applied sciences

MDPI

Article

An Improved Shuffled Frog-Leaping Algorithm for Solving the Dynamic and Continuous Berth Allocation Problem (DCBAP)

Hsien-Pin Hsu [1,*] and Tai-Lin Chiang [2,*]

[1] Department of Supply Chain Management, National Kaohsiung University of Science and Technology, No.142, Haijhuan Rd., Nanzih Dist., Kaohsiung City 81157, Taiwan

[2] Department of Business Administration, Minghsin University of Science and Technology, No.1, Xinxing Rd., Xinfeng Hsinchu 30401, Taiwan

* Correspondence: hphsu@nkust.edu.tw (H.-P.H.); tlchiang@must.edu.tw (T.-L.C.)

Received: 27 August 2019; Accepted: 31 October 2019; Published: 3 November 2019

Featured Application: The proposed Improved Shuffled Frog-Leaping Algorithm (ISFLA) can be applied to allocate berthing positions to ships calling a container terminal with the quay being used a continuous line.

Abstract: This research deals with the *dynamic* and *continuous* berth allocation problem (DCBAP) in which both arrived and incoming ships are considered and a quay is used as a continuous line to accommodate as many ships as possible at one time. The DCBAP is solved by a two-stage procedure. In the first stage a heuristic/metaheuristic is used to generate alternative ship placement sequences while in the second stage a specific heuristic is employed to place ships and resolve overlaps of ships for the development of a feasible solution. Different methods, including FCFS (First Come First Served), SFLA (Shuffled Frog-Leaping Algorithm), and ISFLA (Improved Shuffled Frog-Leaping Algorithm), were employed in the first stage for comparison. The experimental results show that the ISFLA outperforms the others in terms of solution quality, implying that the ISFLA has the potential to deal with the DCBAP in a container terminal.

Keywords: berth allocation problem (BAP); heuristic; shuffled frog-leaping algorithm (SFLA)

1. Introduction

International trade is the exchange of goods or services across nations. There are different kinds of transportations for exchanging goods, including air, land, and maritime. Among these transportations, maritime transport is essential due to its having a relatively lower transport cost as a result of mass transport. Among various kinds of maritime transport, container transport is especially important, as the number of global container shipments is increasing continuously. To serve calling ships, many of the busy global ports, such as Hamburg, Rotterdam, and Antwerp in Europe, and Busan, Shanghai, and Hong Kong, employ multi-user terminals [1]. To improve productivity at this kind of terminal, a better solution for the berth allocation problem (BAP) is essential.

There are many approaches available for improving the productivity of a container terminal. Basically, these approaches can be classified into the two categories: hardware and software approaches. For example, increasing the number of quays or quay cranes (QCs) is an example of a hardware approach, whereas improving the utilization of quays or QCs is an example of a software approach. In this research, the software approach is used, as it is less expensive and time-consuming.

In a container terminal, the operations can mainly be classified into three areas: seaside, yard, and landside. In the seaside area, there are three famous seaside operational problems: the berth allocation

problem (BAP), the quay crane assignment problem (QCAP), and the quay crane scheduling problem (QCSP) [2,3]. These seaside operational problems are more critical than others in the yard area and landside due to the use of berths and QCs. In this research, we therefore focus on one of the three well-known seaside operational problems, i.e., the BAP. However, our literature review shows that the BAP can be further characterized with respect to certain factors. To deal with the BAP, it is necessary to understand the factors and variables affecting it, as further detailed in the next paragraph.

The BAP is a problem of allocating berths to ships. This problem can be further characterized by two main factors: (1) the arrival times of ships; and (2) the configuration of quays. The first factor characterizes the BAP as being a *static* or *dynamic* version of the problem. In the *static* BAP (SBAP), only arrived ships are considered, whereas in the *dynamic* BAP (DBAP), incoming ships are also taken into consideration. In this research, we focus on the dynamic version (DBAP) due to the fact that ships continue to come during BAP planning. The second factor characterizes the BAP as either a *discrete* or *continuous* version of the problem. In the *discrete* BAP, a quay is separated into several fixed sections, and each section can only accommodate one ship at a time, whereas in the *continuous* BAP, a quay in its entirety is regarded as a continuous line that is able to accommodate as many ships as possible at any given time. In this research, we focus on dealing with the *dynamic* and *continuous* BAP (DCBAP), whereby a quay is used as a continuous line, and both arrived ships and incoming ships are considered.

Different approaches are available for dealing with the BAP, including mathematical models, heuristics and metaheuristics. Mathematical models, such as the Integer Programming (IP) and Mixed Integer Programing (MIP) models, are exact approaches commonly used to find the optimal solution. However, due to NP-complete [4–6], the exponentially increasing computational times required for exact approaches prevent them from being employed in practice. As a result, heuristics and metaheuristics have become popular. Recently, evolutionary and population-based metaheuristics have increasingly been being used to deal with various container terminal operational problems. One advantage of this kind of approach is that they can iteratively drive solution(s) towards optimality. This improves the simplicity of simple heuristics. Another advantage is that through control of parameters such as the total number of iterative runs, these kinds of metaheuristics can avoid the computational intractability faced by exact approaches [7]. Examples of these kinds of metaheuristics include Ant Colony Optimization [8–15], Particle Swarm Optimization (PSO) [7,16,17], and genetic algorithm (GA). In [18], it was shown that GAs have been widely used to deal with seaside container terminal operational problems.

Proposed by Eusuff and Lansey [19], the Shuffled Frog-Leaping Algorithm (SFLA) is an evolutionary population-based metaheuristic. It employs a swarm of frogs to mimic memetic evolution with the aim of finding the position (solution) with the greatest amount of food. During evolution, these frogs exchange information that can affect the behavior of other frogs. As genes can only be passed from parents to offspring, memetic spread is considered to be faster, as memes can propagate between any individuals [20]. In fact, the SFLA includes the advantages possessed by evolution-based memetic algorithms (MA) and social behavior-based PSO algorithms [20]. The SFLA and its variants are currently attracting increasing attention and are increasingly being used to deal with different problems. However, to our best knowledge, they have never been used to deal with the BAP.

The main purpose of this research is to develop an improved SFLA (ISFLA) to deal with the DCBAP, as this new kind of metaheuristic has never been used for this purpose. The ISFLA is employed in a two-stage procedure. In the first stage, the ISFLA is used to generate alternative ship placement sequences as inputs to the second stage. In the second stage, a specific heuristic is used to place ships one by one into a berth plan and resolve overlaps of ships in order to generate a feasible solution. The ISFLA includes some novel features, such as self-adaptive jump, direct-jump prevention, push jump, and neighborhood jump, enabling smart leaps for frogs in order to better search the solution space for a possible solution. We conducted experiments investigating the effectiveness of the ISFLA

in comparison to FCFS and SFLA, which were employed in the first stage. The results showed that the ISFLA outperformed the other two in terms of solution quality.

The rest of this research is organized as follows: Section 2 presents a literature review on the CBAP and basic SFLA; Section 3 presents a formulation of the Mixed Integer Linear Programming (MILP) model for the DCBAP; Section 4 develops the ISFLA for solving the DCBAP; Section 5 includes numerical examples; Finally, Section 6 presents the conclusion and future research directions.

2. Literature Review

2.1. The Operations in a Seaport Container Terminal

Figure 1 illustrates two ships berthing alongside the quay of a container terminal. The container terminal is separated into the following three areas: seaside, yard and landside [3].

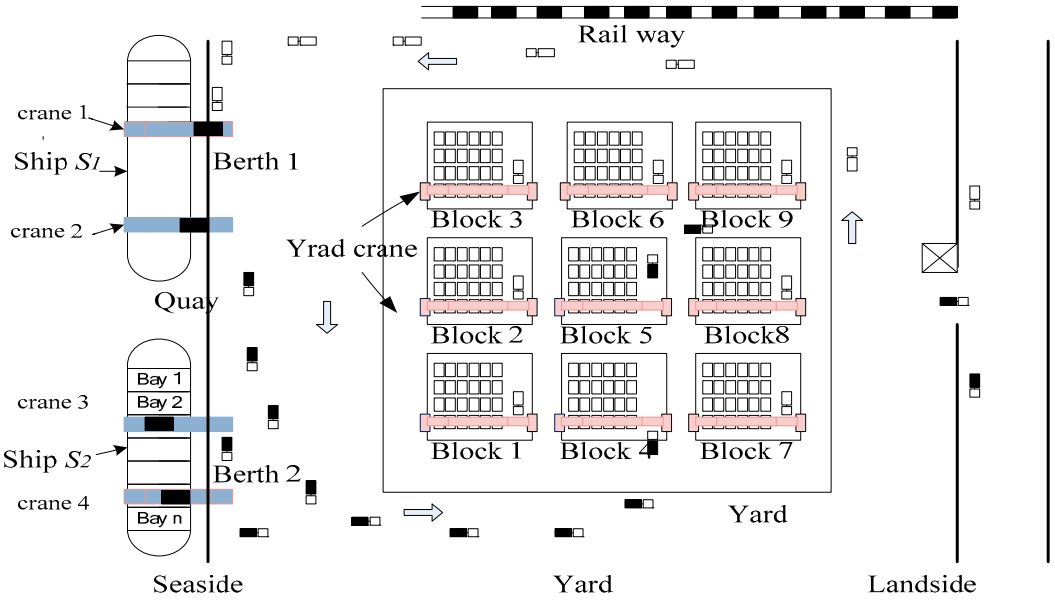

Figure 1. An example of a seaport container terminal.

The quay in the container terminal can be configured as either a discrete or continuous quay. With a discrete configuration, the quay is separated into several fixed sections, and each section is allowed to accommodate one ship at a time. With a continuous configuration, the entire quay is used as a continuous line that can accommodate as many ships as possible at one time. Along the quayside, rail-mounted quay cranes are used to load/unload containers to/from trucks. The trucks will transport containers between the berthed ships and the storage areas in the yard area.

The yard area serves as a buffer between the sea and land sides, which is an area of temporary storage for containers. The yard area is divided into many blocks, and each block is composed of several rows for stacking containers. For each block, there is a yard crane used to load and unload containers on or off the trucks. The containers in the yard area will be transshipped to other ships or transported to customers via road or rail. Figure 2a shows the flows of intra-terminal transshipment between ship 1 and ship 2.

The land side contains operations consisting of transporting containers to and from the container terminal, by trucks, road or railway. Figure 2b shows two types of intermodal transportation: ship-to-road and ship-to-rail. In this yard area, both intra-terminal transshipment and intermodal transshipments operate simultaneously.

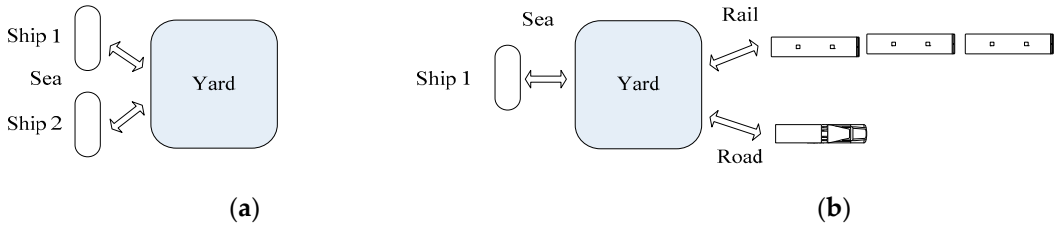

Figure 2. (**a**) Intra-terminal transshipment; (**b**) Inter-modal transportation.

In a container terminal, there are different operational planning problems. These can be listed as follows:

- Allocation of berths for calling ships (BAP).
- Assigning of QCs for berthed ships for loading and unloading containers (QCAP).
- Scheduling of QCs (QCSP).
- Assigning of yard trucks.
- Scheduling of yard trucks.

The BAP, QCAP and QCSP are widely recognized seaside operational problems. In this research, we will focus on the BAP of the continuous quay configuration. Relevant CBAP studies are reviewed in the next section.

2.2. Studies on CBAP

Some studies, such as those by Lai and Shih [21], Brown et al. [22,23], Imai et al. [24–26], and Nishimura et al. [27], have focused on the DBAP, while Lim [4], Li et al. [28], Guan et al. [29], Park and Kim [30,31], and Kim and Moon [5], Imai et al. [1], Wang and Lim [32], Lee and Chen [33], Zhen et al. [34] carried out studies focused on the CBAP. Due to the focus on quays with continuous configuration, we detail these CBAP studies below.

The study by Lim [4] is an early study focusing on the CBAP. In that study, the CBAP was considered as a two-dimensional packing problem, but transformed into a graph theoretical representation once the authors had captured the CBAP characteristics. Then, the author proposed a heuristic for solving this problem with the objective of minimizing the quay length required to accommodate all incoming ships under the assumption that these ships would not further change their berthing positions. On the other hand, Li et al. [28] solved the CBAP both with and without this restriction in ships' movements. Their objective was to minimize the makespan of the schedule. Guan et al. [29] developed a heuristic for solving the CBAP. Their objective was to find the solution with the minimum total weighted completion time of ships. Park and Kim [30] proposed a subgradient optimization method for dealing with the CBAP with the objective of minimizing costs arising from the delayed departures of ships due to undesirable service order and the additional complexity of handling containers when ships are served at non-optimal mooring locations in the port. This study appears to be more practical than the above-mentioned studies, in that it took the actual berthing positions of ships into consideration. Kim and Moon [5] addressed the same CBAP discussed in Park and Kim [30]. The authors proposed a simulated annealing method to deal with this problem, finally concluding that this approach had the capacity to find an optimal solution. However, the CBAP studies mentioned above neglected the impact of berthing position on the handling time of a ship. In a later study, Park and Kim [31] studied a CBAP with a similar objective function to those used in Park and Kim [30] and Kim and Moon [5]. A major difference of the objective function used in [31] was that it accounted for additional costs resulting from the early or late start of ship handling with respect to the ETA of a ship. Their approach generated the optimal starting times of ship services and berthing positions, as well as QC assignments to ships. However, in that study, the handling times of ships were still independent from the ships' mooring positions. Having completed the DBAP [24–26], Imai et al. [1] moved on to deal with the

CBAP, which was treated as a two-dimensional cutting stock problem. The CBAP was solved by using a two-stage heuristic. The first stage generates an initial solution to the DBAP, and then this solution is further transformed into a feasible one by repositioning overlapped ships as well as sparsely located ships at the second stage. In that study, the impacts of the mooring positioning of a ship on its handling time were taken into consideration. Wang and Lim [32] solved the CBAP by proposing a stochastic beam search. Their experiments showed that the proposed approach was able to find a near-optimal solution, with the solution being better than those found by state-of-art metaheuristics and traditional deterministic beam search in terms of accuracy and efficiency. Lee and Chen [33] proposed a neighborhood-search optimization heuristic to solve the CBAP. In that study, factors including the First Come First Served (FCFS) rule, clearance distance between ships, and possible ship shifting were taken into consideration. The experimental results showed that a better decision could be achieved when taking these factors into consideration. In Zhen et al. [34], the authors proposed a two-stage decision model for dealing with the CBAP. The arrival times and handling times of ships in the model are treated as uncertainties. In the first stage, this model first initiates a baseline schedule that is further adjusted by using a reactive recovery strategy in the second stage. The objective of that approach was to minimize penalty costs incurred by deviating from the initial schedule. For the CBAP in a large-scale environment, the authors proposed a metaheuristic approach based on the simulated annealing approach. Our literature review shows there is still a lack of implementation of SFLA-based approaches for dealing with the BAP, especially the DCBAP.

The CBAP is not completely equivalent to the cutting-stock problem (CSP), because not all rectangles (placing items)—which correspond to calling ships in the CBAP—are already available in the CSP, as a result of the incoming ships in the CBAP. In addition, the constraints in the CBAP can be graphically portrayed. For example, some rectangles in a berth plan cannot be placed before corresponding predetermined vertical lines that represent the estimated arrival times (ETAs) of calling ships. This restriction is simply based on the fact that ships cannot be berthed before they have arrived at the port. Thus, the static version of the CBAP is identical to the fixed-orientation CSP but the dynamic BAP is not [1].

In many CBAP studies, such as those by Lim [4], Li et al. [28], and Guan et al. [29], the authors only considered arrived ships and assumed that a ship's handling time was independent of its berthing location. Although incoming ships were taken into consideration in some studies, such as those by Park and Kim [30,31] and Kim and Moon [5], and fixed handling times had been assumed for those ships, Ref. [31] can only be regarded as a static version of CBAP, as ships were able to be served before their ETAs, while penalty costs were then imposed in the objective function for the use of such services. This relaxation downgrades the dynamic version of BAP to a static one [5]. Consequently, its constraint is also equivalent to that of the CSP with fixed orientation. In this present research, the CBAP to be solved is quite different from the conventional CSP, as the handling time of a ship is dependent on the ship's berthing location, and incoming ships are to be taken into consideration.

In addition, heuristics have been widely used to deal with the CBAP. However, high-level heuristics, i.e., metaheuristics, have seldom been used to deal with the CBAP. In particular, there is still a lack of using SFLA to solve the BAP. Such applications are still at the emerging stage.

2.3. The Basic SFLA

As a kind of population and evolutionary-based metaheuristic, the basic SFLA (B-SFLA) employs a swarm of frogs to find the position (solution) with the greatest amount of food. The goodness of a position can be indicated by an objective function value. In a D-dimensional space, the current position of a frog f is denoted as $X_f(t) = (X_{f1}, \ldots, X_{fD})$. During memetic evolution, frogs are separated into m groups, with the best frog being assigned to the memeplex 1; the 2nd best to the memeplex 2; the mth best to the memeplex m; the $m + 1$th best then back to the memeplex 1 again, and so on. As a result, each group has $n = F/m$ frogs, and these frogs are ordered decreasingly according to their objective

function values [35]. Following this, triangular distribution sampling, as in Equation (1), is used to form sub-memeplexes.

$$P_f = 2(n+1-f)/n(n+1)\ f = 1, \ldots,\ n \tag{1}$$

With Equation (1), the best frog in each memeplex has the highest probability $2/(n+1)$, while the worst frog has the lowest probability $2/n(n+1)$, of being selected into a corresponding sub-memeplex. Then, Roulette Wheel Selection (RWS) can be used to select q frogs from the original memeplex into the sub-memeplex for evolution and local searching. At the current time t, the leaping distance of the frog f is denoted as $D_f(t)$ and determined by Equation (2).

$$D_f(t) = \begin{cases} \min\left\{\text{int}\left\{R\left(X_b(t) - X_f(t)\right), S_m\right\}\right\} & \text{for positive step} \\ \max\left\{\text{int}\left\{R\left(X_b(t) - X_f(t)\right), -S_m\right\}\right\} & \text{for negative step} \end{cases} \tag{2}$$

The R is a random number in [0, 1]; $X_b(t)$ is the position of the best frog in the sub-memeplex; S_m is the maximum step allowed. The next position of the frog f is determined by Equation (3).

$$X_f(t+1) = X_f(t) + D_f(t) \tag{3}$$

If feasible and better, this next position will be accepted; otherwise, the $X_b(t)$ in Equation (2) will be replaced with $X_g(t)$ and the next position will be found again, where $X_g(t)$ is the position of the global best frog. If this next position is once again not better, a random jump will be made for the frog. Once all frogs in a sub-memeplex have completed several local searches, they then move on to process the next sub-memeplex until all sub-memeplexes have been completed. After completing one iterative run, the frogs are reshuffled and the next iterative run is started. This is carried out until the termination criteria are reached. For more details about the B-SFLA, one can refer to Eusuff et al. [35].

Our literature review shows that SFLA and its variants have been used in various areas, including water distribution network design [19], project management [36], knapsack problem [37], component pick-and-place problem [38], routing problem [39], gateway loading balance [40], local dimming optimization [41], underwater sonar image detention [42], job scheduling [43], big data [44], feature selection [45], etc. However, to our best knowledge, SFLA has never been used to deal with the BAP.

3. Formulation for a Mathematical Model of the DCBAP

3.1. Definition of the DCBAP

Definition 1. *The Dynamic and Continuous Berth Assignment Planning Problem (DCBAP) is a problem with the aim of finding a feasible berth plan that includes six-tuples:*

$$\text{DCBAP} = (L,\ H,\ V,\ C,\ P,\ Z)$$

where

L: *the length of a quay;*
H: *the planning horizon;*
V: *a finite set of calling vessels; $V = \{1, 2, \ldots, N\}$, where N is the total number of calling ships;*
Ç: *a finite set of berthing constraints;*
P: *a set of plans with assignments of berthing positions and start berthing times for ships; $\forall\ p_i \in P,\ \exists\ p_i = \left\{\cup_{j=1}^{j=N}\left(B_j, S_j\right)\middle| 0 \le B_j \le L - l_j,\ 0 \le S_j\right\}$, where $\left(B_j, S_j\right)$ is a pair of assignments, where B_j is the berthing position, S_j is the start berthing time and l_j is length of the calling ship j.*
Z: *an objective function mapping p_i to a time/cot value Z.*

The objective of the DCBAP is to find a p_i or p^* ($p_i,\ p^* \in P$) that minimizes the objective function value Z, in which p_i is a feasible solution, while p^* is the optimal solution subjecting to Ç. For each ship j a berthing position $\left(B_j, S_j\right)$ on P is assigned. Finding the $\left(B_j, S_j\right)$ for each ship j ($j = 1, \ldots, n$, where n is total number of calling ships) is an NP-hard problem.

3.2. Berthing Plan Representation for the DCBAP

A berth plan can represent a solution to the CBAP. Figure 3 illustrates a berth plan in which the X-axis represents the time dimension (planning horizon), while the Y-axis represents the space dimension (quay length). This plan includes one overlap of two ships j and k that are respectively located at positions at a (x_0^j, y_0^j) and a' $\left(x_0^k, y_0^k\right)$, where x_0^j and x_0^k are berthing times and y_0^j and y_0^k are berthing positions. However, this overlap makes this plan unfeasible. To be feasible, it needs to resolve this overlap. A ship being selected and moved to resolve an overlap is termed "target ship". It is noted that the arrival time (ETA_j) and the desired berthing position (d_j) are the best coordinates in a berthing plan for a ship j, as there are no increased waiting or handling times. Any deviation from these coordinates will increase the cost for the ship.

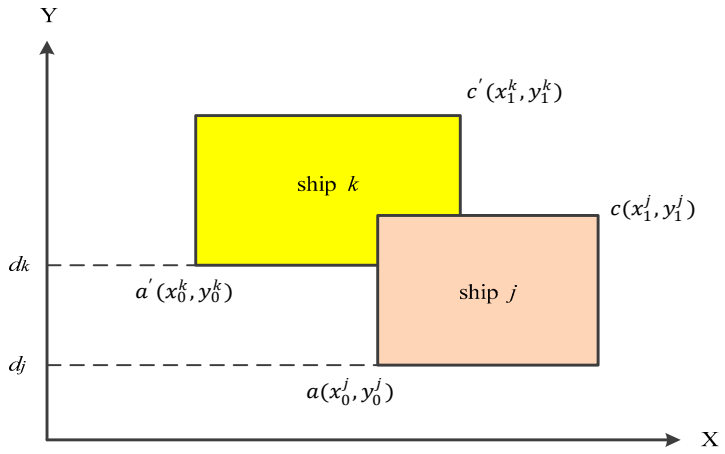

Figure 3. The coordinates of two corners of the ships j and k.

3.3. Estimation of Increased Handling Times for a Ship

In this research, it is assumed that the berthing position of a ship will affect the ship's handling time. In Figure 3, we give each ship j or k an initial handling time based on their respective desired berthing positions. Thus, if any of the ships deviates from its desired berthing position, then the ship's handling time will increase due to the greater distance that a truck will need to move a container.

Equation (4) is the formula for calculating the distance (Δb_j) for a ship j deviating from its desired berthing position.

$$\Delta b_j = \left|B_j - d_j\right| \tag{4}$$

where

B_j is the actual berthing position of ship j alongside the quay;
d_j is the desired berthing position of ship j alongside the quay.

Then, the increased handling times (ΔH_j) for the ship j is determined by Equation (5)

$$\Delta H_j = \beta \cdot \Delta b_j \tag{5}$$

where
β is the berth deviation factor; a rate estimating the increase of handling time due to the deviation from the desired berthing position (d_j).

Equation (6) is used to estimate T_j, the actual handling times of a ship j.

$$T_j = h_j + \Delta H \tag{6}$$

where

h_j is the original handling time for ship j (this is not affected by berthing position).

Finally, given the actual start berthing time of ship j (S_j), the completion time of a ship j can be estimated by Equation (7).

$$C_j = S_j + T_j, \ \ \forall j \in J \tag{7}$$

3.4. Estimation of Increased Waiting for a Ship

In Figure 1, one way of resolving the overlap is to move ship j in the +X direction. This will not increase the handling times, but rather the waiting times of the ship. The increased waiting times (ΔW_j) for a ship j can be estimated by Equation (8), where S_j is the start berthing time of the ship j.

$$\Delta W_j = S_j - ETA_j \tag{8}$$

3.5. Mathematical Model

This section formulates a mathematical model for the DCBAP. Firstly, the assumptions, parameters, indices, and decision variables for this model are introduced.

Assumptions

(1) Each ship is handled continuously until it is completed.
(2) Each ship has an estimated processing time.
(3) Each ship has a desired berthing position.
(4) Inter-ship clearance distance is included in ship length.
(5) A ship departs immediately if completed.

Indices

j a ship number; $j \in J = \{1, \ldots, N\}$
x a start berthing time; $x \in X = \{1, \ldots, H\}$
y a berthing position; $y \in Y = \{1, \ldots, L\}$

Parameters

L the quay length (meters)
H the planning horizon (hours)
N the total number of ships to call within the planning horizon H
l_j the length of ship j
ETA_j the expected time of arrival of ship j
d_j the desired berthing position of ship j
β the berth deviation factor ($\beta \geq 0$); increased handling times in hours per 100 m deviating from the desired berthing position of a ship
$c1$ the cost rate of waiting time (USD/hour)
$c2$ the cost rate of handling time (USD/hour)

Decision Variables

$$X_{jxy} = \begin{cases} 1, \text{ if the ship } j \text{ is assigned with the berthing postion } (x, y) \\ 0, \text{ otherwise} \end{cases}$$

B_j the berthing position of ship j ($j \in J$)
S_j the actual start berthing time of ship j ($j \in J$)
C_j the completion time of ship j ($j \in J$)
ΔW_j the increased waiting time of ship j ($j \in J$)
ΔH_j the increased handling time of ship j ($j \in J$).

The Mixed Integer Programing (MIP) model of the DCBAP can be formulated as follows.

$$Min\ Z = \sum_{j=1}^{N} \left(c_1 \cdot \Delta W_j + c_2 \cdot \Delta H_j \right) \tag{9}$$

$$\text{s.t.} \sum_{x=1}^{H} \sum_{y=1}^{L} X_{jxy} = 1 \quad \forall j \in J \tag{10}$$

$$\Delta W_j \geq 0 \quad \forall j \in J \tag{11}$$

$$\Delta H_j \geq 0 \quad \forall j \in J \tag{12}$$

$$B_j \leq L - l_j \quad \forall j \in J \tag{13}$$

$$B_j \geq 0 \quad \forall j \in J \tag{14}$$

$$l_j > 0 \quad \forall j \in J \tag{15}$$

$$l_j \leq L \quad \forall j \in J \tag{16}$$

$$X_{jxy} \in \{0, 1\} \quad \forall j \in J; \quad \forall x \in X; \quad \forall y \in Y \tag{17}$$

Equation (9) is the objective function for minimizing the total cost (Z), including the increased waiting cost, as well as the increased handling cost. Equation (10) stipulates that one berthing position can be assigned to only one ship at a time. Equation (11) is a requirement for ΔW_j stipulating that S_j should not start before ETA_j. Equation (12) defines a requirement for ΔH_j. Equations (13) and (14) are conditions for the assigned berthing position for a ship. Equations (15) and (16) are physical conditions for a ship. Equations (17) comprise a binary constraint for decision variables X_{jxy}.

4. The ISFLA for Solving the Simultaneous DCBAP

Due to the NP-hard problem, the MILP formulated in Section 3.3 will become computationally intractable when used to deal with problems of practical size. This section details the development of an ISFLA for dealing with the DCBAP.

4.1. Position Representation

Figure 4 shows a position representation for a frog searching a D-dimensional solution space (where $D = n$, n is the total number of calling ships) of the DCBAP. In this scheme, each u_j ($j = 1, \ldots, n$) indicates the placement order of ship j.

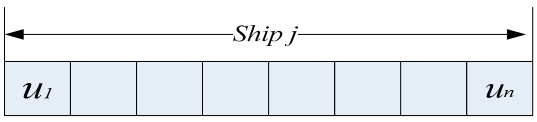

Figure 4. The encoding scheme of frog position.

The ROV technique used in Hsu (2016) [7] is used to transform real values into ranking numbers that represent a placement sequence of ships. For instance, the sequence [0.5,0.6,0.7,0.4] will be transformed into the ranking set [2,3,4,1], corresponding to the placement sequence for ships 1 to 4.

4.2. The ISFLA

The ISFLA possesses the following features: (1) multiple groups of frogs, (2) shuffled mechanism, (3) discrete operators with self-adaptive jump, and (4) self-adaptive mutation mechanism. These features are detailed below.

4.2.1. Multiple Groups of Frogs

For a given solution space, the ISFLA uses multiple groups of frogs and forms multiple search areas (focuses) that can be concurrently searched by the swarm of frogs. Frogs in the same group will search among them for the best frog in that group. Figure 5 shows the evolution of three groups of frogs from the iteration i to the iteration $i + 1$. The ISFLA does not use sub-memeplex or triangular distribution sampling. This allows all frogs in a memeplex to attend to evolution directly, which results in population advantage.

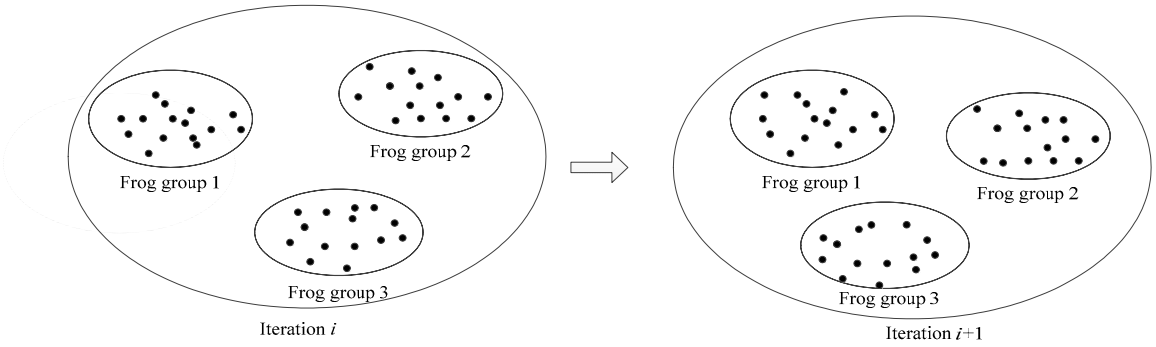

Figure 5. Multiple groups of frogs evolving in a solution space.

4.3. Shuffled Mechanism

The shuffled mechanism is used to regroup frogs belonging to the same swarm, which can initiate other opportunities for these frogs to be affected by other elites (i.e., the best frog in each group). This helps diversify frogs for the exploration of the solution space. In a D-dimensional solution space, the position of a frog i is denoted as $X_i = (X_i1, \ldots, X_{iD})$, and the frogs of this swarm are separated into m groups according to the following rules:

- the best is assigned to group 1;
- the 2nd best is assigned to group 2;
- the mth best is assigned to group m;
- the m + 1th best is assigned to group 1 again, and so on.

As a result, each group then contains F/m frogs (assuming that there are a total number of F frogs), and the frogs in each group are ordered in decreasing order based on their objective function values.

4.4. Discrete Operators with Self-Adaptive Jump

In this research, the ISFLA uses discrete operators to determine the next position for a frog based on the position of the frog relative to that of the target frog. An example is given in order to specify the self-adaptive jump of frogs.

Figure 6 illustrates three frogs, o, i, and j, with positions of $X_o(t)$ = [4,3,2,1,5,6], $X_i(t)$ = [1,2,3,4,5,6], and $X_j(t)$ = [4,1,3,2,5,6], respectively. Among the three frogs, the frog o, termed the "target frog", is in the best position, attracting frogs i and j to search around it. The frogs i and o are considered to be closer, due to sharing four of the same position elements (while frogs j and o only shar two of the same elements). A self-adaptive mechanism is required to cause the non-target frogs to adaptively approach the target frog, i.e., to enable a bigger jump for more distant frogs to be able to approach the target frog

quickly, while enabling smaller jumps for closer frogs to prevent them from jumping over the optima. However, non-target frogs should not jump directly to the target frog, as this will waste a local search.

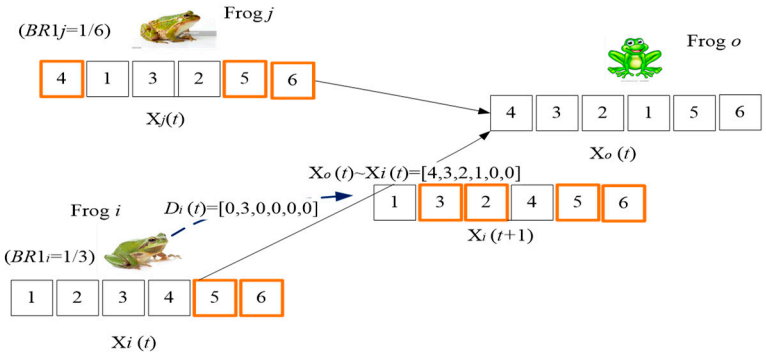

Figure 6. Frogs *i* and *j* jump towards the target frog *o*.

To enable self-adaptive jumps, we change Equations (2)–(18).

$$D_i(t) = (X_o(t) \sim X_i(t)) \odot RV_i(t) \tag{18}$$

$RV_i(t)$ refers to the adaptive binary vector, and is used to adjust the moving distance $D_i(t)$ for a frog i at the time t.

The operator "~" refers to the *total distance measuring operator*, which measures the distance between frogs *o* and *i* in cases where frog *i* jumps towards the target frog *o*. The operator works as follows.

$$X_{o,k}(t) \sim X_{i,k}(t) = \begin{cases} 0, & \text{if } X_{o,k}(t) = X_{i,k}(t) \\ X_{o,k}(t), & \text{if } X_{o,k}(t) \neq X_{i,k}(t) \end{cases} \tag{19}$$

In Equation (19), *k* indicates the *k*th elements in the position vectors of frog *o* and *i*. Given $X_o(t)$ = [4,3,1,2,5,6] and $X_i(t)$ = [1,2,3,4,5,6], the distance between frogs *o* and *i* at the time *t* is determined as follows.

$$X_o(t) \sim X_f(t) = [4,3,2,1,5,6] \sim [1,2,3,4,5,6] = [4,3,2,1,0,0]$$

In Equation (18), the operator "⊙" refers to the binary step multiply operator. It determines the result of the multiplication of $X_o(t)$~X_i (*t*) with $RV_i(t) = [RV_{i,1}, \ldots, RV_{i,D}]$, in which each $RV_{i,k}(t)$ ($k \in [1,D]$) is a binary value. This operator works as shown in Equation (20).

$$\left(X_{o,k}(t) \sim X_{i,k}(t)\right) \odot RV_i(t) = \begin{cases} X_{o,k}(t) \sim X_{i,k}(t), & \text{if } RV_{i,k}(t) = 1 \\ 0, & \text{if } RV_{i,k}(t) = 0 \end{cases} \tag{20}$$

For example, given $RV_i(t)$ = [0,1,0,0,0,0], we can derive $D_i(t)$ as follows.

$$D_i(t) = [4,3,2,1,0,0] \odot [0,1,0,0,0,0] = [0,3,0,0,0,0].$$

The above example shows that more binary values of 1 in $RV_i(t)$ result in a larger leaping distance. Thus, we let each $RV_{i,k}(t)$ be determined by Equation (21).

$$RVI_{i,k}(t) = \begin{cases} 0, & \text{if } R1 \geq BR1_i(t) \\ 1, & \text{if } R1 < BR1_i(t) \end{cases} \tag{21}$$

R1 is a random number within the range [0,1], while $BR1_i(t)$ is a threshold controlling the generation of the binary values 0 or 1 in $RV_i(t)$ for the frog *i*. To enable a bigger jump for a more distant frog, Equations (22)–(24) are designed to initiate a bigger $BR1_i(t)$. First, Equation (22) counts the total number of shared elements between $X_i(t)$ and $X_o(t)$, denoted as $NSE_{i,o}(t)$. Then, Equation (23)

calculates $\omega_i(t)$, the maximum number of binary values of 1 currently allowed for the $RV_i(t)$ of frog i. Finally, Equation (24) is used to determine the value of $BR1_i\ (t)$.

$$NSE_{i,o}(t) = \sum_{i=1}^{D} [X_{o,k}(t) \approx X_{i,k}(t)] \tag{22}$$

$$\omega_i(t) = \mathrm{D} - NSE_{i,o}(t) - 2 \tag{23}$$

$$BR1_i(t) = \begin{cases} \frac{\omega_i(t)}{D}, & \text{if } NSE_{i,o}(t) < D-2 \\ 0, & \text{if } NSE_{i,o}(t) \geq D-2 \end{cases} \tag{24}$$

In Equation (22), the symbol "≈" is the comparing operator, and compares two elements with the same position in $X_o(t)$ and $X_i(t)$, and this operator works as follows.

$$X_{o,k}(t) \approx X_{i,k}(t) = \begin{cases} 1, & \text{if } X_{o,k}(t) = X_{i,k}(t) \\ 0, & \text{if } X_{o,k}(t) \neq X_{i,k}(t) \end{cases} \tag{25}$$

Given $X_o(t)$ = [4,3,2,1,5,6] and $X_i(t)$ = [1,2,3,4,5,6], we can derive $NSE_{i,o}(t)$ = 2, $\omega_i(t)$ = 2, and the $BR1_i(t)$ = 1/3. Finally, the $RV_i(t)$ and $D_i(t)$ can be derived by Equations (18) and (21), respectively, and Equation (26) can be used to derive the next position of the frog i.

$$X_i(t+1) = X_i(t) D_i(t) \tag{26}$$

In Equation (26), the symbol "ð" is as the leaping operator, which is used to determine the next position of a frog. It works on the basis of the following 4 steps:

(1) the operator takes the first non-zero value out of the $D_i(t)$ and replaces the value at the same position in $X_i(t)$,
(2) the replaced value takes the position of the non-zero value in $X_i(t)$,
(3) repeat the first and second steps until there are no non-zero values in $D_i(t)$,
(4) the operator copies the current $X_i(t)$ as $X_i(t + 1)$.

$$X_i(t+1) = X_i(t)\ \eth D_i(t) = [1,2,3,4,5,6]\ \eth [0,3,0,0,0,0] = [1,3,2,4,5,6] \tag{27}$$

Take Equation (27) as an example: the operator first takes the non-zero value "3" from $D_i(t)$ and replaces the "2" in $X_i(t)$. Then, the replaced value "2" takes the place of the "3" in $X_i(t)$. Because there are no non-zero elements in $D_i(t)$, the next position of the frog i will be $X_i(t + 1)$ = [1,3,2,4,5,6].

This example shows that a non-zero element in $D_i(t)$ can exchange two elements in $X_i(t)$. In terms of distance, the two frogs i and o are now considered to be closer due to their having four shared position elements, and the frog i possesses the following data: $NSE_{i,o}(t)$ = 4, $\omega_i(t)$ = 0, $BR1_i(t)$ = 0, and $RV_i(t)$ = [0,0,0,0,0], which will result in $D_i(t)$ = [0,0,0,0,0] being used to stop frog i jumping directly to the target frog o.

In addition to the self-adaptive leap mechanism, the ISFLA also makes it possible to include the following novel features:

- **Direct-jump prevention:** The direct-jump prevention will stop the generation of binary values of 1 for the $RV_i(t)$ of frog i in order to prevent it jumping directly to the target frog, as this would waste one local search. This will happen if the frog i is satisfied with the condition $NSE_{i,o}(t) \geq D - 2$. In Equation (22), the term −2 refers to the direct-jump prevention, which prevents frog i from jumping directly to the target frog o.
- **Neighborhood jump:** However, if $BR1_i(t) \neq 0$, there is always a chance for a frog to jump to the target frog directly. Thus, we let the ISFLA count the total number of binary values of 1 generated

so far for the $RV_i(t)$ of frog i; if the threshold ω_i is reached, instead of allowing a direct jump, the swap(p1, p2) operator, which switches two randomly selected elements in a frog's position vector, will be used to perform a neighborhood search. The neighborhood search avoids wasting local searches for frogs due to the imposition of direct-jump prevention.

- **Push jump:** For a frog i freed from direct-jump prevention but idling at its current position, which happens when $\sum_{k=1}^{D} RV_{i,k}(t) = 0$, the ISFLA will introduce one random binary value 1 into $RV_i(t)$ in order to push jump this frog.

4.5. The Self-Adaptive Mutation Mechanism

The ISFLA uses a self-adaptive mutation mechanism to vary the frogs in order to prevent them from being trapped in local optima. Two mutation operators used in the ISFLA are detailed below.

- **Swap Mutation (SM):** two gene values at two positions (p1 < p2) within a chromosome are randomly selected and then swapped.
- **Thoros Mutation (TM):** this kind of mutation randomly selects three gene positions p1, p2, and p3 (where p1 < p2 < p3) in a chromosome. Then, it changes the gene value of p1 to the position of p2; the gene value of p2 to the position of p3; and finally the gene value of p3 to the position of p1. TM has a higher degree of mutation than SM due to the greater number of mutated genes.

The trigger for the mutation of a frog depends on a random number $R2$ and a pre-defined mutation rate θ; if $R2 < \theta$, then the frog is mutated. Furthermore, the use of SM or TM depends on Equation (28), which measures the degree of similarity between $X_i(t)$ and $X_g(t)$.

$$SD_{i,g}(t) = NSE_{i,g}(t)/D \tag{28}$$

The higher the $SD_{i,g}(t)$, the higher the similarity. We assume that frogs with less similarity require greater variations so as to be able to escape the unfavorable position. Thus, if $SD_{i,g}(t) < \delta$, then TM is used in order to produce a bigger variation; otherwise, SM is used in order to generate a smaller variation. The δ, with its value within the range [0, 1], is a parameter controlling the use of TM or SM.

4.6. The Two Stage Procedure

In addition to the ROV technique, a two-stage approach was used to deal with the DCBAP.

4.6.1. The First Stage

This first stage focuses on generating a ship placement sequence.

4.6.2. The Second Stage

The second stage focuses on developing a feasible DCBAP solution after resolving overlaps of ships. This process contains three steps: place ships one by one into a berth plan, identify each overlap of ships, and resolve the overlap. These three tasks are repeated until all ships have been placed into the berth plan without any overlap.

Step 1: Place ships one by one into the berth plan

From the objective function, we know that if every ship is assigned its best position (ETA_j, d_j), this will lead to the optimal berth plan (i.e., $Z = 0$) due to there being no increased waiting or handling costs for the ships. However, if there are too many ships calling a container terminal at the same time, then this ideal arrangement becomes impossible due to the limited available resources in terms of both space and time. Repositioning overlapped ships is necessary, and repositioning around the best position will lead to the generation of optimal/near-optimal solutions.

Step 2: Identify overlaps

Figure 7 shows a berth plan with two ships that are overlapping in the dimensions of "space" and "time". Given that $a = (x_0^j, y_0^j)$ and $c = (x_1^j, y_1^j)$ are the coordinates of the lower-lef and upper-right corners of the ship j, while $a' = (x_0^k, y_0^k)$ and $c' = (x_1^k, y_1^k)$ are the coordinates of the lower-left and upper-right corners of the ship k, respectively, then Equation (29) is used to check whether the two ships are overlapping. If the following coordinate relationships hold simultaneously, then the two ships are concluded to be overlapping.

$$x_0^j < x_1^k,\ y_0^j < y_1^k,\ x_0^k < x_1^j,\ y_0^k < y_1^j \tag{29}$$

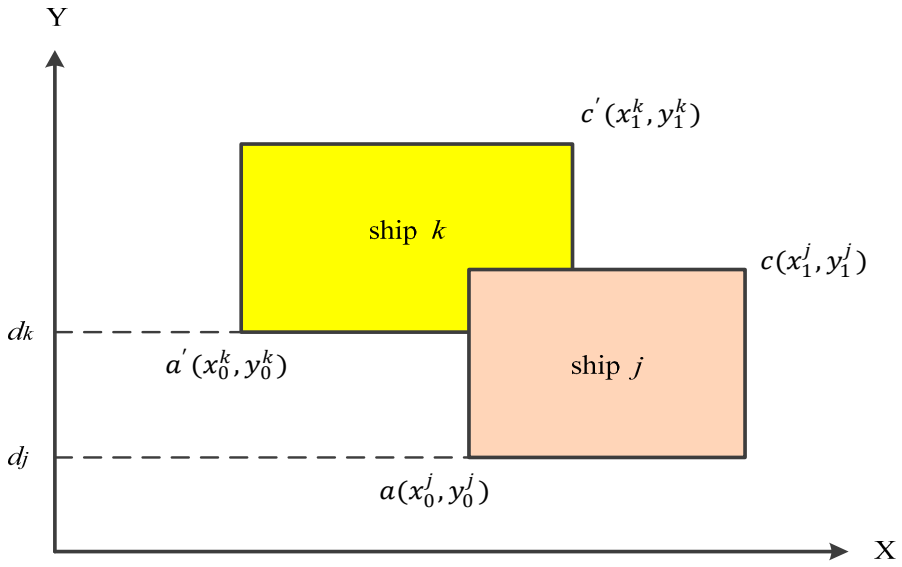

Figure 7. The coordinates of the corner points of ships j and k.

Step 3: Resolve the overlap

Resolving all overlaps of ships is necessary in order to make a feasible plan. In this research, a target ship is permitted to move in one of the following three directions: +Y, −Y, and +X, in order to resolve an overlap. However, while moving towards the +Y/−Y direction, handling times will be increased for the target ship due to the longer moving distances that trucks will have to move containers (Figure 8). When moving towards the +X direction, waiting time (cost) will be increased for the target ship (Figure 9). The lower the moving cost, the higher the priority.

For a target ship j moving toward the +Y/−Y direction, Equation (30) is used for estimating the increased distance ΔY_j.

$$\Delta Y_j = \begin{cases} l_j + \left| y_1^k - y_0^j \right| & \text{if moving towards} + Y \\ -\left| y_1^j - y_0^k \right| & \text{if moving towards} - Y \end{cases} \tag{30}$$

Equation (31) is the cost of increased handling times, where c_2 is the cost rate of handling time, $\beta = 2/(100 * 60)$, and $\Delta b_j = \left|\Delta Y_j\right|$.

$$\Delta C_Y(j) = \beta \cdot \Delta b_j \cdot c_2 = \frac{2\left|\Delta Y_j\right|}{100 * 60} \times c_2 \tag{31}$$

After repositioning, the coordinates of the lower-left and upper-right corners of the target ship j become $a = (x_0^j, y_0^j + \Delta Y_j)$ and $c = (x_1^j, y_1^j + \Delta Y_j)$, respectively.

For a target ship j moving towards the +X direction, Equation (32) is available for estimating the increased waiting time.

$$\Delta X_j = \left| x_0^j - x_1^k \right| \tag{32}$$

Equation (33) is the increased waiting cost, where c_1 is the cost rate of waiting time.

$$\Delta C_x(j) = \Delta X_j \times c_1 \tag{33}$$

After repositioning, the coordinates of the lower-left and upper-right corners of the target ship j become $a = (x_0^j + \Delta X_j, y_0^j)$ and $c = (x_1^j + \Delta X_j, y_1^j)$, respectively.

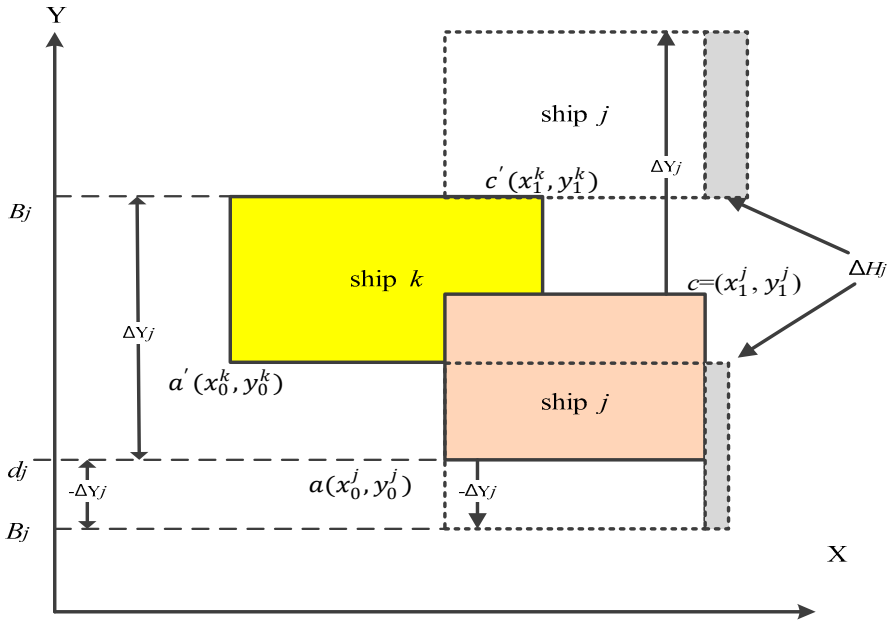

Figure 8. The movement of a target ship towards the +Y or −Y direction.

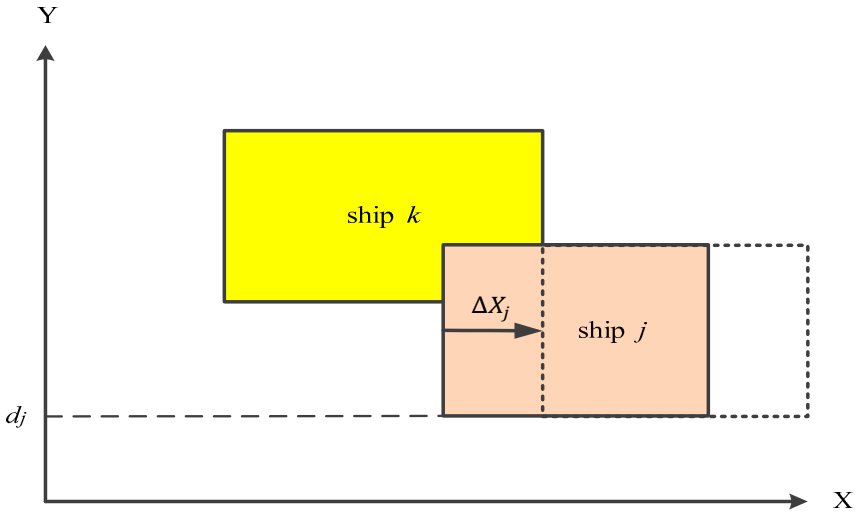

Figure 9. The movement of a target ship toward the +X direction.

4.7. The Main Flow Chart of the Two-Stage Approach

Figure 10 shows the main flow of the two-stage approach for dealing with the DCBAP. The first stage (steps 1 and 2) focuses on generating a ship placement sequence by using different approaches.

The second stage (steps 3 to 10) is a heuristic used to place ships into a berth plan and resolve overlaps of ships. Each step is detailed as follows.

Step 1: Generate data (including a_j, d_j, l_j and h_j) for all calling ships ($j = 1, \ldots, n$; where n is the total number of calling ships).

Step 2: Develop a placement sequence using the ISFLA (or FCFS, GA that are to be compared); set $s = 1$, where s indicates the placement order of a ship.

Step 3: Place the ship j (at the placement order s) into the berth plan, with the lower-left and upper-right corners being located at the coordinates $(x_0^j, y_0^j) = (a_j, c_j)$ and $(x_1^j, y_1^j) = (a_j + h_j, c_j + l_j)$, respectively; Set $k = 1$.

Step 4: Check whether the ship j with the placement order s has overlapped with the ship with the placement order s-k by using Equation (29). If "No", then go to Step 5; otherwise, go to Step 8.

Step 5: Check whether $s = k$. If "Yes", go to Step 6; otherwise $k = k + 1$ and go to Step 4 to check the next ship.

Step 6: Check whether the ship j remains with overlap(s) after repositioning. If "Yes", go to Step 10; otherwise, go to Step 7.

Step 7: Check whether $s = N$. If "Yes", go to Step 11; otherwise $s = s + 1$ and go to Step 3.

Step 8: Estimate the moving cost of each moving direction for repositioning the target ship with the placement order s using Equations (31) and (33). In addition, let $i = 1$ index the least-cost movement direction as the first priority; $i = 2$ index the second least-cost movement direction as the second priority; $i = 3$ index the most-cost movement direction as the 3rd priority.

Step 9: Repositioning the target ship towards the direction with the priority index i. Update the coordinates of the lower-left and upper-right corners of the target ship using Equation (30) or (32). Go to Step 5.

Step 10: Check whether the ship j has been ever overlapped with this ship or had an overlap resolved. If "Yes", set $i = i + 1$ (choose the next repositioning direction) and then go to Step 9; otherwise, go to Step 8.

Step 11: Check whether there is another iteration to perform. If "Yes", go to Step 2; otherwise go to Step 12.

Step 12: Stop.

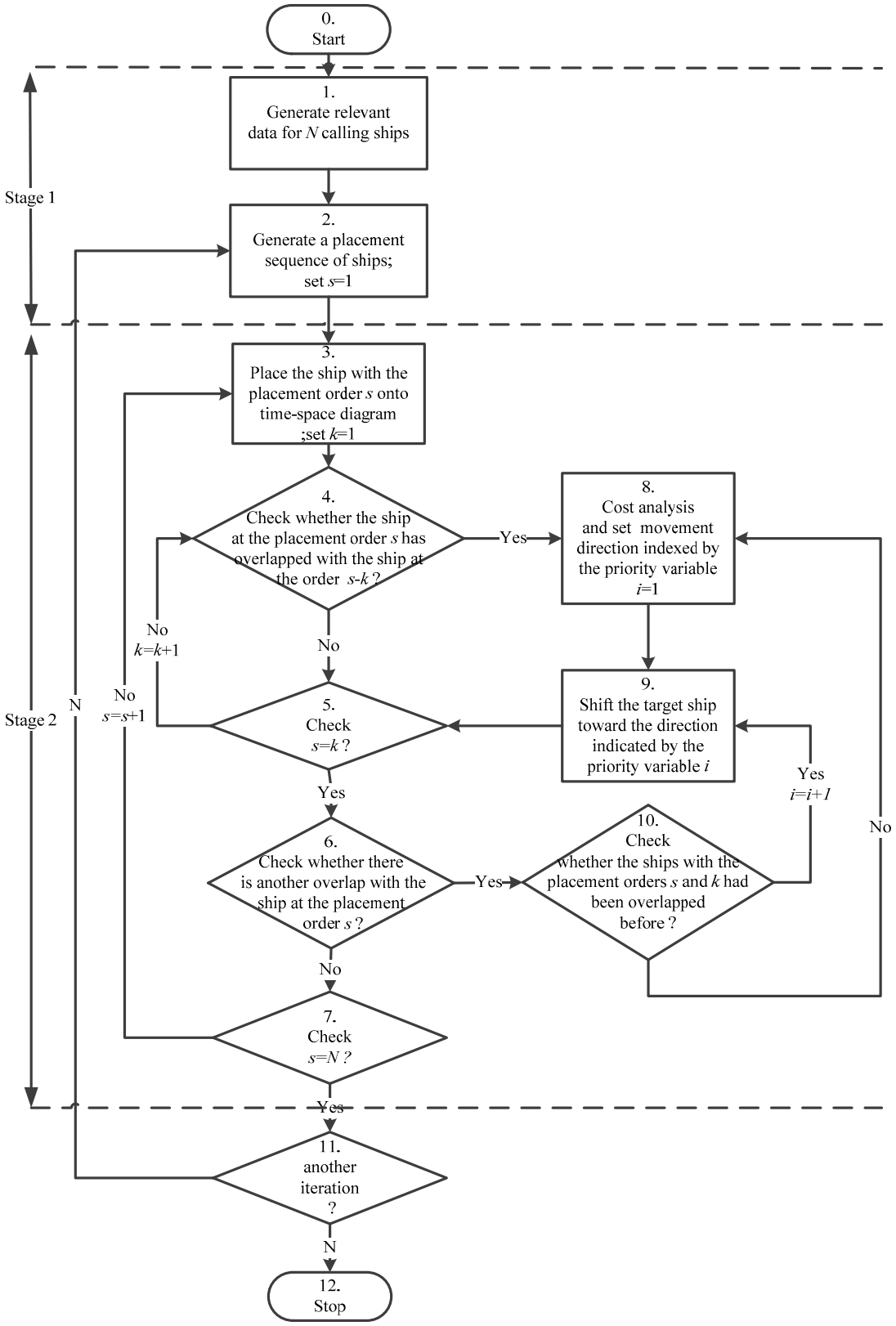

Figure 10. The main flow of the two-stage procedure.

5. Numerical Examples

Java is used as the programming language to implement the different approaches for the purpose of comparison. In the two-stage procedure, the FCFS, SFLA, and ISFLA were each employed in the

first stage, whereas a specific heuristic was employed in the second stage. For each comparison, a set of ship data was randomly generated by a computer, and experiments were conducted on a computer equipped with an Intel PENTIUM CPU (64 bit and 1.8 GHz) and 4G DRAM.

Section 5.1 details the parameter values set for the different approaches. Section 5.2 shows the results obtained from a small-sized experiment with $N = 10$. Section 5.3 shows the results obtained from a large-sized experiment with $N = 50$. Section 5.4 presents an analysis and discussion on the obtained experimental results.

5.1. Parameter Settings for Different Approaches

The following parameter settings were used for the experiments: the planning horizon (H) was set to one week, so that the ETAs of ships would be within the interval [0, 168] (hours); the quay length (L) was set to 1000 m; the length of a ship j (l_j) was a random value within the range [50, 200] (meters); the desired berthing position of a ship j (d_j) was a random value within the interval [0, L–l_j]; the cost rates c_1 and c_2 were set to USD 1000/h; and the $\beta = 2/(100 * 60)$ (per hour), which means the deviation of 100 m from the desired berthing position of a ship would increase handling time by 2 min.

The parameter values for the three different methods used in the first stage of the two-stage procedure are listed in Table 1. The parameter F indicates the total number of frogs in the swarm; the parameter m indicates the total number of memeplexes; the parameter *l_iter* specifies the number of local search for each frog; the parameter *iterations* indicates the total number of iterations; the parameter q indicates the total number of frogs in each memeplex; the parameter δ indicates the rate controlling the use of TM; the parameter θ indicates the mutation rate used; the symbol "-:" indicates not used.

Table 1. Parameter values for different approaches.

Parameters	Approaches		
	FCFS	**SFLA**	**ISFLA**
F	-	100	100
m	-	10	10
$n = F/m$	-	10	10
l_iter	-	5	5
iterations	1	150	150
q	-	5	-
S_m	-	N	N
Rm	-	-	0.3
δ	-	-	0.5
θ	-	-	0.5

F: total number of frogs; m: total number of memeplexes; *l_iter*: number of local search; *iterations*: total number of iterations; q: number of frogs in each memeplex; δ: the rate of using TM; θ the mutation rate; -: not used; N: total number of ships.

5.2. An Example of a Small-Sized Experiment

Table 2 shows the original and repositioned ship data of a small-sized experiment ($N = 10$). The field SID indicates the ID of a ship; the l_j shows the length of ship j; the ETA_j is Estimated Time of Arrival of ship j; d_j is the desired berthing position of ship j; ΔW_j shows the increased waiting time of ship j; ΔH_j shows increased handling time of ship j; (X_o, Y_o) and (X_1, Y_1) are coordinates of left-lower corner and right-upper corners of a ship j, respectively. Two ships (7 and 2) were repositioned to resolve overlaps. Ship 7 is moving towards the +Y direction to avoid overlapping with ship 9, while ship 2 is repositioning towards the −Y direction to avoid overlapping with ship 1. Figure 11 shows the berth plan with overlapping ships, while Figure 12 shows the feasible berth plan after resolving all overlaps of ships.

Table 2. Original and repositioning ship data after resolving overlaps (N = 10).

SID	l_j	ETA_j	d_j	Original				Repositioned				$c_1\Delta W_j$	$c_2\Delta H_j$
				X_0	Y_0	X_1	Y_1	X_0	Y_0	X_1	Y_1		
1	144	92	473.5	92	473.5	125.9	617.5	92.0	473.5	125.9	617.5	0	0
2	164	94.4	452.1	94.4	452.1	123	616.1	94.4	**309.5**	**123.1**	**473.5**	0	55
3	161	60.5	492.2	60.5	492.2	88.1	653.2	60.5	492.2	88.1	653.2	0	0
4	73	58.1	864.8	58.1	864.8	87.2	937.8	58.1	864.8	87.2	937.8	0	0
5	56	19.3	814.9	19.3	814.9	39.3	870.9	19.3	814.9	39.3	870.9	0	0
6	112	55.8	67	55.8	67	71	179	55.8	67.0	71.0	179.0	0	0
7	56	105.7	534.6	105.7	534.6	140.9	590.6	105.7	**675.9**	**140.9**	**731.9**	0	47
8	84	64.1	249.5	64.1	249.5	70.9	333.5	64.1	249.5	70.9	333.5	0	0
9	193	134.6	482.9	134.6	482.9	176.8	675.9	134.6	482.9	176.8	675.9	0	0
10	126	10.9	580.5	10.9	580.5	56.7	706.5	10.9	580.5	56.7	706.5	0	0
Total												0	102

SID: Ship ID; l_j: length of ship j; ETA_j: Estimated Time of Arrival of ship j; d_j: desired berthing position of ship j; ΔW_j : increased waiting time of ship j; ΔH_j: increased handling time of ship j.

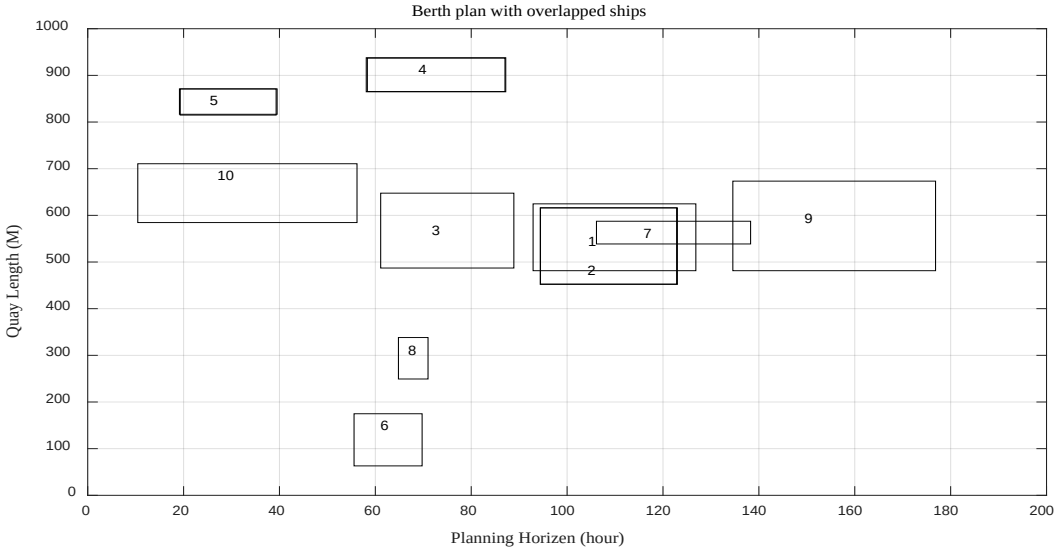

Figure 11. The berth plan with overlapped ships (N = 10 ships).

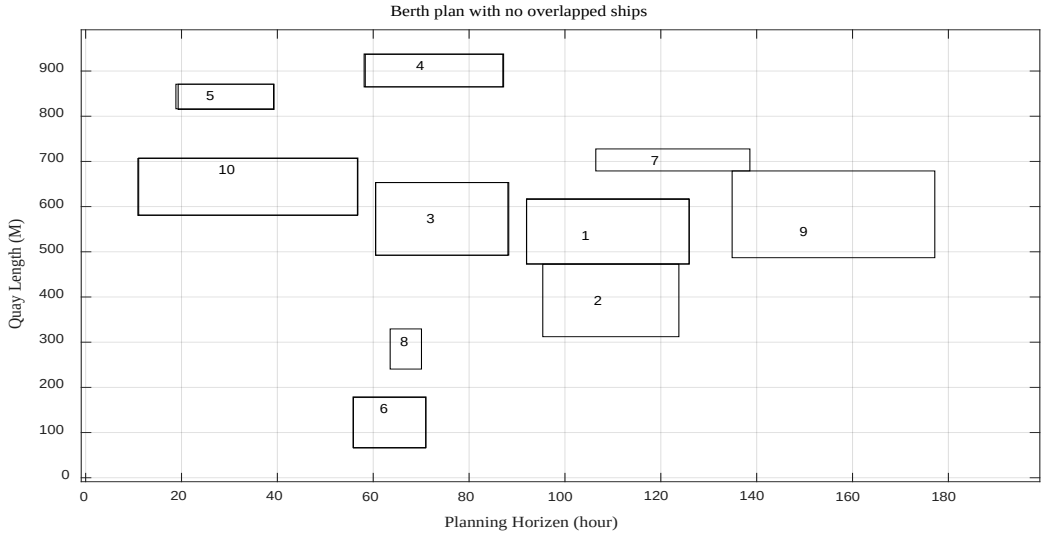

Figure 12. The berth plan with no overlapped ships (N = 10 ships).

5.3. An Example of a Large-Sized Experiment

Table 3 shows the original and repositioned data of a large-sized experiment with $N = 50$ ships. The 1st column shows the ship ID (SID); the 2nd column shows the ship length; the 3rd column shows the Estimated Time of Arrival of ship j (ETA_j); the 4th column shows the desired berthing position of ship j (d_j); the 5th column shows the original coordinates of ship j (X_0, Y_0, X_1, Y_1); the 6th column shows the repositioned coordinates of ship j (X_0, Y_0, X_1, Y_1); the 7th column shows the increased waiting time of ship j ($c_1\Delta W_j$); the 8th column shows the increased handling time of ship j ($c_2\Delta H_j$). Figure 13 shows a berth plan that is unfeasible due to the overlapping of the ships, based on the original coordinates of ships. To derive a feasible solution, it is necessary to resolve all overlaps of ships. Figure 14 shows the feasible berth plan obtained from the ISFLA after resolving all overlaps among the ships. In Table 3, the bolded figures indicate updated coordinates for those target ships that were repositioned to avoid overlapping.

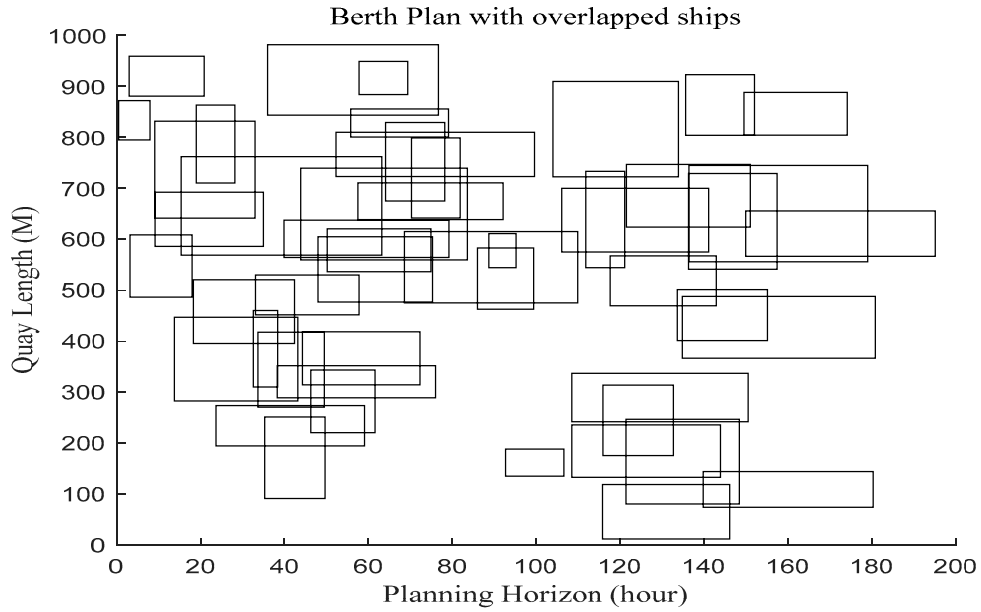

Figure 13. The infeasible berth plan before resolving all overlaps of ships ($N = 50$ ships).

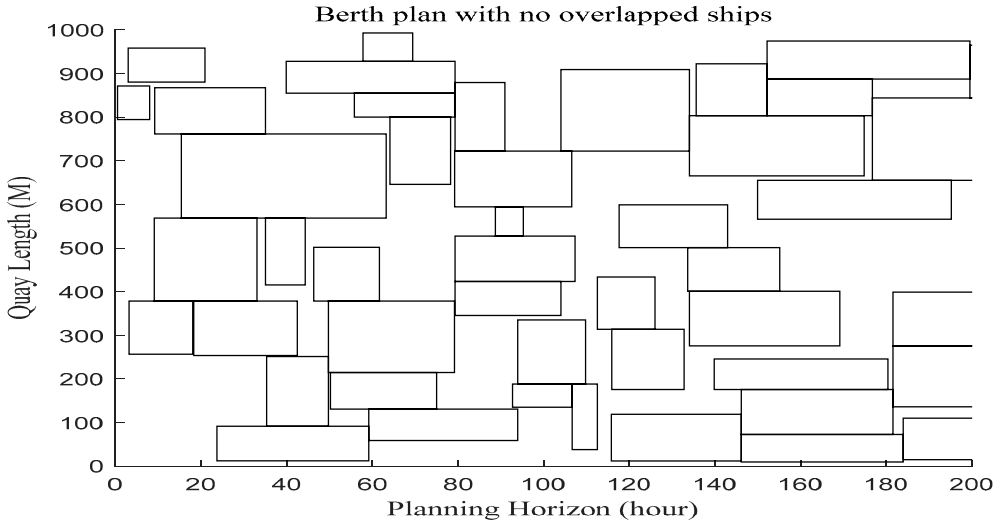

Figure 14. The feasible berth plan after resolving all overlaps of ships ($N = 50$ ships).

Table 3. The ship data obtained from a large-sized experiment (N = 50).

SID	l_j	ETA_j	d_j	Original				Repositioned				$c_1\Delta W_j$	$c_2\Delta H_j$
				X_0	Y_0	X_1	Y_1	X_0	Y_0	X_1	Y_1		
1	100	133.6	400.8	133.6	400.8	155.1	500.8	133.6	400.8	155.1	500.8	0	0
2	123	121.5	623.5	121.5	623.5	151.0	746.5	**181.5**	**275.8**	**211.1**	**398.8**	60,026	115
3	189	111.8	543.8	111.8	543.8	121.1	732.8	**219.4**	**466.0**	**228.7**	**655.0**	107,550	26
4	122	3.2	486.3	3.2	486.3	18.0	608.3	3.2	**256.5**	**18.1**	**378.5**	0	54
5	87	52.3	722.5	52.3	722.5	99.6	809.5	**152.1**	**887.4**	**199.5**	**974.4**	99,817	55
6	53	92.7	135.0	92.7	135.0	106.6	188.0	92.7	**135.0**	**106.6**	**188.0**	0	40
7	65	57.8	883.5	57.8	883.5	69.4	948.5	57.8	**928.0**	**69.4**	**993.0**	0	15
8	106	9.2	585.7	9.2	585.7	35.0	691.7	9.2	**761.5**	**35.1**	**867.5**	0	59
9	180	43.9	559.1	43.9	559.1	83.6	739.1	**222.9**	**275.8**	**262.7**	**455.8**	179,036	95
10	157	70.3	641.4	70.3	641.4	81.9	798.4	**79.3**	**722.2**	**90.9**	**879.2**	8997	27
11	78	3.0	880.4	3.0	880.4	20.9	958.4	3.0	**880.4**	**20.9**	**958.4**	0	26
12	123	46.3	220.2	46.3	220.2	61.6	343.2	46.3	**378.5**	**61.7**	**501.5**	0	53
13	160	35.3	91.2	35.3	91.2	49.7	251.2	35.3	**91.2**	**49.7**	**251.2**	0	47
14	104	44.3	314.0	44.3	314.0	72.3	418.0	**79.3**	**423.2**	**107.3**	**527.2**	34,997	24
15	121	134.8	366.5	134.8	366.5	180.8	487.5	**199.5**	**844.0**	**245.6**	**965.0**	64,672	159
16	125	18.3	395.1	18.3	395.1	42.4	520.1	18.3	**253.5**	**42.5**	**378.5**	0	77
17	188	136.3	540.7	136.3	540.7	157.4	728.7	219.4	655.0	240.5	843.0	83,050	38
18	119	135.6	803.4	135.6	803.4	152.0	922.4	135.6	**803.4**	**152.1**	**922.4**	0	117
19	98	117.6	469.1	117.6	469.1	142.9	567.1	117.6	**500.8**	**142.9**	**598.8**	0	11
20	84	50.2	536.0	50.2	536.0	74.9	620.0	50.2	**130.5**	**75.0**	**214.5**	0	109
21	103	108.5	132.6	108.5	132.6	143.9	235.6	**146.1**	**72.5**	**181.5**	**175.5**	37,605	21
22	140	68.6	474.8	68.6	474.8	109.9	614.8	**181.5**	**135.8**	**222.9**	**275.8**	112,926	110
23	63	38.3	288.7	38.3	288.7	76.0	351.7	**146.1**	**9.5**	**183.9**	**72.5**	107,805	93
24	190	9.1	641.1	9.1	641.1	33.0	831.1	9.1	**378.5**	**33.1**	**568.5**	0	63
25	55	55.8	800.0	55.8	800.0	79.1	855.0	55.8	**800.0**	**79.2**	**855.0**	0	90
26	72	57.5	638.1	57.5	638.1	92.1	710.1	**59.1**	**58.5**	**93.9**	**130.5**	1646	189
27	154	64.1	674.6	64.1	674.6	78.2	828.6	64.1	**646.0**	**78.3**	**800.0**	0	51
28	67	88.7	543.7	88.7	543.7	95.2	610.7	88.7	**527.2**	**95.2**	**594.2**	0	22
29	125	106.1	574.6	106.1	574.6	141.1	699.6	**134.0**	**275.8**	**169.1**	**400.8**	27,922	108
30	79	23.7	194.4	23.7	194.4	59.1	273.4	23.7	**12.2**	**59.1**	**91.2**	0	46
31	78	33.1	451.5	33.1	451.5	57.8	529.5	**79.3**	**345.2**	**104.0**	**423.2**	46,197	4
32	164	13.8	282.5	13.8	282.5	43.2	446.5	**49.7**	**214.5**	**79.2**	**378.5**	35,947	14
33	70	139.8	74.1	139.8	74.1	180.3	144.1	139.8	**175.5**	**180.3**	**245.5**	0	34
34	187	104.0	722.2	104.0	722.2	133.9	909.2	104.0	**722.2**	**134.0**	**909.2**	0	122
35	107	115.8	11.7	115.8	11.7	146.1	118.7	115.8	**11.7**	**146.1**	**118.7**	0	5
36	95	108.5	241.8	108.5	241.8	150.5	336.8	**183.9**	**14.8**	**226.0**	**109.8**	75,398	52
37	89	149.9	566.0	149.9	566.0	195.1	655.0	149.9	**566.0**	**195.1**	**655.0**	0	35
38	147	33.7	270.2	33.7	270.2	49.5	417.2	**93.9**	**188.0**	**109.8**	**335.0**	60,235	28
39	166	121.4	80.6	121.4	80.6	148.4	246.6	**222.9**	**109.8**	**249.9**	**275.8**	101,536	10
40	138	36.0	843.1	36.0	843.1	76.7	981.1	**134.0**	**665.4**	**174.8**	**803.4**	98,022	87
41	153	19.0	709.8	19.0	709.8	28.2	862.8	**35.1**	**415.5**	**44.3**	**568.5**	16,059	53
42	193	15.4	568.5	15.4	568.5	63.2	761.5	15.4	**568.5**	**63.2**	**761.5**	0	41
43	138	115.9	175.5	115.9	175.5	132.7	313.5	115.9	**175.5**	**132.8**	**313.5**	0	61
44	77	0.5	794.5	0.5	794.5	8.0	871.5	0.5	**794.5**	**8.0**	**871.5**	0	19
45	128	48.0	476.6	48.0	476.6	75.3	604.6	**79.2**	**594.2**	**106.5**	**722.2**	31,190	52
46	73	39.9	563.9	39.9	563.9	79.2	636.9	39.9	**855.0**	**79.3**	**928.0**	0	97
47	120	86.0	462.5	86.0	462.5	99.4	582.5	**112.5**	**313.5**	**126.0**	**433.5**	26,528	50
48	150	32.6	309.9	32.6	309.9	38.4	459.9	**106.6**	**38.0**	**112.5**	**188.0**	74,040	88
49	84	149.5	803.6	149.5	803.6	174.1	887.6	**152.1**	**803.4**	**176.7**	**887.4**	2,617	0
50	189	136.4	555.5	136.4	555.5	179.0	744.5	**176.7**	**655.0**	**219.4**	**844.0**	40,317	33
Total												1,534,135	2825

Table 4 shows the results obtained from different approaches with different problem sizes. The Z indicates the objective function value; the T indicates the computational times; the GAP indicates the gap in percentage compared to the ISFLA. The experimental results show that, on average, the ISFLA exhibits better performance in terms of solution quality. Figure 15 gives a graph with the average Zs obtained from different approaches under different problem sizes (N = 10, 20, 30, 40, 50 and 60).

Table 4. The results obtained from different approaches at different problem sizes.

	FCFS			SFLA			ISFLA	
N	**Z**	**T**	**Gap (%)**	**Z**	**T**	**Gap (%)**	**Z**	**T**
10								
1	91.5	1.2	49.4	61.3	232.3	0.0	61.3	277.8
2	378.0	1.1	12.3	336.7	235.3	0.0	336.7	258.4
3	62.6	1.0	19.8	58.3	253.1	11.5	52.3	263.4
4	300.7	1.2	0.0	300.7	237.6	0.0	300.7	229.7
5	196.1	1.1	70.4	136.3	244.7	18.5	115.1	288.7
6	111.4	1.1	21.5	91.6	236.8	0.0	91.6	254.8
7	45.7	1.1	43.6	31.8	242.2	0.0	31.8	242.3
8	44.7	1.1	0.0	46.4	225.6	3.8	44.7	252.7
9	115.2	0.9	18.1	104.6	238.2	7.2	97.6	255.0
10	178.4	1.1	26.0	159.3	214.3	12.6	141.5	239.3
Avg.	152.4	1.1	19.7	132.7	236.0	4.2	127.3	256.2
20								
1	15,420.1	1.3	44.3	11,857.1	477.7	11.0	10,682.7	532.3
2	6534.0	1.4	11.9	6358.1	473.2	8.9	5837.8	589.0
3	20,298.3	1.3	45.7	14,688.2	476.4	5.4	13,931.4	555.0
4	190.3	1.3	15.6	186.4	489.3	13.2	164.6	634.1
5	13,228.6	1.2	54.1	9162.4	493.2	6.7	8584.6	505.2
6	24,208.8	1.3	12.4	23,830.8	483.1	10.6	21,539.5	531.1
7	12,199.4	1.3	73.4	8023.5	486.1	14.0	7036.7	503.3
8	1184.4	1.4	23.0	1072.2	477.3	11.4	962.7	504.0
9	2617.2	1.3	(9.1)	3521.1	492.2	22.3	2878.1	519.0
10	16,312.8	1.5	60.7	11,826.3	472.2	16.5	10,148.6	534.8
Avg.	11,219.4	1.3	37.2	9052.6	482.1	10.7	8176.7	540.8
30								
1	205,881.1	1.7	38.9	172,451.7	636.3	16.3	148,249.0	707.1
2	314,277.2	1.7	7.1	316,217.2	604.3	7.8	293,372.9	774.5
3	217,210.3	1.4	24.4	200,271.8	625.3	14.7	174,589.9	763.5
4	73,850.2	1.8	12.8	66,985.2	614.1	2.3	65,480.4	769.0
5	208,806.1	1.7	54.6	156,281.1	595.2	15.7	135,101.2	790.8
6	267,383.0	1.5	23.1	227,438.4	593.9	4.7	217,219.7	734.8
7	66,336.9	1.6	58.2	45,473.9	624.2	8.5	41,919.3	727.2
8	63,760.8	1.5	50.6	48,353.4	640.2	14.2	42,330.5	921.0
9	426,888.0	1.6	27.7	410,358.4	619.4	22.8	334,170.2	903.2
10	203,769.2	1.5	89.1	117,136.1	644.1	8.7	107,743.6	730.7
Avg.	204,816.3	1.6	31.3	176,096.7	619.7	12.9	156,017.7	782.2
40								
1	546,703.7	1.7	38.0	462,767.2	1859.2	16.8	396,146.0	2400.5
2	175,292.0	1.7	1.3	199,519.4	1974.2	15.3	173,015.2	2451.3
3	705,100.7	1.7	19.4	672,451.2	1833.9	13.8	590,775.2	2527.8
4	357,388.6	1.7	30.1	305,318.6	1968.2	11.1	274,705.5	2537.7
5	706,807.6	1.7	2.8	716,680.6	1992.7	4.2	687,587.7	2430.5
6	1,562,685.5	1.8	50.0	1,132,683.5	1934.2	8.7	1,041,642.0	2801.5
7	667,840.4	1.7	36.2	528,633.0	1917.2	7.8	490,389.6	2566.2
8	1,371,378.0	1.7	44.8	1,167,187.2	1906.5	23.2	947,068.4	2729.9
9	1,060,564.8	1.8	33.0	926,136.3	1939.1	16.2	797,123.9	2618.1
10	755,054.9	1.7	64.1	643,065.9	1989.3	39.8	460,135.2	2715.9
Avg.	790,881.6	1.7	35.0	675,444.3	1931.5	15.3	585,858.9	2578.0

Table 4. *Cont.*

	FCFS			SFLA			ISFLA	
N	**Z**	**T**	**Gap (%)**	**Z**	**T**	**Gap (%)**	**Z**	**T**
50								
1	2,592,060.5	1.9	(0.1)	3,070,361.5	2456.1	18.3	2,595,948.5	3223.3
2	2,648,510.2	2.0	14.1	2,644,852.3	2341.4	13.9	2,321,576.5	3530.5
3	2,785,803.8	2.0	18.9	2,843,812.2	2442.1	21.3	2,343,809.5	3098.9
4	2,507,274.3	1.9	27.9	1,973,747.2	2543.2	0.7	1,960,933.9	2977.0
5	1,636,155.9	1.9	17.9	1,656,152.2	2523.5	19.3	1,388,291.0	2815.0
6	1,475,022.1	2.1	8.9	1,555,821.3	2422.4	14.9	1,354,645.0	2988.4
7	2,293,968.9	2.0	74.7	1,593,912.9	2366.2	21.4	1,313,400.6	2999.6
8	1,555,521.0	2.4	6.7	1,865,321.0	2229.2	27.9	1,458,157.2	2931.3
9	2,291,265.0	2.0	4.8	2,471,635.0	2553.1	13.0	2,186,907.5	3142.6
10	1,756,002.7	2.7	32.3	1,553,631.6	2323.2	17.1	1,327,251.5	3214.5
Avg.	2,154,158.4	2.1	18.0	2,122,924.7	2420.0	16.3	1,825,092.1	3092.1
60								
1	4,709,809.5	7.5	35.6	4,299,681.4	5213.3	23.8	3,472,396.1	5822.7
2	2,953,025.4	7.6	16.4	2,854,712.4	5327.0	12.5	2,536,974.7	5538.9
3	4,767,009.0	7.6	30.2	4,336,401.3	5282.1	18.4	3,662,545.9	5397.2
4	3,228,876.4	7.6	38.9	2,686,814.4	5177.0	15.6	2,324,015.3	3254.3
5	5,007,260.9	7.6	3.2	5,196,307.9	5153.0	7.1	4,852,986.7	3432.5
6	4,287,543.9	7.3	43.0	3,766,731.2	5431.2	25.6	2,997,992.0	5497.2
7	3,798,522.1	7.9	(4.1)	4,336,601.6	5369.6	9.5	3,959,739.6	6162.2
8	4,813,647.0	8.9	10.3	5,254,204.0	5161.3	20.4	4,364,050.0	2824.0
9	3,559,317.1	8.1	44.0	3,153,893.1	5134.1	27.6	2,470,898.9	3330.4
10	4,990,244.7	8.3	3.0	5,770,216.7	5284.1	19.1	4,842,827.4	7417.4
Avg.	4,211,525.6	7.8	18.7	4,165,556.4	5253.3	17.4	3,548,442.7	4867.7

Z: Z value; T: Time (s); GAP: the gap in percentage compared to the ISFLA.

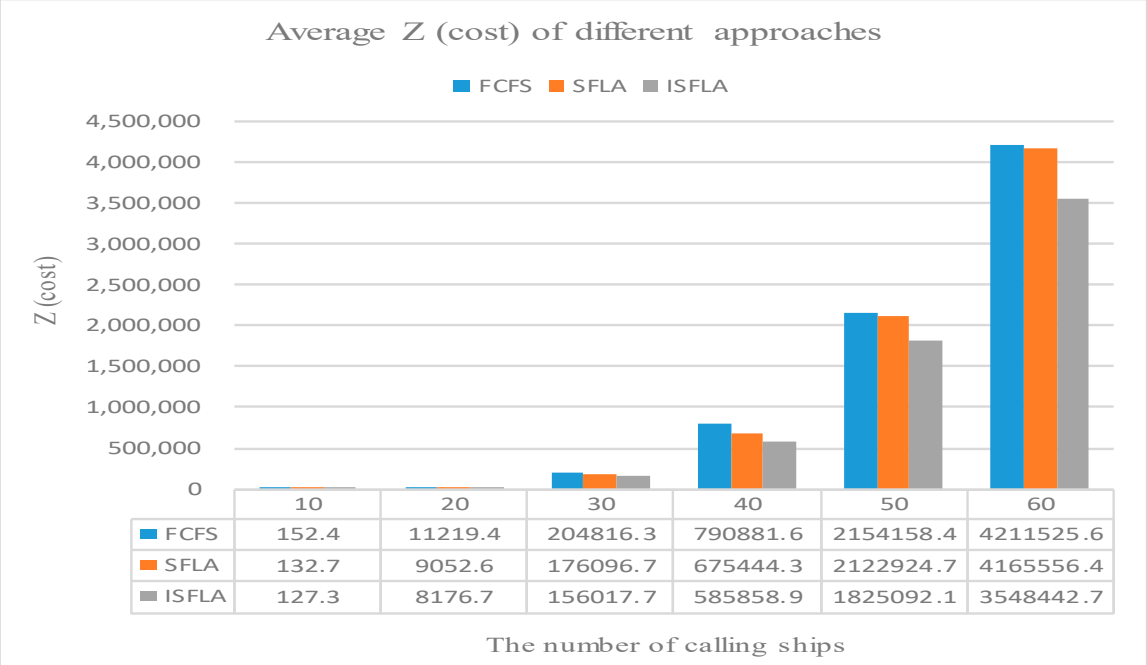

Figure 15. The average Z (cost) of different approaches under different coming ships.

5.4. Analysis of Results and Discussion

5.4.1. Analysis of Results

(1) Different methods used in the first stage of the two-stage procedure can lead to different planning results.
(2) Among the FCFS, SFLA, and ISFLA, the experimental results showed the ISFLA outperformed the two others, but at the cost of longer computational times.
(3) The FCFS rule is simple and fast in finding a solution. However, with an expected lower waiting time (cost), the FCFS rule lacks the capacity to improve solution quality continuously.
(4) The increase in iterative runs enables ISFLA to explore a wider solution space. This helps improve solution quality, but at the cost of longer computational times.
(5) When the ISFLA repositions a target ship towards the +Y/−Y direction, the handling time for the target ship will increase; when repositioning a target ships towards the +X direction, the waiting time for the target ship will increase. Since the movement towards the +Y/−Y direction results in a smaller increase in handling cost for a target ship, these two directions will have a higher priority, and this can result in better utilization of quay space. However, due to the limited quay space, movement towards the +X direction is sometimes necessary for a target ship. Such repositioning of ships leads to the finding of a near or the optimal solution.
(6) As the average computational times required for the ISFLA to solve a problem size of 60 calling ships is about 1.4 h, the ISFLA is thus concluded to be applicable in practice.
(7) For the ISFLA, increasing the total number of iterative runs usually leads to the finding of a higher-quality solution, due to the wider exploration of the solution space.
(8) Both cost factors c1 and c2 are found able to affect the selection of repositioning direction for a target ship. They can affect the choices of moving direction toward +Y, −Y, or +X for a target ship. In this research, the two cost rates are set to USD 1000/h.

5.4.2. Discussion

(1) To our best knowledge, this is the first research employing SFLA for dealing with the BAP in a seaport container terminal. The SFLA also shows potential in dealing with problems in the yard and landside areas, in addition to the seaside area.
(2) Some early CBAP studies, such as those by Lim [4], Li et al. [28], and Guan et al. [29], assumed static berth allocation and independent handling times of ships in their berthing positions. In such studies, the problems to be solved are equivalent to the CSP with fixed orientation of placed items. In contrast, this present research considers the dynamic nature of ship arrivals and assumes that the handling times of ships are dependent on their berthing positions.
(3) This present research also differs from the studies of Park and Kim [30,31] and Kim and Moon [5], as those studies assumed fixed handling times for ships. In addition, in a strict sense, the BAP solved in the study of Park and Kim [31] was downgraded to a static version of the problem due to ships being able to be served before their ETAs, albeit with some penalty cost imposed in the objective function [1].
(4) The CBAP solved in this study is quite different from the conventional CSP due to the following three differences: first, in the CBAP a variable handling time is considered; second, the CBAP considers the dynamic nature of ship arrivals; third, the CBAP considers the orientation of the ship placed in a berth plan.
(5) Ref. [1] also proposed a two-stage approach for dealing with the CBAP. The first stage generates an initial solution for the DBAP, and then this solution is further transformed into a feasible one by repositioning overlapped and sparsely located ships at the second stage. However, this approach appears to be rigorous, because to find the final solution to the CBAP, it is first necessary to find the solution to the DBAP. In addition, in the first stage, it is necessary to set a minimum

and maximum berth length for the DBAP, and to prepare several intermediate berth lengths for this approach. Such calculations for the setting of parameters take a long time. In contrast, our approach appears to be simpler, as it finds the solution to the CBAP directly.

(6) In [1], the proposed heuristic arranges ships in ascending order of their handling time, which is similar to the FCFS rule, before resolving overlaps of ships. This differs from our approach, which creates alternative placement sequences of ships by using the ISFLA so as to explore more alternative solutions.

(7) Based on a simulated annealing approach, the metaheuristic proposed in [34] is capable of exploring alternative solutions by means of an R parameter that denotes the number of sequences of the neighborhood search. However, the sequence approach proposed in [34] is different from the ISFLA proposed in this research. In addition, [34] did not describe the resolution of overlapping ships.

(8) Compared with [1,34], our approach appears to be simpler and more practically applicable. However, it is too early to say that our approach is better than the approaches proposed in [1,34], as more experiments are required for comparison. Nevertheless, according to the objective defined and used in this research, we know that the optimal berthing position and time for a ship in a berth plan are the d_j and ETA_j of the ship. As the ISFLA will assign a ship to or around the optimal berthing position (d_j) and start working time (ETA_j) for each ship, we can conclude that the resulting berth plan will be the optimal solution (when $Z \geq 0$), or a near-optimal solution (when $Z > 0$). As the average computational times for our approach when solving a problem with 60 ships was about 1.4 h; this shows the applicability of the proposed method in practice.

(9) This ISFLA is able to improve the simplicity of simple heuristics while avoiding the computational intractability of exact approaches by tuning the total number of iterative runs.

6. Conclusions and Future Research Direction

The *dynamic* and *continuous* berth allocation problem is a seaside operational problem routinely faced by container terminal planners. As this problem can considerably affect the productivity of a container terminal, a better solution for this problem is needed. In this research, we have proposed a two-stage approach for dealing with this problem. In the first stage, the First Come First Served rule (FCFS), Shuffled Frog-Leaping Algorithm (SFLA), and Improved Shuffled Frog-Leaping Algorithm (ISFLA) were each employed in order to generate alternative ship placement sequences that could serve as inputs for the second stage. In the second stage, a specific heuristic was used to assign berthing positions and resolve overlaps of ships in order to develop a feasible solution. The experimental results showed that the Improved Shuffled Frog-Leaping Algorithm outperformed the other two methods in terms of solution quality. The contributions of this research are highlighted below:

(1) A novel Shuffled Frog-Leaping Algorithm-based approach (i.e., the ISFLA) was developed to deal with the *dynamic* and *continuous* berth allocation problem. To our best knowledge, this kind of approach has never been used for this purpose in a container terminal.

(2) In addition to the Improved Shuffled Frog-Leaping Algorithm, a heuristic was developed, in the second stage of a two-stage procedure, for placing ships and resolving overlaps of ships for the development of a feasible solution.

(3) The Java programing language was used to implement the two-stage procedure, as it facilitates the generation of BAP solutions for practical use. Our small-sized experiments showed that the proposed approach was capable of finding optimal/near-optimal solutions in terms of the objective function defined in this research. In addition, our experiments demonstrated the feasibility of the ISFLA with respect to computational time for solving a large-sized problem of up to 60 ships.

In this research, we achieved the first step in dealing with the *dynamic* and *continuous* berth allocation problem (DCBAP) by using an Improved Shuffled Frog-Leaping Algorithm (ISFLA). The next step could consist of applying the Improved Shuffled Frog-Leaping Algorithm to deal with

the dynamic and continuous berth allocation problem and quay crane assignment problem (QCAP) simultaneously. Using ISFLA to deal with the quay crane scheduling problem (QCSP) is another research direction. In addition, a comparison between the Improved Shuffled Frog-Leaping Algorithm and other approaches proposed in past research (such as the [1,34]) could further be conducted. In addition, improving the heuristic used in the second stage of the two-stage procedure is another direction of research.

Author Contributions: H.-P.H. guided the research direction, developed the algorithm and found the solutions; T.-L.C. summarized and analyzed the data, revised and edited this paper. All authors have contributed to this research.

Funding: This research was supported by the Ministry of Science and Technology of Taiwan under the grant MOST 107-2410-H-992-037.

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

References

1. Imai, A.; Sun, X.; Nishimura, E.; Papadimitriou, S. Berth allocation in a container port: Using a continuous location space approach. *Transp. Res. Part B* **2005**, *39*, 199–221. [CrossRef]
2. Vis, I.F.A.; de Koster, R. Transshipment of container at a container terminal: An overview. *EJOR* **2003**, *147*, 1–16. [CrossRef]
3. Bierwirth, C.; Meisel, F. A survey of berth allocation and quay crane scheduling problems in container terminals. *EJOR* **2010**, *202*, 615–627. [CrossRef]
4. Lim, A. The berth scheduling problem. *Oper. Res. Lett.* **1998**, *22*, 105–110. [CrossRef]
5. Kim, K.H.; Moon, K.C. Berth scheduling by simulated annealing. *Transp. Res. Part B* **2003**, *37*, 541–560. [CrossRef]
6. Salido, M.A.; Mario, R.M.; Barber, F. A decision support system for managing combinatorial problems in a container terminal. *Knowl.-Based Syst.* **2012**, *29*, 63–74. [CrossRef]
7. Hsu, H.P. A HPSO for solving dynamic and discrete berth allocation problem and dynamic quay crane assignment problem simultaneously. *Swarm Evol. Comput.* **2016**, *27*, 156–168. [CrossRef]
8. Dorigo, M.; Gambardella, L.M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evol. Comput.* **1997**, *1*, 53–66. [CrossRef]
9. Gravel, M.; Price, W.L.; Gagné, C. Scheduling Continuous Casting of Aluminium using a Multiple Objective Ant Colony Optimization Metaheuristic. *Eur. J. Oper. Res.* **2002**, *143*, 218–229. [CrossRef]
10. Solnon, C.; Fenet, S. A study of ACO capabilities for solving the Maximum Clique Problem. *J. Heuristics* **2006**, *12*, 155–180. [CrossRef]
11. Bullnheimer, R.F.; Hartl, C. Strauss, an Improved Ant System Algorithm for the Vehicle Routing Problem. *Ann. Oper. Res.* **1999**, *89*, 319–328. [CrossRef]
12. Han, X.L.; Lu, Z.Q.; Xi, L.F. A proactive approach for simultaneous berth and quay crane scheduling problem with stochastic arrival and handling time. *EJOR* **2010**, *207*, 1327–1340. [CrossRef]
13. Chang, D.F.; Jiang, Z.H.; Yan, W.; He, J.L. Integrating berth allocation and quay crane assignments. *Transp. Res. Part E* **2010**, *46*, 975–990. [CrossRef]
14. Liang, C.J.; Guo, J.Q.; Yang, Y. Multi-objective hybrid genetic algorithm for quay crane dynamic assignment in birth allocation planning. *J. Intell. Manuf.* **2011**, *22*, 471–479. [CrossRef]
15. Rodriguez-Molins, M.; Ingolotti, L.; Barber, F.; Salido, M.A.; Sierra, M.R.; Puente, J. A genetic algorithm for robust berth allocation and quay crane assignment. *Prog. Artif. Intell.* **2014**, *2*, 177–192. [CrossRef]
16. Yoshida, H.; Kawata, K.; Fukuyama, Y.; Takayama, S.; Nakanishi, Y. A particle swarm optimization for reactive power and voltage control considering voltage security assessment. *IEEE Trans. Power Syst.* **2000**, *15*, 1232–1239. [CrossRef]
17. Janson, S.; Middendorf, M. A hierarchical particle swarm optimizer and its adaptive variant. *IEEE Trans. Syst. Man Cybernatics Part B Cybernatics* **2005**, *35*, 1272–1282. [CrossRef]
18. Bierwirth, C.; Meisel, F. A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. *Eur. J. Oper. Res.* **2015**, *244*, 675–689. [CrossRef]

19. Eusuff, M.M.; Lansey, K.E. Optimization of Water Distribution Network Design using the Shuffled Frog Leaping Algorithm. *J. Water Resour. Plan. Manag.* **2003**, *129*, 210–225. [CrossRef]
20. Merz, P.; Freisleben, B. A genetic local search approach to quadratic assignment. In Proceedings of the 7th International Conference on Genetic Algorithm, East Lansing, MI, USA, 19–23 July 1997; Morgan Kaufmann: San Diago, GA, USA, 1997; pp. 465–472.
21. Lai, K.K.; Shih, K. A study of container berth allocation. *J. Adv. Transp.* **1992**, *26*, 45–60. [CrossRef]
22. Brown, G.G.; Lawphongpanich, S.; Thurman, K.P. Optimizing ship berthing. *Nav. Res. Logist.* **1994**, *41*, 1–15. [CrossRef]
23. Brown, G.G.; Cormican, K.J.; Lawphongpanich, S.; Widdis, D.B. Optimizing submarine berthing with a persistence incentive. *Nav. Res. Logist.* **1997**, *44*, 301–318. [CrossRef]
24. Imai, A.; Nagaiwa, K.I.; Tat, C.W. Efficient planning of berth allocation for container terminals in Asia. *J. Adv. Transp.* **1997**, *31*, 75–94. [CrossRef]
25. Imai, A.; Nishimura, E.; Papadimitrious, S. The dynamic berth allocation problem for a container port. *Transp. Res. Part B* **2001**, *35*, 401–417. [CrossRef]
26. Imai, A.; Nishimura, E.; Papadimitriou, S. Berth allocation with service priority. *Transp. Res. Part B* **2003**, *37*, 437–457. [CrossRef]
27. Nishimura, E.; Imai, A.; Papadimitriou, S. Berth allocation planning in the public berth system by genetic algorithms. *Eur. J. Oper. Res.* **2001**, *131*, 282–292. [CrossRef]
28. Li, C.-L.; Cai, X.; Lee, C.-Y. Scheduling with multiple-job-on-one-processor pattern. *IIE Trans.* **1998**, *30*, 433–445. [CrossRef]
29. Guan, Y.; Xiao, W.-Q.; Cheung, R.K.; Li, C.-L. A multiprocessor task scheduling model for berth allocation: Heuristic and worst-case analysis. *Oper. Res. Lett.* **2002**, *30*, 343–350. [CrossRef]
30. Park, K.T.; Kim, K.H. Berth scheduling for container terminals by using a sub-gradient optimization technique. *J. Oper. Res. Soc.* **2002**, *53*, 1054–1062. [CrossRef]
31. Park, Y.-M.; Kim, K.H. A scheduling method for berth and quay cranes. *OR Spectr.* **2003**, *25*, 1–23. [CrossRef]
32. Wang, F.; Lim, A. A stochastic beam search for the berth allocation problem. *Decis. Support Syst.* **2007**, *42*, 2186–2196. [CrossRef]
33. Lee, Y.; Chen, C.Y. An optimization heuristic for the berth scheduling problem. *EJOR* **2009**, *196*, 500–508. [CrossRef]
34. Zhen, L.; Lee, L.H.; Chew, E.P. A decision model for berth allocation under uncertainty. *EJOR* **2011**, *212*, 54–68. [CrossRef]
35. Eusuff, M.M.; Lansey, K.E.; Pasha, F. Shuffled frog-leaping algorithm: A memetic metaheuristic for discrete optimization. *Eng. Optim.* **2006**, *38*, 129–154. [CrossRef]
36. Elbeltagi, E.; Hegazy, T.; Grierson, D. A Modified Shuffled Frog-Leaping Optimization Algorithm: Applications to Project Management. *Struct. Infrastruct. Eng.* **2007**, *3*, 53–60. [CrossRef]
37. Bhattacharjee, K.K.; Sarmah, S.P. Shuffled Frog Leaping Algorithm and Its Application to 0/1 Knapsack Problem. *Appl. Soft Comput.* **2014**, *19*, 252–263. [CrossRef]
38. Zhu, Y.; Zhang, W.M. An Improved Shuffled Frog-leaping Algorithm to Optimize Component Pick-and-Place Sequencing Optimization Problem. *Expert Syst. Appl.* **2014**, *41*, 6818–6829. [CrossRef]
39. Luo, J.; Li, X.; Chen, M.R.; Liu, H. A Novel Hybrid Shuffled Frog Leaping Algorithm for Vehicle Routing Problem with Time Windows. *Inf. Sci.* **2015**, *316*, 266–292. [CrossRef]
40. Edla, R.; Lipare, A.; Cheruku, R.; Kuppili, V. An Efficient Load Balancing of Gateways Using Improved Shuffled Frog Leaping Algorithm and Novel Fitness Function for WSNs. *IEEE Sens. J.* **2017**, *12*, 6724–6733. [CrossRef]
41. Zhang, T.; Zhao, X.; Pan, X.; Li, X.; Lei, Z. Optimal Local Dimming Based on an Improved Shuffled Frog Leaping Algorithm. *IEEE Access* **2018**, *6*, 40472–40484. [CrossRef]
42. Wang, X.; Liu, S.; Li, Q.; Liu, Z. Underwater Sonar Image Detection: A Novel Quantum-Inspired Shuffled Frog Leaping Algorithm. *Chin. J. Electron.* **2018**, *27*, 588–594. [CrossRef]
43. Lei, D.; Cao, S. A novel shuffled frog-leaping algorithm for flexible job shop scheduling with interval processing time. In Proceedings of the IEEE Conferences 2017 36th Chinese Control Conference (CCC), Dalian, China, 26–28 July 2017; pp. 2708–2713.

44. Shan, W.; Nie, S.-P. Shuffled frog-leaping algorithm based neural network and its using in big data set. In Proceedings of the IEEE Conferences 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Guilin, China, 29–31 July 2017; pp. 707–711.
45. Hu, B.; Dai, Y.; Su, Y.; Moore, P.; Zhang, X.; Mao, C.; Chen, J.; Xu, L. Feature Selection for Optimized High-dimensional Biomedical Data using the Improved Shuffled Frog Leaping Algorithm. *IEEE ACM Trans. Comput. Biol. Bioinform.* **2018**, *15*, 1765–1773.