Locks are important components of a waterway system, which are used to control water levels. A lock consists of at least one chamber, in which ships can be transferred from one water level to another. Nevertheless, locks sometimes cause bottlenecks for transport over water with the increased traffic volume. For example, in the Three Gorges locks on Yangtze River of China, the passing ships have to wait at least for hours, even for days under certain extreme weather situations [1]. Therefore, to improve the efficiency of inland waterway transport, it is important to ensure the utilization of locks and avoid unnecessary waiting times for ships. Meanwhile, with the trends towards digitalized and smart waterways, it is also worth investigating how the information availability could contribute to optimizing lock operations and ship arrivals on inland waterways. Therefore, this paper proposes an optimization method to schedule ships’ arrivals and their placements in locks on inland waterways, based on a mixed-integer programming model, and solves the optimization problem with large neighborhood search based heuristics. The optimization objective is threefold: first, optimizing the arrival sequence of ships at the locks; second, maximize the utilization of each lockage operation; and third, reducing the overall time that each ship spends from entering the waterway area till leaving the last lock on the waterway. Simulations are carried out to evaluate the performance of the proposed method.

1. Introduction

Locks are important components of a waterway system, which are used to control water levels. A lock consists of at least one chamber, in which ships can be transferred from one water level to another. Nevertheless, locks sometimes cause bottlenecks for transport over water with the increased traffic volume. For example, in the Three Gorges locks on Yangtze River of China, the passing ships have to wait at least for hours, even for days under certain extreme weather situations [1]. Therefore, to improve the efficiency of inland waterway transport, it is important to ensure the utilization of locks and avoid unnecessary waiting of ships. Meanwhile, with the trends towards digitalized and smart waterways and the installation of IT equipment on inland ships, the availability and preciseness of information regarding ships and waterway infrastructure have also been improved in recent years. Thus, it is also worth investigating how the information availability could contribute to solving the lock scheduling problems in waterways.

A lockage includes the ships that are transferred together and their positions in the lock chamber; a lockage operation contains the timing information of ships entering the lock, water level changing, and ships’ departures from the chamber [2]. Depending on the chamber size, one or more ships can be transferred together in one lockage operation. Processing a ship in a lock involves three main decisions, selecting the chamber to transfer the ship, determining a position for the ship in that chamber, and setting a starting time for the lockage operation, which makes lock scheduling an optimization problem. Optimization methods of lockage operations have not been fully explored in the scientific literature, yet it has started to attract more attention in recent years.

Verstichel et al. [3] proposed a mathematical model of ship placement problem in locks to assign positions for a set of ships in different chambers. A fast and exact decomposition algorithm and a multiorder best-fit heuristic are developed to solve the problem. Then, they generalized the lock
scheduling problem as three related subproblems including ship placement, chamber assignment, and lockage operation scheduling based on mixed-integer programming in [2]; a commercial solver is used to solve the problem both in inland locks and locks in a tide independent port. To reduce the computational complexity of the lock scheduling problem, Verstichel et al. [4] proposed a combinatorial Benders’ decomposition approach, which decomposes the lock scheduling problem into a master problem and a subproblem. The master problem assigns the ships to lock chambers and schedules lockages. The subproblem allocates positions to the ships inside chambers.

A mathematical model for lock scheduling of Three Gorges-Gezhouba Dams (TG-GD) is proposed in [1], which consists of lock assignment, timetable optimization, and ship dispatch problems. Then, chaotic embedded particle swarm optimization algorithm is proposed to solve the formulated problem. Yuan et al. [5] studied the coscheduling problem of a single lock and different water-land transshipment docks mixed transportation system for ships passing a dam. A genetic operator-based artificial bee colony (GB-ABC) algorithm is used to determine the ships assigned to locks and the water-land transshipment mode. A first-come-first-served strategy is used to schedule the berths in transshipment docks.

Passchyn et al. [6] studied the lock scheduling problem with a series of consecutive locks and formulated the problem based on mathematical programming, with the objective to minimize total greenhouse gas emissions and the total flow time. Then, they discussed the lockmaster’s problem in [7] and proposed a dynamic programming algorithm to solve the problem in polynomial time. Ji et al. [8] studied the coscheduling problem of cascaded locks with multiple chambers and solved the problem with an iterative hybrid approach which combines quantum-inspired binary gravitational search and modified moth-flame optimization.

Most research studies consider the arrival times of ships before locks as known information and consider the lock scheduling problem from the lock’s perspective, with the aim to maximize the lock capacity to handle more ships in each lockage operation. This paper takes a new perspective by also taking into account the optimization of ships’ arrivals at locks. This could reduce the ships’ waiting times and avoid ship traffic congestions at the locks. Moreover, with the adjustments of ship arrivals, the ships could choose a lower sailing speed in waterways, which could contribute to NOx emission reduction [9]. In practice, shipping companies have already started to adopt the “slow steaming” strategy to reduce fuel usage and overcapacity in recent years [9, 10].

This paper proposes an optimization method to schedule ships’ arrivals and their placements in locks on inland waterways, based on a mixed integer-programming model, and solves the optimization problem with large neighborhood search based heuristics. The optimization objective is threefold: first, optimizing the arrival sequence of ships at the locks; second, maximize the utilization of each lockage operation; and third, reducing the overall time that each ship spends from entering the waterway till leaving the last lock.

The remainder of this paper is organized as follows. Section 2 gives the coordination structure to schedule ship arrivals and lockage operations at locks. Section 3 introduces the model formulation. The optimization steps are given in Section 4. The experimental results are presented in Section 5. Conclusions and future work are given in Section 6.

2. Proposed Coordination Scheme

Figure 1 gives an illustrative example of ships passing a lock on the inland waterway. This paper assumes that there is a controlled waterway area before the lock entrance, in which the positions and speeds of ships could always be tracked. This waterway area is divided into multiple segments; in each segment, appropriate speed and travel time of each ship need to be determined. In this way, the arrivals of ships at locks could be adjusted. To be specific, the lock scheduling problem includes three aspects:

(i) Coordination of ship arrivals: suitable ship sailing speed needs to be determined for each ship so as to coordinate arrival times of different ships at locks to avoid congestion and long waiting times

(ii) Placement of ships in locks: the entering ships need to be placed in the lock in the way that lock capacity is fully utilized

(iii) Planning of lockage operations: the start and finish times of each lockage operation need to be determined, ensuring that the ships could pass the locks with a shorter time

Considering these aspects altogether in one problem makes it difficult to get exact solutions with a reasonable amount of time. Therefore, large neighborhood search (LNS-) based heuristics are proposed in this paper to solve the formulated optimization problem. The LNS metaheuristic introduced by Shaw [11] considers a neighborhood defined implicitly by a destroy and a repair method. A destroy method destructs part of a current solution, while a repair method rebuilds the destroyed solution. The destroy method typically contains an element of stochasticity such that different parts of the solution are destroyed in every iteration of the method [12, 13].

The proposed coordination scheme is given in Figure 2. Each ship sends its current distances from the lock and its planned arrival time at lock to a central coordinator, and each lock sends its earliest available time to the coordinator, upon which the lock scheduling problem is formulated. A large neighborhood search-based heuristic is proposed to solve the formulated problem. After the optimization process terminates, the ships receive information regarding recommended lock arrival times and expected waiting times, as well as recommended ship speed. The start and finish time of the lockage operation as well as the set of ships that are assigned to each lockage and their positions in the lock chamber will also be determined for locks by the coordinator. The time when the last ship in set I leaves the lock will be updated as the earliest available time of the lock for the next set of upcoming ships.
3. Model Formulations

This section first introduces the parameters and decision variables used in this paper. Then, the optimization objectives and constraints are given. Constraints are grouped into five types: constraints on ship arrivals, ship placements, lockage operations, ship priorities, and underoccupied lockages.

3.1. Parameters and Decision Variables. Table 1 gives the list of parameters. In practical situations, weather conditions, currents, and sometimes special reasons (such as document checks) by the authorities may affect the locking schedule and keep the ships waiting or change their order. These factors will be considered in the formulation of constraints. Figure 4 illustrates the minimal safety distances that should be kept between any two ships in the same chamber, as well as between each ship and the lock chamber. Table 2 gives the definitions of the decision variables. Variables \( x_{iq} \) and \( y_{iq} \)

\[
\sum_{k \in K} p_{ikq} = 1 \forall i \in I, \; \forall q \in Q, \tag{2}
\]

\[
t_{iz}^{enter} \geq T_{i, q}^{start} \forall i \in I, \tag{3}
\]

\[
t_{iz+1}^{enter} \geq t_{iz}^{enter} + t_{ij}^{sailing} \forall i \in I, \; \forall z = 1, 2, \ldots, |Z|, \tag{4}
\]

\[
t_{iz}^{arrival} \geq t_{iz}^{enter} + t_{iz}^{sailing} \forall i \in I, \; \forall q \in Q, \tag{5}
\]

\[
t_{iz}^{arrival} \geq t_{iz}^{departure} + \sum_{z' = z+1}^{Z} t_{iz'}^{sailing} \forall i \in I, \; \forall q \geq 2, q \in Q, \tag{6}
\]

\[
t_{iz}^{sailing} \geq T_{i, (n)}^{sailing} v_{in} \forall i \in I, \; \forall z \in Z, \forall n \in N, \tag{7}
\]

\[
-M(1 - p_{ikq}) + t_{ikq}^{lockage} \leq t_{iz}^{departure} \leq t_{ikq}^{lockage} + M(1 - p_{ikq}) \forall i \in I, \; \forall k \in K, \forall q \in Q. \tag{8}
\]

Constraint (2) ensures that all ships will be handled in one of the lockages of each lock. Constraints (3)–(8) describe the relations of ships’ arrivals, processing, and departures at different waterway segments and multiple locks. Constraint (3) defines the start time of each ship when it enters the controlled waterway area. Constraint (4) represents that the enter time of ship \( i \) at segment \( z + 1 \) equals its enter time at previous segment \( z \) plus the sailing time at segment \( z \). Constraint (5) ensures that the arrival time of ship \( i \) at the first lock equals its enter time at the last segment before the
lock entrance plus the sailing time in this segment. It is noted that $Z_1$ represents number of segments before the lock entrance of the first lock. For the subsequent locks, the arrival time at lock $p$ equals the ship’s departure time from the last lock $p-1$ plus the sum of sailing times on the waterway segments from $p-1$ to $p$. This is expressed via constraint (6), in which $Z_q$ represents the number of segments before the entrance of lock $q$. Constraint (7) determines the sailing time of each ship according to its chosen speed. Constraint (8) represents that if ship $i$ is processed in lockage $k$ of lock $q$, its departure time equals the finish time $t_{kq}^{\text{lockage}}$ of lockage $k$ of lock $q$.

#### 3.2.2. Constraints of Ship Placements.

$$\sum_{j \in I} s_{ijq}^{\text{left}} \geq 1 \forall i \in I, \quad \forall q \in Q. \quad (9)$$

$$\sum_{j \in I} s_{ijq}^{\text{behind}} \geq 1 \forall i \in I, \quad \forall q \in Q. \quad (10)$$

$$s_{ijq}^{\text{left}} + s_{ijq}^{\text{behind}} \leq 1 \forall i, j \in I, \quad \forall q \in Q. \quad (11)$$

$$s_{ijq}^{\text{behind}} + s_{ijq}^{\text{behind}} \leq 1 \forall i, j \in I, \quad \forall q \in Q. \quad (12)$$

$$s_{ijq}^{\text{left}} + s_{ijq}^{\text{behind}} + s_{ijq}^{\text{behind}} + (1 - p_{jkq}) + (1 - p_{jkq}) \geq 1 \forall i, \quad j \in I, \forall k \in K, \forall q \in Q. \quad (13)$$

$$D_{\text{horizontal}} \leq x_{ijq} \leq W_q - D_{\text{horizontal}}, \forall i \in I, \quad \forall q \in Q. \quad (14)$$

$$C_{\text{vertical}} \leq y_{ijq} \leq L_q \forall i \in I, \quad \forall q \in Q. \quad (15)$$

$$x_{ijq} + \text{width}_i \leq W_q - D_{\text{horizontal}} + M(1 - p_{jkq}) \forall i \in I, \quad \forall k \in K, \forall q \in Q. \quad (16)$$

$$y_{ijq} + \text{length}_i \leq L_q - C_{\text{chamber}} + M(1 - p_{jkq}) \forall i \in I, \quad \forall k \in K, \forall q \in Q. \quad (17)$$

$$x_{ijq} + \text{width}_i \leq x_{ijq} + W_q(1 - s_{ijq}^{\text{left}}) \forall i, j \in I, \quad \forall q \in Q. \quad (18)$$

$$y_{ijq} + \text{length}_i \leq y_{ijq} + L_q(1 - s_{ijq}^{\text{behind}}) \forall i, \quad j \in I, \forall q \in Q. \quad (19)$$

$$x_{ijq} \geq x_{ijq} + \text{width}_i + D_{\text{horizontal}} - M_{s_{ijq}^{\text{behind}}} - M(1 - s_{ijq}^{\text{left}}) \forall i, \quad j \in I, \forall q \in Q. \quad (20)$$

$$y_{ijq} \geq y_{ijq} + \text{length}_i + D_{\text{vertical}} - M_{s_{ijq}^{\text{behind}}} - M(1 - s_{ijq}^{\text{left}}) \forall i, \quad j \in I, \forall q \in Q. \quad (21)$$

Constraints (9)–(21) ensure that the minimum safe distances are kept between the ships that are assigned to the same lockage operation and that all the ships keep enough distances between themselves and the lock chamber.

To ensure that the lock chamber is fully occupied in each lockage, there should be ships placed on the left and right sides, as well as on the front and back sides. Constraints (9) and (10) ensure that, for each ship $i \in I$, there should be ships on its left and back sides, respectively. Constraint (11) represents that if ship $i$ is on the left side of ship $j$, then ship $j$ cannot be placed on the left side of ship $i$, and vice versa. This also applies to the front and back situations of any two ships $i, j \in I$, represented via constraint (12). Constraint (13) represents that if ships $i$ and $j$ are placed in the same lockage $k$ of lock $q$, one of them will be on either the left or back side of the other ship.

Constraints (14) and (15) ensure that the minimum safety distance between each ship and the lock chamber is kept. Constraints (16) and (17) ensure that each ship will be placed within the chamber. Constraint (18) represents that if ship $i$ is on the left side of ship $j$, the $x$-coordinate of ship $j$ should be at least larger than the $x$-coordinate of ship $i$ plus its width $\text{width}_i$. Constraint (19) represents that if ship $i$ is on the back side of ship $j$, the $y$-coordinate of ship $j$ should be at least larger than the $y$-coordinate of ship $i$ plus its length $\text{length}_i$. Constraint (20) ensures that if ship $i$ is on the completely left side of ship $j$, the horizontal intership safety distances are kept. Constraint (21) ensures that if ship $i$ is
Figure 1: An illustrative example of ships passing a lock on the inland waterway.

Figure 2: The proposed coordination scheme.

Figure 3: Time horizon.

Table 1: List of parameters.

| Symbols | Definitions |
|---------|-------------|
| I       | Set of upcoming ships |
| I_s     | Set of prioritized ships |
| Q       | Set of locks |
| Z       | Set of segments of the waterway |
| K       | Set of lockages |
| N       | Set of ship speed |
| W_q     | Width of lock q |
| L_q     | Length of lock q |
| width_i | Width of ship i |
| length_i| Length of ship i |
| D_horizontal | Minimal safety distance between ships next to each other horizontally |
| D_vertical | Minimal safety distance between ships behind each other |
| C_chamber | Minimal safety distance between ships and the front and back chamber |
| Tsail_i (n) | Sailing time of ship i when it chooses speed n |
| T_start | Arrival time of ship i into the controlled area |
| T_process | The processing time of ship i entering and leaving the chamber |
| T_earliest | The earliest available time of the lock |
3.2.3. Constraints of Lockage Operations.

\[ t_{ij}^{\text{lockage}} \geq T_{\text{earliest}}, \forall q \in Q, \]  
\[ t_{kj}^{\text{lockage}} \geq p_{skq}^{\text{arrival}} + t_{kj}^{\text{processing}}, \forall i \in I, \forall k \geq 2, \forall q \in Q, \]  
\[ t_{kj}^{\text{processing}} \geq T_{\text{process}} \left( \sum_{i \in I} p_{skq} \right), \forall k \in K, \forall q \in Q, \]  
\[ t_{k+1,q}^{\text{lockage}} \geq t_{kj}^{\text{lockage}} + t_{k+1,q}^{\text{processing}}, \forall k \in K, \forall q \in Q. \]  

Constraint (22) represents that the first lockage operation should be carried out after the earliest available time of each lock. Constraint (23) ensures that the completion time of lockage \( k \) of lock \( q \) should be larger than the arrival times of the ships that are handled plus the processing time of them. Constraint (24) represents that if any ship will be handled in lockage \( k \), the processing time equals \( T_{\text{process}} \). Constraint (25) ensures that the completion time of a lockage \( k \) should be at least larger than the completion time of the previous lockage \( k+1 \) plus the processing time of ships.

3.2.4. Constraints of Ship Priorities. Considering the fact that certain types of ships have priorities in passing locks, these priorities should be considered in the model. As the ship priorities may differ in different countries, this paper mainly does not specify the ship types. For any ship \( i \in I_\ast \), constraints (26) and (27) ensure that it will be handled in the first lockage of each lock and that the deadline for leaving the lock is respected, respectively:

\[ p_{l,i,q} = 1 \forall i \in I_\ast, \forall q \in Q, \]  
\[ t_{iq}^{\text{departure}} \leq T_{\text{deadline}}, \forall i \in I_\ast, \forall q \in Q. \]

3.2.5. Constraints of Underoccupied Lockages. The proposed coordination scheme is carried out every \( N_{\text{time}} \), and it is possible that the last lockage may be underoccupied. This implies that the next set of upcoming ships could also be placed in the lockages that are not fully occupied in the previous coordination. After solving coordination problem \( p-1 \) for the previously entered ships, represented as set \( I_{p-1} \), the lockage operations for these ships are determined. For the next set of \( I_p \) upcoming ships, when solving problem \( p \), the ships in \( I_{p-1} \) that are handled in the last lockage will be considered as "virtual ships" in coordination problem \( p \). In other words, for these virtual ships \( v \in I_{p-1} \), it is assumed that \( p_{v,1,q} = 1 \) holds in coordination problem \( p \). The placements of these ships in the lock will be considered known and cannot be violated by the ships in \( I_p \). If ship \( m \in I_p \) will be placed in the last lockage of coordination problem \( p-1 \), then \( p_{m,1,q} = 1 \).

4. Solution Steps

We propose an LNS-based solution method to solve the formulated problem. An LNS heuristic consists of the following steps:

1. Initialization with the current global best solution
2. First application of the destroy method and then the repair method to obtain a new solution
3. Evaluation of the new solution and determination of whether this solution should become the new current solution or whether it should be rejected
4. Updating the new current global best solution if needed
5. Checking if the termination condition to end the search process is satisfied

More details of the LNS heuristics can be found in [9, 10]. It has been commonly used and is well known to be effective in solving vehicle routing problem and its variants [14, 15].
problems (2)–(27), in which the values of the found. At this point, a new solution are kept and solutions of the already removed characteristics are described as follows:

\[ \text{the removal heuristic to decide on which two removal heuristics to decide on which} \]

The removal heuristics are described as follows:

(i) Random removal: the random removal simply selects \( m \) ships at random and removes them from the solution.

(ii) Worst removal: this removal strategy removes the ships with the longest duration time from each lock \( q \), represented as \( t_{\text{departure}} - t_{\text{arrival}} \). It is reasonable to remove the ships that have to spend longer time in passing locks and reinsert them to obtain a better solution.

After the removal, we use a commercial solver to solve problems (2)–(27), in which the values of \( |I| - m \) ships are kept and solutions of the already removed \( m \) ships are found. At this point, a new solution \( s' \) is generated, which is compared with the current best solution \( s_{\text{best}} \). If the objective value of the new solution is smaller than the current best objective value, the new solution is better and is saved as the current best solution, and the new objective value is saved as the current best objective value. Then, the removal and reinsertion process is started over again. If the objective value of the new solution is larger than the current best objective value, the new solution will be left out and the removal and reinsertion iteration is carried out again by removing another \( m \) ships and reinserting them with the solver. The whole procedure stops after a predefined number of iterations.

### 5. Simulation Results

Simulations are carried out to assess and analyze the effectiveness of the proposed approach. We first describe the experimental setting and then present examples of lockage plans for each lock and the involved ships. Then, a comparison of solution quality in one of the cases is presented as an instance. The performance of the proposed LNS-based heuristics is also compared with respect to lock-relevant and ship-relevant key performance indicators.

#### 5.1. Experimental Settings

Our simulations are performed on an Intel Core i7-7500 CPU with 8 GB RAM running Windows 10 and are implemented in MATLAB, in which CPLEX 12.8.1 is used as the optimization solver. For the LNS heuristics, 20% ~ 30% of the ships are removed and inserted in each iteration. We set up 10 different cases in which 64 ships are entering three sequential locks in the next 2–8 hours. In other words, there are 8 ~ 32 ships entering the entrance of waterway every hour. In addition, it is assumed that the locks are empty at the beginning of optimization.

The lengths of ships range from 50 m to 80 m, and their widths range from 10 m to 20 m. For each lock, \( L_1 = 230 \text{ m}, W_1 = 30 \text{ m}; \quad L_2 = 150 \text{ m}, W_2 = 50 \text{ m}; \quad\) and \( L_3 = 200 \text{ m}, W_3 = 50 \text{ m} \). The processing time for each lockage operation \( T_{\text{process}} \) is equal to 30 minutes. There are 5 segments from the waterway entrance to Lock 1, from Lock 1 to Lock 2, and from Lock 2 to Lock 3; the length of each segment is 12 km. In each segment, four types of ship speeds are considered: 24km/h, 32km/h, 40km/h, and 48km/h. Subsequently, four types of sailing time are also considered (in minutes): 30 minutes, 22 minutes, 18 minutes, and 15 minutes. For each case, 30 iterations are carried out in the optimization process and we run 10 repetitions of simulations and use the average values. To generate the initial solution, it is assumed that all ships choose the fastest sailing speed and they will be served at each lock on a first-come-first-served basis.

#### 5.2. Example of Lockage Plans

Figures 5 and 6 give examples of ships’ placements in one of the locks according to the optimization results of LNS-random and LNS-worst heuristics in Case 1, in which L1–L20 represent the numbers of lockage operations and L1 represents the first lockage operation. As can be seen, all the ships keep enough distances between themselves and the lock chambers. Comparing with Figure 5, less lockage operations are required to handle the same set of ships in Figure 6.

Figures 7 and 8 give the lockage plans of different locks in Case 1. First, it could be seen that LNS-worst heuristics generate lock scheduling plans that require less number of lockage operations to handle the same set of ships, compared with LNS-random heuristics. This also implies that the locks

### Table 2: List of variables.

| Symbols | Definitions |
|---------|-------------|
| \( x_{iq} \) | Ship \( i \)'s position on \( x \)-axis in lock \( q \) |
| \( y_{iq} \) | Ship \( i \)'s position on \( y \)-axis in lock \( q \) |
| \( s_{left} \) | =1 if ship \( i \) is on the completely left side of ship \( j \) in lock \( q \) |
| \( s_{behind} \) | =1 if ship \( i \) is behind ship \( j \) in lock \( q \) |
| \( p_{iq} \) | =1 if ship \( i \) is processed at lockage \( k \) of lock \( q \) |
| \( t_{\text{lockage}} \) | Completion time of lockage \( k \) of lock \( q \) |
| \( t_{\text{processing}} \) | Processing time of lockage \( k \) of lock \( q \) |
| \( t_{\text{arrival}} \) | Arrival time of ship \( i \) at the lock \( q \) |
| \( t_{\text{departure}} \) | Departure time of ship \( i \) from lock \( q \) |
| \( t_{\text{setup}} \) | Preparation time of ship \( i \) before entering lock \( q \) |
| \( v_{\text{sailing}} \) | =1 if ship \( i \) chooses speed \( n \) at segment \( z \) |
| \( t_{\text{enter}} \) | Entering time of ship \( i \) to segment \( z \) |
are more utilized in Figure 8, as more ships are handled in each lockage operation. On the other hand, the idle times of the locks in Figure 8 are also longer than those in Figure 7. This is because the locks sometimes need to wait for more ships to enter in each lockage.

Figures 9–12 show the time that each ship spends from entering the controlled waterway area till leaving the last lock. It can be seen from Figure 10 that the ships could spend less time waiting at each lock with the optimized plans and passing all the locks with a shorter time, comparing with Figure 9.

5.3. Analysis of Solution Quality. Figure 13 compares the quality of solutions of LNS-random and LNS-worst heuristics with different imposed number of iterations. The values are calculated as the objective values of the LNS-based heuristics divided by the objective value of optimal solution generated with the CPLEX solver, in percentages. It is noted that these values are the average values of all cases. As can be seen, both heuristics could reach optimality within 40–50 iterations. LNS-worst heuristic performs better than LNS-random heuristic as it converges faster to optimal solutions.

5.4. Comparison of Key Performance Indicators. This paper considers two types of key performance indicators (KPIs) to evaluate the lock scheduling plans: lock-relevant KPIs that reflect the utilization of lock capacity for handling ships and ship-relevant KPIs that reflect if the ships have passed all the locks efficiently.

5.4.1. Lock-Relevant KPIs. The lock-relevant KPIs are given as follows:

(i) KPI 1: the number of lockages required to process the ships per lock
(ii) KPI 2: the time when the last lockage operation finishes per lock
(iii) KPI 3: the sum of idle times in which the lock does not process any ship
KPI 1 reflects whether each lockage operation handles as many ships as possible and that lower values imply higher utilization. KPI 2 reflects the time required for the ships to pass all locks; a shorter time represents higher efficiency. KPI 3 represents the waiting times of the locks for the upcoming ships to arrive. While lower value of KPIs 1 and 2 means higher lock utilization, this is not necessary in the case for KPI 3. This is because if more ships are handled in the same lockage, then the lock may need to wait for all ships to enter so that it could start the lockage operation.

Table 3 shows the maximum, minimum, and average ratio of the proposed heuristics to the benchmark scenario.
with respect to KPIs 1, 2, and 3 from the 10 cases. The values reported in the table are obtained from dividing the values of KPIs 1–3 generated with each heuristic by the values generated in the benchmark scenario. First, both heuristics could improve efficiency and utilization of the locks as they all lead to lower values of KPIs 1, 2, and 3. Second, LNS-worst performs better than LNS-random with respect to KPIs 1, 2, and 3. With LNS-worst heuristic, each lock is expected to spend 11% less time in lockage operations and 13% less time to handle ships and the idle time during which the locks are not in use can also be reduced. It is noted that the values are obtained from the 10 cases in which the frequency of ship arrivals differs and that this approach may lead to larger improvements in extreme situations with denser ship arrivals.

5.4.2. Ship-Relevant KPIs. The ship-relevant KPIs are given as follows:

(i) KPI 4: each ship’s departure time from the last lock on the waterway
(ii) KPI 5: the sum of waiting times of each ship at the locks

KPIs 4 and 5 reflect the efficiency of ships in passing the locks on the waterway; a shorter time represents higher efficiency. Table 4 shows the maximum, minimum, and average ratio of the proposed heuristics to the benchmark scenario from the 10 cases. The values reported in the table are obtained from dividing the values of KPIs 4 and 5 generated with each heuristic by the values generated in the benchmark scenario. As can be seen, with
both heuristics, the ships could leave the lock with a shorter time and do not need to wait for long time at the lock. Meanwhile, LNS-worst performs better than LNS-random with respect to both KPIs 4 and 5, and with the suggested ship arrivals by LNS-worst, the ships could leave the last lock around 9% earlier, as well as spend 28% less waiting time at the locks according to the simulation settings of this paper.
6. Conclusions and Future Work

This paper addresses an optimization approach for scheduling lock operations and ship arrivals at locks on inland waterways. Two large neighborhood search based heuristics are proposed to coordinate the arrival and departure of ships and plan lockage operations for each lock. Unnecessary ship waiting times at lock entrances could be reduced by adjusting their sailing times and speeds on the waterway, which when implemented leads to shorter passing times through all locks.

The simulation results show that the proposed approach could be beneficial for both ships and locks. For the ships, they can spend 28% less time waiting at the locks and leave the last lock 9% earlier with the optimized ship arrivals. For the locks, 10% less lockage operations and 13% less time are required to handle the same number of ships with the optimized lockage plans. This means that the proposed approach could reduce the congestion at locks and thereby improve the utilization of inland waterway transport. It is noted that, in practice, the exact reduce waiting times can be different as taller and wider inland ships exist and the machinery failure may affect the actual schedules. With more real data of locks and the arrival ships, the proposed method can be further validated.

Further research can be extended in several directions. First, for practical implementations, an information platform for exchanging information between ships and locks is required, and decision support software also needs to be developed and installed to send and receive messages to/from the platform. Second, the model can be extended to inland waterway network which involves more locks, bridges, and other waterway elements.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was funded by Hubei Key Laboratory of Inland Shipping Technology.

References

[1] Y. Yuan, B. Ji, X. Yuan, and Y. Huang, “Lockage scheduling of three gorges-gezhouba dams by hybrid of chaotic particle swarm optimization and heuristic-adjusted strategies,” *Applied Mathematics and Computation*, vol. 270, pp. 74–89, 2015.
[2] J. Verstichel, P. De Causmaecker, F. Spieksma, and G. Vanden Berghe, "The generalized lock scheduling problem: an exact approach," *Transportation Research Part E: Logistics and Transportation Review*, vol. 65, no. 1, pp. 16–34, 2014.

[3] J. Verstichel, P. De Causmaecker, F. C. R. Spieksma, and G. Vanden Berghe, "Exact and heuristic methods for placing ships in locks," *European Journal of Operational Research*, vol. 235, no. 2, pp. 387–398, 2014.

[4] J. Verstichel, J. Kinable, P. De Causmaecker, and G. Vanden Berghe, "A combinatorial benders’ decomposition for the lock scheduling problem," *Computers & Operations Research*, vol. 54, pp. 117–128, 2015.

[5] X. Yuan, B. Ji, Y. Yuan, X. Wu, and X. Zhang, "Co-scheduling of lock and water-land transshipment for ships passing the dam," *Applied Soft Computing*, vol. 45, pp. 150–162, 2016.

[6] W. Passchyn, D. Briskorn, and F. C. R. Spieksma, "Mathematical programming models for lock scheduling with an emission objective," *European Journal of Operational Research*, vol. 248, no. 3, pp. 802–814, 2016.

[7] W. Passchyn, S. Coene, D. Briskorn, J. L. Hurink, F. C. R. Spieksma, and G. Vanden Berghe, "The lockmaster’s problem," *European Journal of Operational Research*, vol. 251, no. 2, pp. 432–441, 2016.

[8] B. Ji, X. Yuan, and Y. Yuan, "A hybrid intelligent approach for co-scheduling of cascaded locks with multiple chambers," *IEEE Transactions on Cybernetics*, vol. 49, no. 4, pp. 1236–1248, 2018.

[9] K. F. Boersma, G. C. M. Vinken, and J. Tournadre, "Ships going slow in reducing their NOx emissions: changes in 2005–2012 ship exhaust inferred from satellite measurements over Europe," *Environmental Research Letters*, vol. 10, no. 7, Article ID 074007, 2015.

[10] J. J. Corbett, H. Wang, and J. J. Winebrake, "The effectiveness and costs of speed reductions on emissions from international shipping," *Transportation Research Part D: Transport and Environment*, vol. 14, no. 8, pp. 593–598, 2009.

[11] P. Shaw, "Using constraint programming and local search methods to solve vehicle routing problems," in *Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming*, pp. 417–431, Springer, Pisa, Italy, October 1998.

[12] D. Pisinger and S. Ropke, "Large neighborhood search," in *Handbook of Metaheuristics*, pp. 99–127, Springer, Berlin, Germany, 2019.

[13] M. A. F. Belo-Filho, P. Amorim, and B. Almada-Lobo, "An adaptive large neighbourhood search for the operational integrated production and distribution problem of perishable products," *International Journal of Production Research*, vol. 53, no. 20, pp. 6040–6058, 2015.

[14] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transportation Science*, vol. 40, no. 4, pp. 455–472, 2006.

[15] D. Briskorn and E. Pesch, "Variable very large neighbourhood algorithms for truck sequencing at transshipment terminals," *International Journal of Production Research*, vol. 51, no. 23-24, pp. 7140–7155, 2013.