage start times. In this subproblem, two decision variables of the objective function can be obtained: the ship speed and lockage opening schedule (the time when ships leave the lock system), and then the objective function value is calculated, obtaining a feasible new solution.

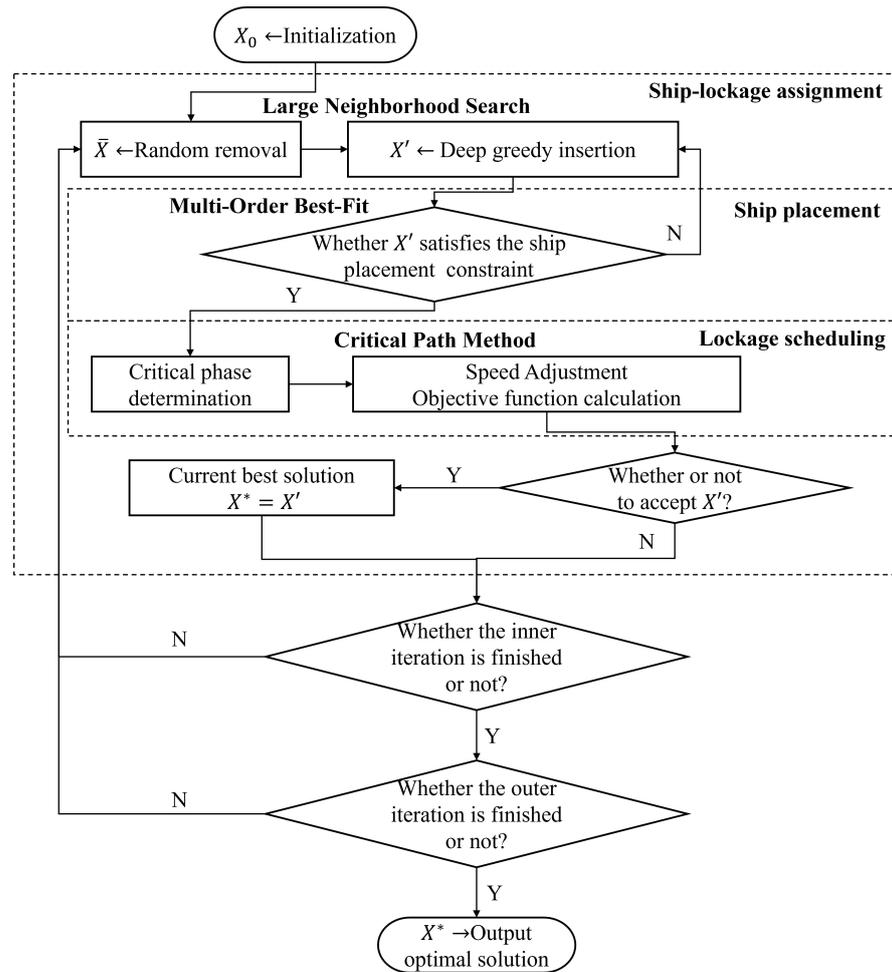


Figure 3. Solution framework for GLSP.

As for the lock scheduling problem, the start time of each lock is not only affected by the timing of other locks, but also by the ship speed. Therefore, the difficulty in solving the lock scheduling subproblem lies in how to optimize the ship speed while obtaining the optimal opening scheme of the locks. To this end, we innovatively map this problem to a network planning scheme problem and use the critical path method (CPM) to solve it.

$$X = \begin{bmatrix} \Omega_1(1) & \dots & \Omega_1(q) & \dots & \Omega_1(\mathcal{L}_1) \\ \vdots & \ddots & \dots & \ddots & \vdots \\ \Omega_n(1) & \dots & \Omega_n(q) & \dots & \Omega_n(\mathcal{L}_n) \\ \vdots & \ddots & \dots & \ddots & \vdots \\ \Omega_N(1) & \dots & \Omega_N(q) & \dots & \Omega_N(\mathcal{L}_N) \end{bmatrix} \tag{54}$$

In LNS, it is crucial to first define the structure of the solution for the problem. For ease of representation, ships will be divided into several ship phases based on the locks they need to pass through. Each phase will be assigned to a lockage in the lock it passes through. This paper designs the solution structure as shown in Equation (54), where $\Omega_n(q)$ represents the set of phases assigned to lock n 's lockage q .

The basic approach to solving the problem is to reconstruct the solution structure using removal and insertion operators to generate feasible solutions. During the removal process, to avoid repetitively removing the same ship passage in iterations, a tabu list is used to store the removed ship passages. These removed ship phases are not removed again for a certain tabu tenure. During the insertion process, the lockage's compliance with ship placement constraints and other relevant spatiotemporal constraints is verified to ensure

the feasibility of the reconstructed solution structure. With the obtained solution structure, the CPM is employed to decide the optimal start time for ships and lockages, taking into account ship speeds. Subsequently, the solutions are evaluated, and the acceptance of these solutions is determined using solution acceptance rules. This iterative process aims to seek an optimized solution and can be described using Algorithm 1.

Algorithm 1: The solution framework of LNSDF

Input: An instance of GLSP.
Output: A solution of GLSP.

- 1 Generating the initial solution X_0 . Calculating its objective function FV_0 (see Equation (3)). $X^* = X_0, FV^* = FV_0$;
- 2 **while** *maximum out-layer iterations not met* **do**
- 3 **while** *maximum in-layer iterations not met* **do**
- 4 Generate a random integer ρ , which indicate the number of ship phases to be removed;
- 5 Perform the remove operator to remove ρ ship phases that are not in the tabu list from X , and obtain a temporary solution \bar{X} ;
- 6 Perform the insertion operator to insert the removed phases back into \bar{X} , obtaining a new solution X' ;
- 7 Perform the CPM to calculate the ship speed and lockage starting time for X' , obtaining FV' ;
- 8 Compare FV' with FV^* and determine whether to accept X' based on the acceptance criterion;
- 9 **if** X' is accepted **then**
- 10 $X^* = X', FV^* = FV'$;
- 11 Update tabu list.;

4.2. Initialization

A heuristic method is used to generate an initial solution. For each lock, a separate lockage is assigned to each ship phase that needs to pass through it. The ship speed is set to its minimum value for each ship. This process results in an initial solution structure where ship speed information is determined.

Treating lockages as nodes and the minimum interval between lockages as edges, the initial solution structure can be represented as a network graph. By considering the temporal relationships between lockages, the shortest path algorithm can be employed to determine the optimal start time for each lockage. This provides the objective function value for the initial solution. Specific descriptions are in Section 4.4.2.

4.3. Removal and Insertion

LNS was first proposed by Shaw and Paul [29] for solving vehicle path problems, and has been widely used in engineering fields. In addition to the vehicle path problem, LNS and its derivative adaptive large neighborhood search (ALNS) have shown their superb global search ability in discrete combinatorial optimization problems such as port berth scheduling [30] and flexible job shop scheduling [31]. Based on the characteristics of the problem, this paper constructs a tabu-based random removal operator and a deep greedy insertion operator to reconstruct the solution structure.

For ease of presentation, we divide the process of navigating a ship between the serial-lock systems into a number of phases, each of which represents only the ship's situation within a particular lock. The removal and insertion operations we perform next are both performed on the ship phases.

4.3.1. Random Removal

To enhance the diversity of the solution structure, a random removal operator is used to randomly remove a certain number (ρ) of ships from the lockages in the current solution structure. To prevent the repetitive and inefficient removal of the same ship phases, a tabu strategy is added. Specifically, there is a 50% chance that the removed phases will be added to the tabu list, and they will be released after a certain number of iterations. After removing a portion of phases from solution X , a temporary solution \bar{X} is obtained.

4.3.2. Deep Greedy Insertion

The deep greedy insertion operator, initially proposed by Ropke et al. [32] to solve vehicle routing problems, is adapted in this paper for the GLSP. The basic idea is to compute the objective function value for all possible insertion positions of the removed phases into the temporary solution \bar{X} , considering all feasible insertion locations. The determination conditions for the feasible positions are as follows: Some of these constraints are implicitly satisfied by the setup of the solution structure (Constraints (6) and (7)). The other part of the constraints requires feasibility determination, such as the operation constraints (Constraints (4) and (5)) and the ship placement constraints.

If the lockage to be inserted is an existing lockage, it is necessary to determine whether all the ships in the lockage satisfy the ship placement constraints (Constraints (17)–(38)). If the insertion violates these constraints, the objective function value is set to infinity. The best insertion position and corresponding objective function value for each removed phase are recorded. Then, phases are inserted into their optimal positions in descending order of the objective function value, resulting in a new solution X' .

The feasibility check for ship placement is achieved through MOBF. This algorithm was initially proposed by Verstichel et al. [33] for solving the ship placement problem and has been proven to have good computational efficiency. The key purpose of the MOBF algorithm is to determine the lock chamber arrangement scheme for the given lockage by sorting the ships based on width, length, or area, and adjusting gap sizes after each placement to maximize the safe and efficient placement of ships within the lock chamber. Algorithm 2 describes the specific steps of the deep greedy insertion operator.

Algorithm 2: Deep Greedy Insertion

Input: A temporary solution \bar{X} , set of phases removed $\bar{\mathbb{R}}$.

Output: A feasible solution X' .

```

1 while  $\bar{\mathbb{R}} \neq \emptyset$  do
2   for  $k \in \bar{\mathbb{R}}$  do
3     Obtain the set  $\Theta_k$  of feasible insertion locations of phase  $k$  for the
       temporary solution  $\bar{X}$ ;
4     Perform MOBF determination and obtain a two-dimensional placement
       scheme for the ship in the lock chamber;
5     Calculate the objective function for each insertion position,  $FV_b(k, p)$ ,
        $\forall p \in \Theta_k$ ;
6     Record the insertion position that minimizes the objective function of
       ship  $i$  as the optimal insertion position,  $\hat{p}_k$ ,
        $\hat{p}_k = \operatorname{argmin}_{p \in \Theta_k} \{FV_b(k, p)\}$ ;
7   Insert phase  $\hat{k} = \operatorname{argmin}_{k \in \bar{\mathbb{R}}} \{\min_{p \in \Theta_k} FV_b(k, p)\}$  into its optimal position  $\hat{p}_k$ ;
8    $\bar{\mathbb{R}} = \bar{\mathbb{R}} - k$ , Updating the solution structure  $\bar{X}$ ;
```

4.4. Lockage Scheduling Subproblem

With sailing speed as a parameter to customize the inputs, lockage scheduling at a serial-lock system can be described as a shortest-path problem, and finding the optimal start time of lockages can be translated into finding the shortest path [23]. However, in the

GLSP, ships' speed is considered a variable, so this approach cannot be directly applied to the GLSP. Therefore, from the perspective of job shop scheduling, each ship's transit through a lock is analogized to a workpiece's processing step, with lockage representing a batch of these processing steps. The fuel consumption incurred during the ship's journey is the resource consumption during the process. As a result, lock scheduling, considering speed optimization, is mapped to the job shop scheduling problem, considering the balance of resource consumption. The problem is described and a network planning diagram is constructed, followed by its resolution using the critical path method (CPM) [34].

The constraints to be considered in the lockage scheduling problem are Constraints (8)–(15).

4.4.1. Ship Speed Decision

The process of speed decision-making involves three steps: determining critical ship phases, adjusting speed, and relaxing time constraints.

1. Determining critical ship phases

In the solution structure X , associations are established between the phases and the lockage. Using the lockages as vertices and the ship sailing and lockage operation processes as arcs, we can derive several paths. Among them, the longest duration path starting from the earliest starting lockage and ending at the latest ending lockage is termed the critical path. The lockages involved in this path are referred to as critical lockages. The latest ship phase to arrive at the critical lockage is termed a critical phase.

As shown in Figure 4, path $Q_{11} \rightarrow Q_{22} \rightarrow Q_{23} \rightarrow Q_{14}$ represents the longest duration critical path, and the latest arrival ship phases $s_{21}, s_{22}, s_{12}, s_{72}$ of critical lockages on this path are termed critical phases. This process is mapped to a network planning problem, and the critical lockages, as well as the critical ship phases that affect the overall lockage efficiency, are determined by using the CPM.

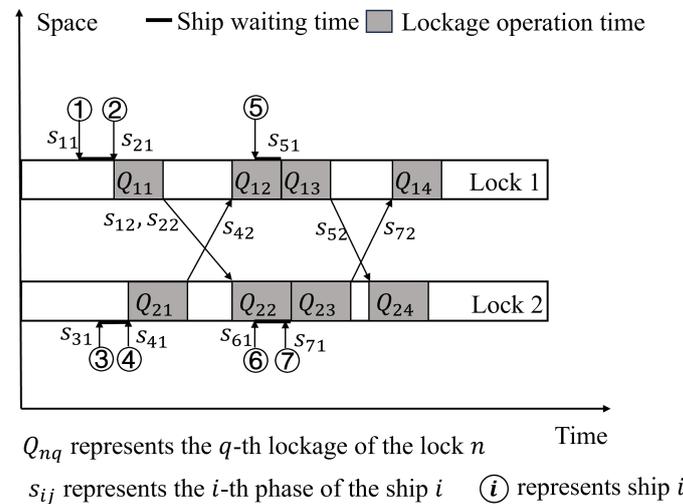


Figure 4. Illustration of critical phase.

The specific steps for determining critical lockages and ship critical phases using the critical path method are as follows.

Step 1 Construct the network graph. Based on the relationship between serviced ships and lockages, a double-labeled network graph is constructed, as seen in Figure 5. Except for the start node (0) and the end node (*), each node represents the end time of each lockage, and the arcs describe the operation process of subsequent nodes representing the lockage. The weight of the arcs represents the duration of the corresponding operation for the lockage. The arcs in the graph include two types: arcs connecting two lockages, which served the consecutive phases of the same ship (A_1), and arcs connecting two consecutive lockages in the same chamber (A_2).

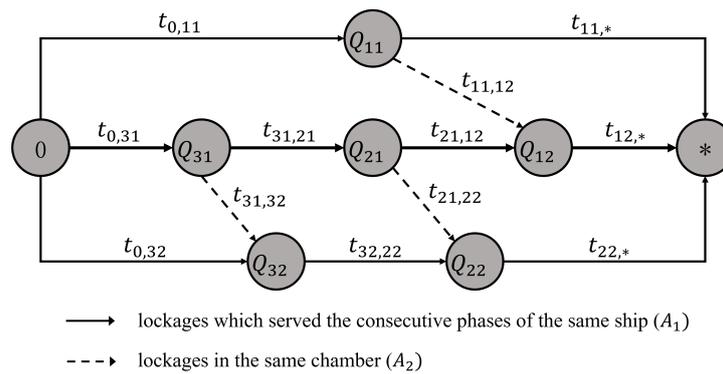


Figure 5. Double-labeled network graph.

Step 2 Compute the weights of each arc as follows. The weight of arcs in set A_1 is the sum of the sailing time and the lockage processing time (as seen in the first and second equation of Equation (55)), and the weight of arcs in set A_2 is the sum of the lockage interval time and the processing time (as seen in the third equation of Equation (55)).

$$t_{mp,nq} = \begin{cases} \max \left(\frac{D_m}{\sum_{v \in V_i} (\bar{v} \tau_{inv})} \right), mp = 0, \forall q \in \mathcal{L}_n, n = I_{i1}, i \in \mathbb{R} \\ \max \left(c_{im} + \frac{D_m}{\sum_{v \in V_i} (\bar{v} \tau_{inv})} \right) + P_{nq}, \forall p \in \mathcal{L}_m, q \in \mathcal{L}_n, m, n \in \mathbb{N}, i \in \mathbb{R} \\ \lambda_{npq} + P_{nq}, m = n, \forall p < q, p, q \in \mathcal{L}_n, n \in \mathbb{N} \\ 0, nq = * \end{cases} \quad (55)$$

where $t_{mp,nq}$ represents the arc weight connecting node Q_{mp} to node Q_{nq} , which denotes the duration of the corresponding operation, c_{im} denotes the departure time of ship i from lock m , λ_{npq} represents the switching time between lockage p and lockage q in lock n , and P_{nq} is the processing time of lockage q in lock n .

Step 3 Use forward pass calculations to determine the earliest start and finish times ES_{nq} for each node in the network graph (see Equation (56)), and employ backward pass calculations to determine the latest start and finish times LF_{nq} for each node in the network graph (see Equation (57)). Let \mathbb{C}_{nq}^F denote the set of predecessors connected to node Q_{nq} , and \mathbb{C}_{nq}^B denote the set of successors connected to node Q_{nq} .

$$ES_{nq} = \begin{cases} \max_{p \in \mathbb{C}_{nq}^B} (ES_{np} + t_{pn,q}), \forall q \in \mathcal{L}_n, n \in \mathbb{N} \\ 0, nq = \{0\} \end{cases} \quad (56)$$

$$LF_{nq} = \begin{cases} \max_{p \in \mathbb{C}_{nq}^F} (LF_{np} - t_{nq,p}), \forall q \in \mathcal{L}_n, n \in \mathbb{N} \\ ES_{nq}, nq = \{*\} \end{cases} \quad (57)$$

Step 4 Determine critical lockages and ship phases. In Step 3, R_{nq} represents the range of time that can be adjusted without affecting the overall duration (see Equation (58)). A lockage with $R_{nq} = 0$ is considered a critical lockage. The phase with the latest arrival time in the critical lockage is identified as the critical phase in that lockage. The path consisting of arcs corresponding to $R_{nq} = 0$ in critical lockages forms the critical path. If the arc corresponding to $R_{nq} = 0$ in a critical lockage belongs to A_2 , then the corresponding critical phase is a virtual phase. The final set of critical phases obtained is denoted as S^* .

$$R_{nq} = LF_{nq} - ES_{nq}, \forall q \in \mathcal{L}_n, n \in \mathbb{N} \quad (58)$$

2. Adjusting Speed

After obtaining S^* through the CPM, adjustments need to be made to the sailing speeds of each critical phase to reduce the completion time of the planning horizon and enhance the overall efficiency of serial-lock system. Clearly, phases with the potential to achieve greater efficiency with fewer resources should be prioritized for speed adjustments. Let us assume that v_{ik} and v_{ik}^+ represent the initial speed and the speed after accelerating once, respectively, for phase k of ship i , $k \in S^*$. The contribution of each phase to the objective function value, denoted by ϕ_k , is calculated based on Equations (59). By evaluating ϕ_k , the sailing speeds of the critical phases are increased successively while ensuring that the critical lockages remain unchanged, until all critical phases reach their maximum speeds or just before the critical lockages change.

$$\phi_k = \frac{f_1(X^{v_i}) - f_1(X^{v_i^+})}{f_2(X^{v_i^+}) - f_2(X^{v_i})} \tag{59}$$

3. Relaxing Time Constraint

After adjusting the sailing speeds of critical phases, it is necessary to examine whether there is any redundant resource usage, i.e., whether ships are waiting. This process is known as time relaxation. Unlike adjusting the sailing speeds, this process does not affect the start times of lockages. As mentioned earlier, the arrival time of critical phases determines the start time of their corresponding lockages. Therefore, non-critical phases may experience some waiting time upon arrival at the corresponding lockage. By reducing the sailing speed of non-critical phases, the waiting time can be converted into the sailing time, reducing fuel consumption.

This process involves traversing all non-critical phases in each lockage. First, non-critical phases with a non-zero waiting time and current speeds not at the minimum speed are identified. Subsequently, the speed of these phases is adjusted to the next lower level. If reducing the speed leads to an increase in the objective function value or renders the current solution infeasible, the reduction is abandoned, and the original sailing speed is retained.

4.4.2. Objective Calculation

The GLSP has two objectives: the total ship's staying time in the lock system and the total ship's fuel consumption, which are determined by two decision variables: the time a ship leaves the last lock ($c_{iI_{|N_i|}}, \forall i \in \mathbb{R}$) and the ship's speed ($\tau_{inv}, \forall i \in \mathbb{R}, \forall n \in \mathbb{N}$), respectively.

In Section 4.4.1, we obtained the ship's speed scenario, which allows us to calculate the ship's total fuel consumption, and the ship's total staying time, which needs to be determined by making a decision on the lockage start time. If a ship's speed is known, the decision of lockage start time for the current solution X structure can be transformed into a shortest path problem. The Bellman–Ford algorithm [35] can be used to solve it.

Specifically, a graph $G = (V, A)$ is used to represent the solution structure, as shown in Figure 6. Here, V represents the set of vertices (i.e., lockages) in the solution structure; A represents the sets of arcs between vertices, where solid lines represent arcs between lockages serving the same ship, and dashed lines represent arcs between lockages in the same chamber but occur sequentially. The weights of the arcs correspond to both the sailing time between two lockages for a ship and the switching time (λ_{npq}) between two lockages. Therefore, deciding the start time for each lockage is equivalent to solving a shortest path problem, which can be achieved using the Bellman–Ford algorithm, thereby obtaining the objective function value of the solution structure.

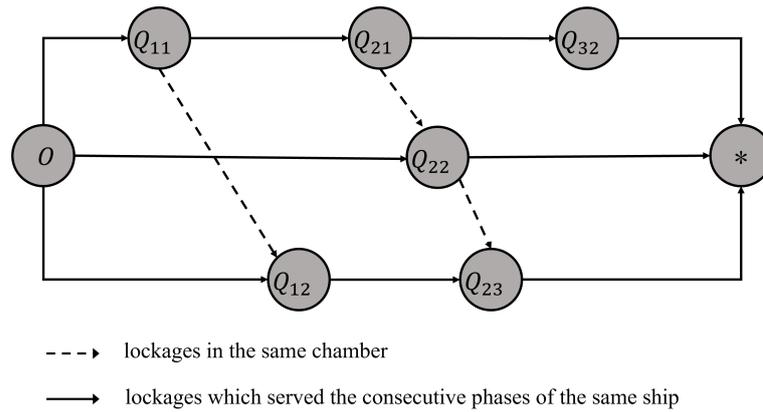


Figure 6. Schematic diagram of the precedence relations of different lockages. *O* represents the origin, and * represents the destination.

4.5. Acceptance Criteria

Each new solution X' obtained after the disruption and repair operations applied to each solution X uses the exponential Monte Carlo acceptance criterion [36] to determine whether it should be accepted. Considering the large-scale optimization capability of the LNS, when $FV(X') < FV(X)$, the new solution will be accepted. Inferior solutions will be accepted with a probability of $e^{-\omega(FV(X')-FV(X))}$, where ω is the coefficient controlling the probability of accepting new solutions that are worse than the current one.

5. Numerical Experiments

5.1. Instance Introduction

5.1.1. First Instance Class

This class of instances utilizes the serial-lock instance proposed by Ji et al. [22] for numerical simulation experiments. The configuration of chambers in the locks is shown in Table 2, while the specific ship data are presented in Table 3. Specifically, there are four locks in total. The number of ships varies between 10 and 20, with an average arrival interval ranging from 5 to 30 min. It is assumed that the ship speed can be decided within the range of [15, 20], and the ship speed decision segment includes the first 10 km before reaching the first lock. There are a total of 16 scenarios comprising different proportions of upstream and downstream ships and varying average arrival intervals for each ship size.

Table 2. Chamber configurations.

Chamber Type	Dimension (m ²)	Processing Time (min)	Switching Time (min)
SC	1600 × 13,600	16	16
LC	2400 × 20,000	16	16

Table 3. Information of the first instance class.

Lock and Ship Flow Configurations	Instance
Number of ships	10, 20
Mean inter-arrival time (min)	5, 10, 15, 30
Upstream and downstream ratio	0.3, 0.5
Range of ship speed (km/h)	[15, 20]
The name of instance	5-10-0.3
Lock configuration	LC-LC-SC-LC
Distance between adjacent locks (km)	(10)-10-18-25

5.1.2. Second Instance Class

This class of instances utilizes the instances proposed by Ji et al. [22]. The lock attributes parameters are the same as in Table 2, with the main difference lying in the number of locks and the average arrival interval of ships, as shown in Table 4. Specifically, there are three locks in total. The number of ships varies between 10 and 20, with the average arrival interval ranging from 1 to 30 min. It is assumed that the ship speed can be decided within the range of [12, 15], and the ship speed decision segment includes the first 10 km before reaching the first lock. For each group of different numbers of ships, 10 instances are generated by changing the upstream and downstream traffic ratios and the average arrival interval, resulting in a total of 20 instances.

Table 4. Information of the second instance class.

Lock and Ship Flow Configurations	Instance
Number of ships	10, 20
Mean inter-arrival time (min)	1, 5, 10, 15, 30
Upstream and downstream ratio	0.3, 0.5
Range of ship speed (km/h)	[12, 15]
The name of instance	1-10-0.3
Lock configuration	LC-SC-LC
Distance between adjacent locks (km)	(10)-12-18

All experiments in this paper were conducted on a computer equipped with an Intel(R) Core (TM) i5-1340P CPU and 16GB of RAM. The MILP and LNSDF were implemented using CPLEX 12.6.3 and C language, respectively.

5.2. Parameter Setting

This paper utilized the Irace software package developed by López-Ibáñez M et al. [37] to determine the parameters and settings of LNS. The basic principle is to provide a set of representative instances and a set of possible values for each parameter. Then, using an iterative racing method, combinations of parameter values are adjusted to approach the optimal solution as closely as possible.

The method involves three steps:

1. Randomly sampling parameters within a specific range, with the probability of selection following a specific distribution;
2. Comparing the sampled parameters and selecting the current optimal configuration based on the performance of the objective function values;
3. Updating the current optimal parameters and repeating the experiments until the stopping conditions are met. For each of the two instance classes, one instance is randomly selected for applying the sampled parameters, with the objective weights set at $k_1 = 0.6, k_2 = 0.4$.

All instances are run 20 times under the same set of parameters, ultimately outputting a set of optimal parameters, as shown in Table 5.

Table 5. Parameters and settings of the LNSDF.

Parameters	Description	Tuned Value
ρ	Removes the degree of destruction in the operator	Random value in the range of $(1, 0.5(\sum_{i \in \mathbb{R}} \mathbb{N}_i - \mathcal{R}_T^1))$
ε	Reaction factor in weight adjustment process	0.1
Ψ	Maximum number of LNSDF iterations (outer iteration)	20
ψ	The maximum number of iterations of inner iteration	$\min(50, \max(\mathbb{R} , 20))$
ω	Index threshold parameter of Monte Carlo acceptance criteria	0.3
ζ	The tabu period of ship phase in the tabu table	5

¹ \mathcal{R}_T represents the phase of the ship in the current tabu table.

5.3. Experiment of GLSPs

This section of the experiment utilizes the first instance class and conducts experiments for four different sets of objective weights: $k_1 = 0.8, k_2 = 0.2$, $k_1 = 0.6, k_2 = 0.4$, $k_1 = 0.4, k_2 = 0.6$, and $k_1 = 0.2, k_2 = 0.8$. The experiments consider the FCFS principle for ships, where the importance weight for all ships ($\pi_i, i \in \mathbb{R}$) is 1, and the ship fuel consumption coefficient ($\alpha_i, i \in \mathbb{R}$) is 1.049.

To facilitate a clearer comparison between the performance of the LNSDF method and the deterministic method, this study contrasts the average objective function values obtained by MILP and LNSDF over 20 computations. Additionally, to provide a more intuitive comparison of their solutions, the concept of relative deviation (RD) is introduced (see Equation (60)), where $FV_{LNSDF}(i)$ represents the objective function value obtained by the LNSDF algorithm in the i -th computation.

$$RD = \left(\sum_{i=1}^{20} FV_{LNSDF}(i) - FV_{MILP} \right) / FV_{MILP} \quad (60)$$

The comparison results between MILP and LNSDF under various lockage configurations are shown in Tables 6 and 7. In all 16 instances, MILP could optimally solve eight small-scale instances, with an average computation time of 3751.04 s. However, as the size of the instances increased, the solving time required by MILP rapidly escalated. When the number of ships exceeded 10, MILP failed to obtain the optimal solution within two hours. In contrast, the proposed algorithm (LNSDF) could achieve high-quality solutions in a shorter time. The average differences between LNSDF and MILP under various objective weights were -0.37% , -0.56% , -0.88% , and -0.83% , respectively. The results of the instances indicate that LNSDF can obtain superior solutions to MILP in a shorter amount of time.

A comparison of the solution obtained using MILP solving with the lower bound of the solution and the LNSDF algorithm is given in Figure 7. First, the results of MILP and LNSDF were similar or equal in some instances. With the four objective weight configurations, MILP was able to optimally solve the algorithm for all 10 ships with a Gap value of 0. With the 20-ship algorithm, the solutions and the lower bounds of MILP and LNSDF were very close to each other in the instances of 5 and 10 min ship intervals, suggesting that both methods are able to find a similar optimal solution in these instances. However, the Gap values of MILP with its lower bound were relatively more significant in the arithmetic case of ship arrival time of 15 vs. 30 min, and the results with LNSDF were more divergent. This was more evident in the objective configuration with weights $k_1 = 0.6, k_2 = 0.4$ and $k_1 = 0.6, k_2 = 0.4$. This may be due to the fact that when the ship arrival time interval is close to or more than 1 unit of lockage operation time, it is more difficult for MILP to balance the two objectives of ship speed and lock crossing efficiency when scheduling lockages. When dealing with complex integer programming problems, the CPLEX branch-and-bound algorithm searches the entire solution space to find the optimal solution. However, due to the NP difficulty of integer programming problems, searching the entire solution space can be very expensive and there may be a large number of branches and constraints in the solution space. This may result in the inability to find a globally optimal solution in a given time, thus producing a significant difference from the lower bound.

In this case, the LNSDF algorithm found a solution similar to the MILP algorithm in a more advantageous solution time, which demonstrates the ability of the LNSDF to efficiently solve larger scale GLSPs. In addition, the lower bound of the solution provides valuable information compared to the solution. If the solution of the MILP or LNSDF algorithm is close to or equal to the lower bound of its corresponding solution, this indicates that the resulting solution may be close to the optimal solution.

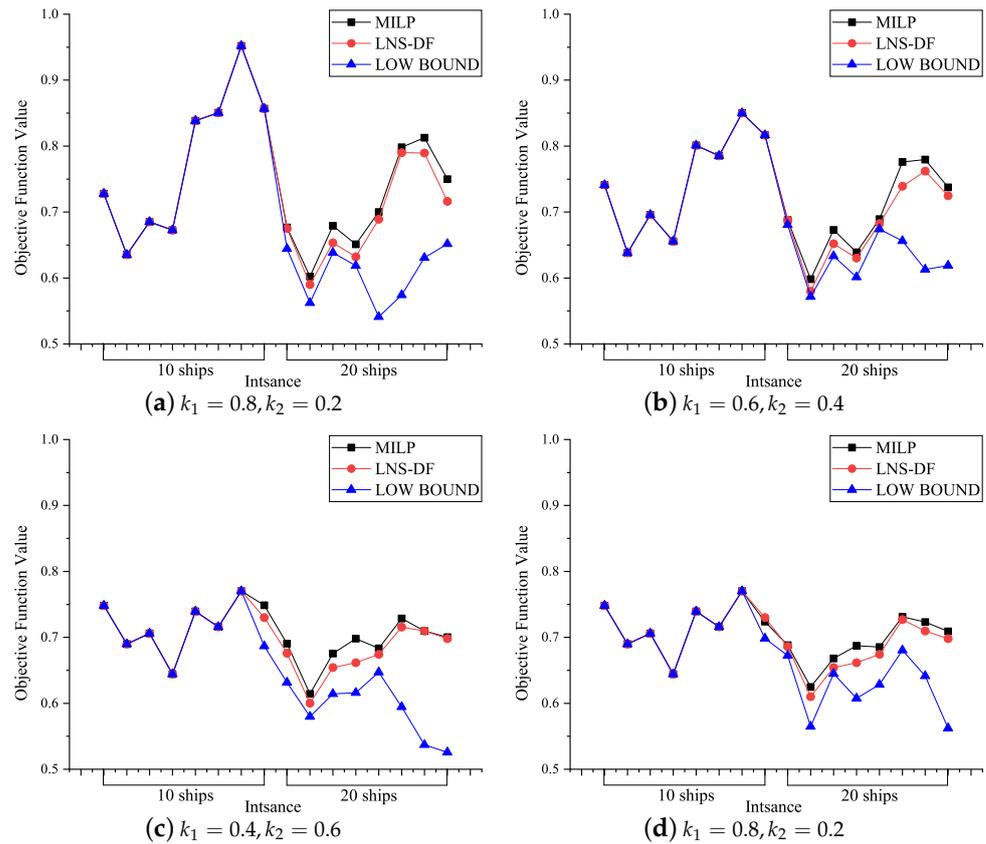


Figure 7. Comparison of upper and lower bounds obtained from CPLEX solution with LNSDF solution.

From Figure 8, it can be observed that, as the weight of the fuel emission objective k_2 increased, the overall fuel consumption of ships gradually decreased, indicating that the scheduling scheme tended to prioritize the low-fuel objective. In practical scheduling processes, the operators of lockage facilities can adjust the weights of different objectives according to different traffic scenarios. When the lockage capacity is insufficient, greater weight can be given to the lockage efficiency objective to ensure ships pass through the lockage as quickly as possible. Conversely, greater weight can be given to the fuel emission objective to minimize ship fuel emissions while ensuring smooth ship passage conditions. Figure 9 gives a schematic diagram of the optimal solution for instance 5_10_0.3, which contains the ship-lockage assignment information, lockage opening time information, ship speed information, and ship placement information, demonstrating that the proposed MILP with the LNSDF is capable of proposing an effective and high-quality solution to the practice GLSP.

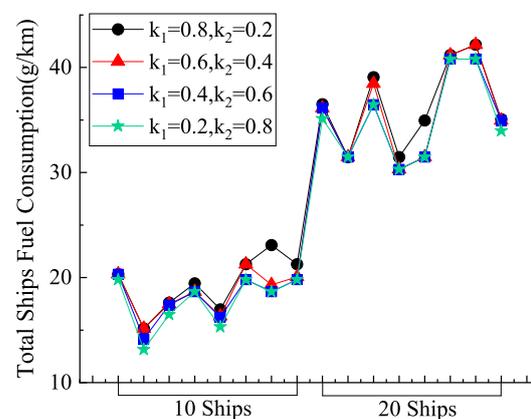


Figure 8. Total fuel consumption of ships with different objective weights.

Table 6. Comparative results of MILP and LNSDF objectives weights of $k_1 = 0.8, k_2 = 0.2, k_1 = 0.6, k_2 = 0.4$.

Weight	$k_1 = 0.8, k_2 = 0.2$						$k_1 = 0.6, k_2 = 0.4$					
	MILP	Gap (%)	Time (s)	LNSDF	Time (s)	RD (%)	MILP	Gap (%)	Time (s)	LNSDF	Time (s)	RD (%)
5_10_0.3	0.7278	0.00	4.18	0.7278	1.81	0.00	0.7410	0.00	5.55	0.7410	1.37	0.00
5_10_0.5	0.6352	0.00	10.32	0.6352	1.46	0.00	0.6381	0.00	8.79	0.6381	1.16	0.00
10_10_0.3	0.6851	0.00	9.91	0.6851	2.04	0.00	0.6959	0.00	7.52	0.6959	1.61	0.00
10_10_0.5	0.6585	0.00	55.44	0.6585	2.46	0.00	0.6698	0.00	70.54	0.6698	2.08	0.00
15_10_0.3	0.8385	0.00	22.52	0.8385	1.98	0.00	0.8010	0.00	26.08	0.8010	1.70	0.00
15_10_0.5	0.8505	0.00	54.67	0.8505	1.79	0.00	0.7852	0.00	35.16	0.7852	1.46	0.00
30_10_0.3	0.9519	0.00	446.62	0.9519	0.80	0.00	0.8500	0.00	246.19	0.8500	0.80	0.00
30_10_0.5	0.8568	0.00	125.87	0.8568	2.05	0.00	0.8165	0.00	148.26	0.8165	1.62	0.00
5_20_0.3	0.6766	5.02	7203.00	0.6751	11.68	-0.22	0.6878	1.08	7205.30	0.6877	14.94	-0.01
5_20_0.5	0.6021	7.11	7204.58	0.5900	12.13	-2.01	0.5982	4.67	7200.28	0.5890	14.69	-1.54
10_20_0.3	0.6790	6.38	7205.80	0.6735	22.13	-0.81	0.6729	26.25	7209.89	0.6620	28.65	-1.62
10_20_0.5	0.6512	5.30	7203.11	0.6487	12.23	-0.38	0.6390	6.29	7205.45	0.6338	15.03	-0.81
15_20_0.3	0.7001	29.42	7203.11	0.6936	27.54	-0.93	0.6893	2.27	7200.00	0.6868	27.45	-0.36
15_20_0.5	0.7982	39.06	7200.31	0.7908	28.41	-0.93	0.7759	18.23	7201.02	0.7711	28.15	-0.62
30_20_0.3	0.8126	28.82	7200.45	0.8095	19.60	-0.38	0.7797	25.23	7204.59	0.7619	18.77	-2.28
30_20_0.5	0.7498	15.05	7200.53	0.7482	27.00	-0.21	0.7374	24.29	7200.91	0.7248	25.32	-1.71
Ave	0.7430	8.51	3646.90	0.7396	10.94	-0.37	0.7236	6.77	3635.97	0.7197	11.55	-0.56

Table 7. Comparative results of MILP and LNSDF objectives weights of $k_1 = 0.4, k_2 = 0.6, k_1 = 0.2, k_2 = 0.8$.

Weight	$k_1 = 0.4, k_2 = 0.6$						$k_1 = 0.2, k_2 = 0.8$					
	MILP	Gap (%)	Time (s)	LNSDF	Time (s)	RD (%)	MILP	Gap (%)	Time (s)	LNSDF	Time (s)	RD (%)
5_10_0.3	0.7483	0.00	5.54	0.7483	1.25	0.00	0.7711	0.00	7.26	0.7711	1.68	0.00
5_10_0.5	0.6898	0.00	4.03	0.6898	1.26	0.00	0.7023	0.00	4.02	0.7023	1.38	0.00
10_10_0.3	0.7059	0.00	32.36	0.7059	2.12	0.00	0.6344	0.00	51.44	0.6344	2.04	0.00
10_10_0.5	0.6572	0.00	63.64	0.6572	2.33	0.00	0.6396	0.00	25.13	0.6396	2.32	0.00
15_10_0.3	0.7395	0.00	23.34	0.7395	1.73	0.00	0.7604	0.00	14.13	0.7604	1.66	0.00
15_10_0.5	0.7158	0.00	32.37	0.7158	1.26	0.00	0.7457	0.00	61.76	0.7457	1.41	0.00
30_10_0.3	0.7103	0.00	126.94	0.7103	0.66	0.00	0.7703	0.00	112.19	0.7703	0.76	0.00
30_10_0.5	0.7487	0.00	544.99	0.7487	1.22	0.00	0.7239	3.66	7202.17	0.7300	1.37	0.84
5_20_0.3	0.6903	9.28	7201.22	0.6861	14.68	-0.61	0.6882	2.37	7201.08	0.6861	16.69	-0.31
5_20_0.5	0.6142	1.93	7202.34	0.6101	15.19	-0.67	0.6247	10.60	7200.53	0.6101	17.25	-2.34
10_20_0.3	0.6754	39.92	7201.86	0.6542	27.04	-3.14	0.6682	2.05	7201.12	0.6542	30.24	-2.10
10_20_0.5	0.6982	13.32	7200.89	0.6615	15.66	-5.26	0.6874	13.13	7203.20	0.6615	16.09	-3.77
15_20_0.3	0.6835	5.64	7200.91	0.6744	26.61	-1.33	0.6853	9.04	7202.05	0.6744	21.12	-1.59
15_20_0.5	0.7287	22.56	7207.78	0.7270	24.67	-0.23	0.7312	7.46	7201.16	0.7270	21.76	-0.57
30_20_0.3	0.7097	32.16	7201.20	0.7097	17.70	0.00	0.7232	12.76	7202.03	0.7097	14.81	-1.87
30_20_0.5	0.7003	33.17	7201.16	0.6980	24.88	-0.33	0.7092	26.11	7201.01	0.6980	20.80	-1.58
Ave	0.7010	9.87	3653.16	0.6960	11.14	-0.88	0.7049	5.45	4068.14	0.6978	10.71	-0.83

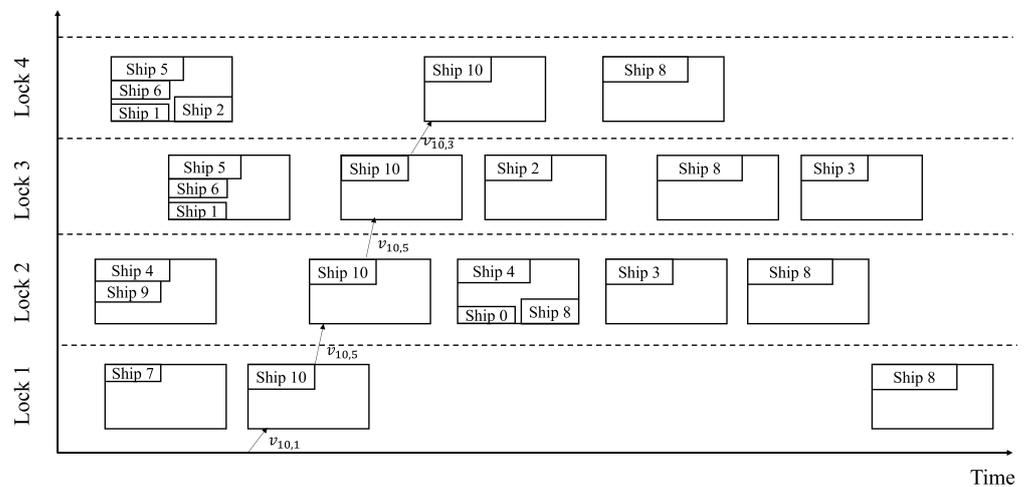


Figure 9. Optimal solution obtained by LNSDF and MILP (5_10_0.3).

5.4. Sensitivity Analysis

This section analyzes the impact of ship importance weights and ship speed on the GLSP without considering the FCFS constraint (i.e., constraints (40)–(43)).

5.4.1. Impact of Ship Importance Weights

This section of the experiment utilized the second instance class conducting experiments under two different sets of objective weights ($k_1 = 0.8, k_2 = 0.2, k_1 = 0.2, k_2 = 0.8$). The importance weight of ships π_i was randomly generated among the values 1, 2, and 3, reflecting the priority of ships in passing through the locks. The ship’s staying time and fuel consumption under each objective weight scenario are illustrated in Figure 10. As shown in Figure 10a,c, in the scenario where $k_1 = 0.8$ and $k_2 = 0.2$, out of the 20 instances, 16 instances had the highest fuel consumption for ships with an importance weight of 3. Additionally, ships with higher importance weights tended to have lower average staying times. This is because, under an emphasis on the efficiency of ship passage through the locks, ships with higher importance weights were prioritized, leading them to travel at higher speeds, thus increasing their carbon emissions during navigation. Figures 10b,d illustrate that, under the scenario where $k_1 = 0.2, k_2 = 0.8$, the priority of ships with an importance weight of 3 was not significantly pronounced. Compared to ships with lower weights, the fuel consumption of ships with an importance weight of 3 was relatively low. This is because, with the increase in k_2 , the scheduling scheme tended to prioritize the low-fuel objectives of ships. In actual scheduling operations, the operators of lock systems can assign different importance weights to different types of ships based on different scenarios. When the lock system is operating at capacity, ships with higher importance weights can be given priority passage.

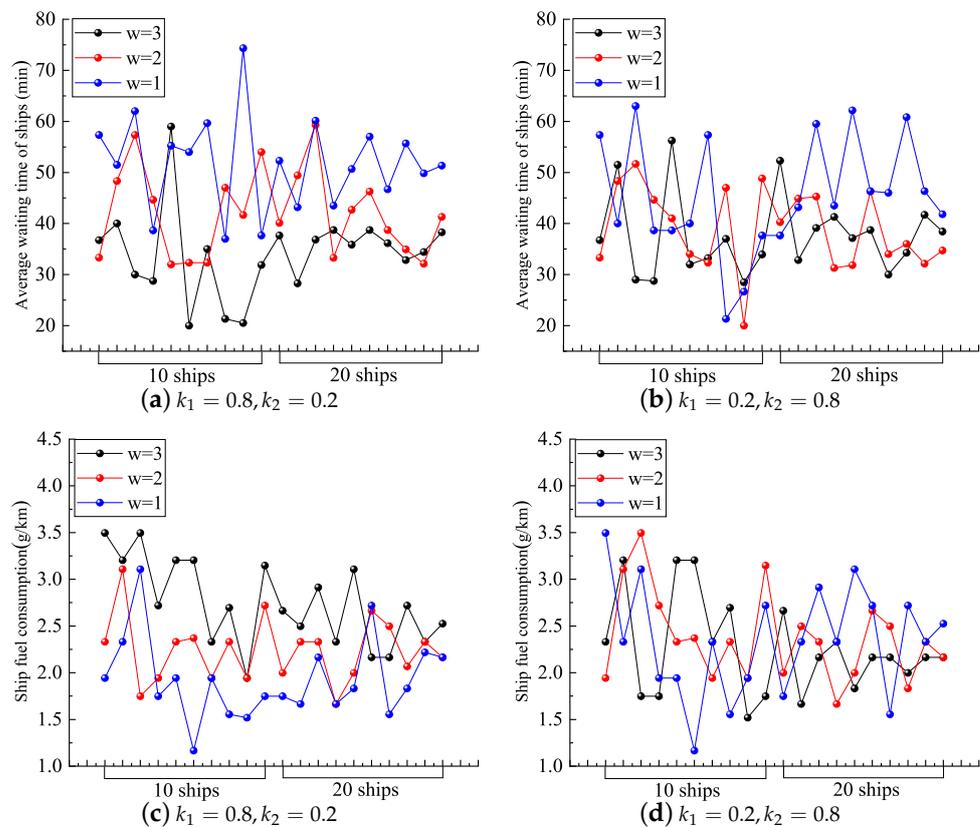


Figure 10. Impact of ship importance weights on dwell time and fuel consumption with different objectives weights.

5.4.2. The Impact of Ship Speed Optimization

In these experiments using the second instance class, we aimed to verify the impact of considering speed optimization on GLSP scheduling solutions. We conducted comparative experiments between LNSDF-NSO (LNSDF without speed optimization, excluding the CPM part) and LNSDF.

To keep the parameters of LNSDF-NSO consistent with those of LNSDF, the navigational distance between all locks was set to 10km, and the decision range of navigational speed was set to 15 km/h to 20 km/h in LNSDF, which corresponds to the navigational time of the ship between 20 min and 30 min. Since LNSDF-NSO does not adjust the sailing speed, the sailing time of ships obeyed a uniform distribution $U(20, 30)$ in LNSDF-NSO.

Then, LNSDF was employed to solve the results of the GLSP under the three objective weighting scenarios: $k_1 = 0.2, k_2 = 0.8$; $k_1 = 0.6, k_2 = 0.4$; $k_1 = 0.8, k_2 = 0.2$.

The effectiveness of ship speed optimization was evaluated using the indicator of fuel savings efficiency θ , as defined in Equation (61). Here, f_1' represents the total time ships spent waiting without considering speed optimization, and f_2' is calculated based on the preset travel times obtained without considering speed optimization. f_1 and f_2 then denote the objective values obtained by considering the speed optimization.

$$\theta\left(\frac{g}{km}\right) = \frac{f_2' - f_2}{f_1 - f_1'} * \frac{\max(f_1, f_1')}{(\max(f_2, f_2'))^2} \tag{61}$$

Table 8 compares the results of LNSDF considering and not considering speed optimization (LNSDF-NSO) under three different objective weight scenarios. The average fuel efficiency savings rates provided by LNSDF for the three solution scenarios were 4.51%, 8.86%, and 2.46%, respectively. This indicates that the ship speed optimization method CPM introduced by LNSDF achieved some effectiveness in reducing ship fuel emissions.

Table 8. Average of LNSDF results for each weight compared to the LNSDF-NSO.

Instance	$k_1 = 0.8, k_2 = 0.2$			$k_1 = 0.6, k_2 = 0.4$			$k_1 = 0.2, k_2 = 0.8$			LNSDF-NSO	
	f_1	f_2	$\theta(\%)$	f_1	f_2	$\theta(\%)$	f_1	f_2	$\theta(\%)$	f_1'	f_2'
1_10_0.3	3477.0	30.0	6.8	3518.0	25.6	9.6	4964.0	25.6	2.4	3212.0	37.1
1_10_0.5	3596.0	35.0	5.8	3654.0	29.1	8.5	5403.0	29.1	2.0	3333.0	42.8
5_10_0.3	3725.0	30.2	7.5	3756.0	26.8	9.2	5288.0	26.8	2.3	3416.0	40.2
5_10_0.5	3478.5	28.3	5.7	3564.3	21.2	11.7	4902.0	21.0	3.2	3239.0	32.4
10_10_0.3	3870.0	35.8	2.0	3989.0	25.6	10.2	4798.0	25.6	3.6	3660.0	37.3
10_10_0.5	3666.0	31.1	4.7	3711.6	23.4	11.0	4910.0	23.3	3.1	3388.0	35.6
15_10_0.3	3973.3	33.9	2.4	4104.3	22.4	11.5	5659.0	22.1	3.1	3731.0	35.8
15_10_0.5	4139.0	30.5	4.5	4244.0	22.4	12.8	5234.0	21.0	4.4	3921.0	33.1
30_10_0.3	4749.0	27.3	1.1	4824.1	19.3	17.5	5289.0	17.5	9.2	4523.0	27.7
30_10_0.5	4891.4	36.9	3.8	4991.0	26.1	11.8	6696.0	24.5	3.1	4622.0	40.3
1_20_0.3	6849.8	45.8	5.7	6961.0	43.1	5.4	9699.0	43.1	1.5	6285.0	66.2
1_20_0.5	6727.0	48.0	4.8	6760.4	43.2	5.9	9555.0	43.1	1.5	6162.0	65.4
5_20_0.3	7895.6	56.4	4.4	7969.7	49.1	5.5	16,304.0	49.0	0.8	7308.0	74.9
5_20_0.5	7512.7	48.2	5.1	7606.0	42.0	6.0	10,413.0	42.0	1.7	6894.0	67.0
10_20_0.3	8303.0	52.0	4.7	8418.0	42.0	6.9	12,684.0	42.0	1.4	7750.0	65.3
10_20_0.5	8198.1	54.7	4.0	8382.6	44.7	5.8	11,478.0	43.1	1.6	7678.0	65.6
15_20_0.3	9056.3	51.4	5.6	9222.0	45.5	5.9	16,301.0	45.5	1.0	8449.0	69.6
15_20_0.5	10,942.7	56.1	5.9	11,024.0	50.1	6.9	19,698.0	50.1	0.9	10,320.0	74.9
30_20_0.3	11,031.6	57.6	4.1	11,143.1	45.8	7.7	18,576.0	45.5	1.1	10,457.0	67.1
30_20_0.5	12,285.1	67.2	1.8	12,662.1	45.6	7.2	19,432.0	45.5	1.3	11,779.0	70.9
Ave	6418.4	42.8	4.5	6525.3	34.7	8.9	9864.2	34.3	2.5	6006.4	52.5

As shown in Table 8, across the three scenarios, the solutions provided by LNSDF achieved ship speed optimization without significantly increasing ship waiting times at the locks, effectively balancing the objectives of ship fuel consumption and waiting time. In the scenario where $k_1 = 0.2$ and $k_2 = 0.8$, the influence of the speed decision operator led to a slight increase in ship waiting times in LNSDF solutions compared to those of LNSDF-NSO. However, the corresponding total fuel consumption of ships decreased. This is because,

even when only considering the efficiency of ship passage as the objective, LNSDF still optimizes ship speeds to some extent, reducing fuel consumption.

In the scenario where $k_1 = 0.6$ and $k_2 = 0.4$, the average fuel savings rate of the obtained solutions was highest among the three scenarios. This is because optimizing both ship waiting times and fuel consumption simultaneously can lead to more environmentally friendly ship passage plans without significantly reducing passage efficiency.

In the scenario where $k_1 = 0.8$ and $k_2 = 0.2$, introducing ship speed optimization methods can effectively reduce ship fuel emissions. In practical scheduling, by adjusting scheduling strategies and objective weights, a balance can be achieved between ship fuel consumption and waiting time objectives. This ensures efficient operations while minimizing ship fuel emissions.

6. Conclusions

This study investigates the green lock scheduling problem (GLSP) in the process of ship passage through a serial-lock system. Considering practical factors such as ship placement, an MILP was constructed with the objectives of maximizing lockage efficiency and minimizing ship navigation fuel emissions. To efficiently solve the large-scale GLSP, the problem was decomposed into ship-lockage assignment, lock scheduling, and ship placement subproblems. Based on this problem, a decomposition framework, i.e., a hybrid heuristic solving algorithm called LNSDF, based on the large neighborhood search and critical path method, was proposed.

Extensive numerical simulations were conducted to validate the effectiveness of the model and algorithm proposed in this study. The results demonstrate that the MILP proposed in this paper can efficiently solve optimal solutions for small-scale problems using CPLEX. Additionally, LNSDF can achieve high-quality scheduling solutions for larger-scale GLSPs in shorter time frames. Furthermore, considering ship fuel emission objectives can better balance lockage efficiency and ship navigation fuel emissions in different traffic scenarios. Sensitivity analysis results indicate that optimizing ship speed can provide more flexible lockage services. When the lock capacity is insufficient, reducing ship speed can distribute the ship waiting time evenly throughout the navigation process, thereby reducing ship navigation fuel emissions. At the same time, for certain important ships, accelerating navigation can ensure a reduction in the waiting time at the serial-locks system, thereby expediting lockage.

This study is also subject to some limitations. Firstly, the discretization of ship speeds may result in suboptimal solutions due to the limited granularity of the decision space. Secondly, the reliance on weighted optimization for addressing both objectives introduces a significant dependency on the predetermined weights, potentially biasing the obtained solutions. To address these limitations, future research could explore alternative approaches. For instance, employing continuous optimization techniques for ship speeds could enable the exploration of a broader solution space, leading to potentially improved scheduling outcomes. Additionally, the investigation of multi-objective optimization algorithms that do not require predefined weights could offer more candidate solutions.

Author Contributions: Conceptualization, Z.W. and B.J.; methodology, Z.W. and B.J.; software, Z.W.; validation, Z.W. and B.J.; formal analysis, Z.W. and B.J.; investigation, Z.W.; resources, B.J.; data curation, Z.W. and B.J.; writing—original draft preparation, Z.W. and S.S.Y.; writing—review and editing, B.J. and S.S.Y.; visualization, Z.W. and S.S.Y.; supervision, B.J.; project administration, B.J.; funding acquisition, Z.W. and B.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China, grant number 72371250, and Central South University Graduate Research Innovation Project under Grant No. 1053320222623.

Data Availability Statement: The datasets generated during and analyzed during the current study are available from the corresponding author upon reasonable request.

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

References

- Jia, S.; Li, C.L.; Xu, Z. Managing navigation channel traffic and anchorage area utilization of a container port. *Transp. Sci.* **2019**, *53*, 728–745. [\[CrossRef\]](#)
- Ji, B.; Huang, H.; Samson, S.Y. An enhanced NSGA-II for solving berth allocation and quay crane assignment problem with stochastic arrival times. *IEEE Trans. Intell. Transp. Syst.* **2022**, *24*, 459–473. [\[CrossRef\]](#)
- Zhang, Y.; Zheng, Q.Q.; He, L.J.; Tian, H.W. Ship traffic optimization method for solving the approach channel and lock co-scheduling problem of the Three Gorges Dam on the Yangzi River. *Ocean Eng.* **2023**, *276*, 114196. [\[CrossRef\]](#)
- Zhang, H.; Ke, J. An Intelligent scheduling system and hybrid optimization algorithm for ship locks of the Three Gorges Hub on the Yangtze River. *Mech. Syst. Signal Process.* **2024**, *208*, 110974. [\[CrossRef\]](#)
- Yang, X.; Gu, W.; Wang, S. Optimal scheduling of vessels passing a waterway bottleneck. *Ocean Coast. Manag.* **2023**, *244*, 106809. [\[CrossRef\]](#)
- Meisel, F.; Fagerholt, K. Scheduling two-way ship traffic for the Kiel Canal: Model, extensions and a matheuristic. *Comput. Oper. Res.* **2019**, *106*, 119–132. [\[CrossRef\]](#)
- Smith, L.D.; Nauss, R.M. Investigating strategic alternatives for improving service in an inland waterway transportation system. *Int. J. Strateg. Decis. Sci. (IJSDS)* **2010**, *1*, 62–81. [\[CrossRef\]](#)
- Verstichel, J.; Vanden Berghe, G. Scheduling Serial Locks: A Green Wave for Waterbound Logistics. In *Sustainable Logistics and Supply Chains*; Lu, M., De Bock, J., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 91–109. [\[CrossRef\]](#)
- Zhao, X.; Lin, Q.; Yu, H. A co-scheduling problem of ship lift and ship lock at the Three Gorges Dam. *IEEE Access* **2020**, *8*, 132893–132910. [\[CrossRef\]](#)
- Wang, X.; Qi, H.; Xiao, H.; Zhang, X.; Hu, Y.; Feng, X. Series queuing network scheduling approach to co-scheduling model of three Gorges-Gezhou dam. *J. Syst. Sci. Complex.* **2010**, *23*, 715–726. [\[CrossRef\]](#)
- Passchyn, W.; Coene, S.; Briskorn, D.; Hurink, J.L.; Spieksma, F.C.; Berghe, G.V. The Lockmaster’s Problem. *Eur. J. Oper. Res.* **2016**, *251*, 432–441. [\[CrossRef\]](#)
- Verstichel, J.; De Causmaecker, P.; Spieksma, F.C.; Berghe, G.V. Exact and Heuristic Methods for Placing Ships in Locks. *Eur. J. Oper. Res.* **2014**, *235*, 387–398. [\[CrossRef\]](#)
- Verstichel, J.; De Causmaecker, P.; Spieksma, F.; Berghe, G.V. The Generalized Lock Scheduling Problem: An Exact Approach. *Transp. Res. Part E Logist. Transp. Rev.* **2014**, *65*, 16–34. [\[CrossRef\]](#)
- Verstichel, J.; Kinable, J.; De Causmaecker, P.; Berghe, G.V. A Combinatorial Benders’ decomposition for the lock scheduling problem. *Comput. Oper. Res.* **2015**, *54*, 117–128. [\[CrossRef\]](#)
- Ji, B.; Yuan, X.; Yuan, Y.; Lei, X.; Iu, H.H. An Adaptive Large Neighborhood Search for Solving Generalized Lock Scheduling Problem: Comparative Study with Exact Methods. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 3344–3356. [\[CrossRef\]](#)
- Liu, C.; Qi, J.; Chu, X.; Zheng, M.; He, W. Cooperative ship formation system and control methods in the ship lock waterway. *Ocean Eng.* **2021**, *226*, 108826. [\[CrossRef\]](#)
- Li, X.; Mou, J.; Chen, L.; Huang, Y.; Chen, P. Ship–Infrastructure Cooperation: Survey on Infrastructure Scheduling for Waterborne Transportation Systems. *J. Mar. Sci. Eng.* **2022**, *11*, 31. [\[CrossRef\]](#)
- Yuan, Y.; Ji, B.; Yuan, X.; Huang, Y. Lockage scheduling of Three Gorges-Gezhouba dams by hybrid of chaotic particle swarm optimization and heuristic-adjusted strategies. *Appl. Math. Comput.* **2015**, *270*, 74–89. [\[CrossRef\]](#)
- Zheng, Q.Q.; Zhang, Y.; Guo, W.J.; Tian, H.W.; He, L.J. Solving energy-efficient lock group co-scheduling problem with ship lift and approach channel using a collaborative adaptive multi-objective algorithm. *Expert Syst. Appl.* **2024**, *242*, 122712. [\[CrossRef\]](#)
- Prandtstetter, M.; Ritzinger, U.; Schmidt, P.; Ruthmair, M. A Variable Neighborhood Search Approach for the Interdependent Lock Scheduling Problem. In *Evolutionary Computation in Combinatorial Optimization*; Ochoa, G., Chicano, F., Eds.; Springer International Publishing: Cham, Switzerland, 2015; Volume 9026, pp. 36–47. [\[CrossRef\]](#)
- Passchyn, W.; Briskorn, D.; Spieksma, F.C. Mathematical Programming Models for Lock Scheduling with an Emission Objective. *Eur. J. Oper. Res.* **2016**, *248*, 802–814. [\[CrossRef\]](#)
- Ji, B.; Zhang, D.; Samson, S.Y.; Fang, X. An Exact Approach to the Generalized Serial-Lock Scheduling Problem from a Flexible Job-Shop Scheduling Perspective. *Comput. Oper. Res.* **2021**, *127*, 105164. [\[CrossRef\]](#)
- Ji, B.; Zhang, D.; Zhang, Z.; Samson, S.Y.; Van Woensel, T. The Generalized Serial-Lock Scheduling Problem on Inland Waterway: A Novel Decomposition-Based Solution Framework and Efficient Heuristic Approach. *Transp. Res. Part E Logist. Transp. Rev.* **2022**, *168*, 102935. [\[CrossRef\]](#)
- Xie, W.; Xu, S.; Zhang, N.; Liu, J.; Yin, K.; Mao, L. Ship Speed Optimization Method in Canal Environments Considering Waiting Times for Crossing Locks. *J. Mar. Sci. Eng.* **2024**, *12*, 375. [\[CrossRef\]](#)
- Defryn, C.; Golak, J.A.P.; Grigoriev, A.; Timmermans, V. Inland Waterway Efficiency through Skipper Collaboration and Joint Speed Optimization. *Eur. J. Oper. Res.* **2021**, *292*, 276–285. [\[CrossRef\]](#)
- Tan, Z.; Wang, Y.; Meng, Q.; Liu, Z. Joint ship schedule design and sailing speed optimization for a single inland shipping service with uncertain dam transit time. *Transp. Sci.* **2018**, *52*, 1570–1588. [\[CrossRef\]](#)
- Buchem, M.; Golak, J.A.P.; Grigoriev, A. Vessel Velocity Decisions in Inland Waterway Transportation under Uncertainty. *Eur. J. Oper. Res.* **2022**, *296*, 669–678. [\[CrossRef\]](#)

28. Golak, J.A.P.; Defryn, C.; Grigoriev, A. Optimizing Fuel Consumption on Inland Waterway Networks: Local Search Heuristic for Lock Scheduling. *Omega* **2022**, *109*, 102580. [[CrossRef](#)]
29. Shaw, P. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In *Principles and Practice of Constraint Programming—CP98*; Goos, G., Hartmanis, J., Van Leeuwen, J., Maher, M., Puget, J.F., Eds.; Springer: Berlin/Heidelberg, Germany, 1998; Volume 1520, pp. 417–431. [[CrossRef](#)]
30. Liu, B.; Li, Z.C.; Sheng, D.; Wang, Y. Integrated planning of berth allocation and vessel sequencing in a seaport with one-way navigation channel. *Transp. Res. Part B Methodol.* **2021**, *143*, 23–47. [[CrossRef](#)]
31. Cao, S.; Li, R.; Gong, W.; Lu, C. Inverse model and adaptive neighborhood search based cooperative optimizer for energy-efficient distributed flexible job shop scheduling. *Swarm Evol. Comput.* **2023**, *83*, 101419. [[CrossRef](#)]
32. Ropke, S.; Pisinger, D. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transp. Sci.* **2006**, *40*, 455–472. [[CrossRef](#)]
33. Verstichel, J.; De Causmaecker, P.; Berghe, G.V. An Improved Best-fit Heuristic for the Orthogonal Strip Packing Problem. *Int. Trans. Oper. Res.* **2013**, *20*, 711–730. [[CrossRef](#)]
34. Bishnoi, N. Critical Path Method (CPM): A Coordinating Tool. *Int. Res. J. Manag. Sci. Technol.* **2018**, *9*, 459–467.
35. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*; MIT Press: Cambridge, MA, USA, 2022.
36. Ayob, M.; Kendall, G. A Monte Carlo Hyper-Heuristic to Optimise Component Placement Sequencing for Multi Head Placement Machine. In Proceedings of the International Conference on Intelligent Technologies, InTech, Halifax, NS, Canada, 13–17 October 2003; Volume 3, pp. 132–141.
37. López-Ibáñez, M.; Dubois-Lacoste, J.; Cáceres, L.P.; Birattari, M.; Stützle, T. The Irace Package: Iterated Racing for Automatic Algorithm Configuration. *Oper. Res. Perspect.* **2016**, *3*, 43–58. [[CrossRef](#)]

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.