Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 1260196, 13 pages
https://doi.org/10.1155/2020/1260196

Research Article

An Improved Shuffled Frog Leaping Algorithm for Multiload AGV Dispatching in Automated Container Terminals

Xiaoyang Ma, Yongming Bian, and Fei Gao

College of Mechanical Engineering, Tongji University, Shanghai 201804, China

Correspondence should be addressed to Xiaoyang Ma; mr_shawn@foxmail.com

Received 26 March 2019; Revised 20 November 2019; Accepted 26 December 2019; Published 29 January 2020

Academic Editor: Zhaoxia Guo

Multiload AGVs, which can carry more than one container at a time, are widely used in automated container terminals. The dispatching decisions for multiload AGVs serving in automated container terminals on the target of minimum travel distance are significant in the process of container transportation in terms of improving operating efficiency. Previous work usually focused on AGVs working in a single-carrier mode, which was not only inconsistent with actual circumstances but also a waste of resources. In this paper, we establish a new mathematical model to describe multiload AGVs operating in automated container terminals, which is closer to the actual situation in real terminals. Based on this improved model, we propose a priority rule-based algorithm, termed as shuffled frog leaping algorithm with a mutant process (SFLAMUT), which can increase the diversity of the population and improve the convergence rate. Experiments were carried out based on data generated randomly according to the working properties of container terminals, and it is observed that the proposed SFLAMUT presents an effective and efficient exploration process and yields promising results in solving the proposed mathematical model.

1. Introduction

Automatic guided vehicles (AGVs), driverless and battery-powered vehicles, are applied extensively for container handling operations in container terminals. The use of multiload AGVs, which can carry up to two 20 ft containers or one 40 ft container in container terminals, is not fully investigated. In this paper, we try to investigate multiload AGV dispatching at highly automated container terminals, which can be divided into three main areas, namely, the berthing area, transport area, and yard area. Figure 1 illustrates the layout of automated container terminals. In the berthing area, containers are loaded or unloaded from a vessel by several quay cranes. The yard area is divided into several blocks, where containers are loaded and unloaded by stacking cranes. The sequence of the loading and unloading process is predetermined before a vessel arrives. The function of AGVs is to transport containers between the yard and the berth within the transport area according to the predetermined sequence.

In most container terminals, AGVs are used to carry a 20 ft or 40 ft container in a single-load carrier mode, which means an AGV cannot carry another container until it has completed the previous delivery task. In this way, there is no need to consider the capacity of AGVs and the size of containers, and thus the complexity of AGV dispatching is reduced. However, when delivering a 20 ft container, the half capacity of AGVs is not fully utilized, leading to a waste of resources and imposing a great burden on the traffic, especially when there are a large number of AGVs serving in the terminals. Given the high automation level of quay cranes and stacking cranes these days, the unloading and loading process of quay cranes and stacking cranes is of high efficiency. AGV dispatching in the transport area is becoming the bottleneck in automated container terminals.

Specifically, the AGV dispatching problem is composed of three parts, respectively, assigning tasks, routing AGVs, and traffic control [1]. Since algorithms for routing and traffic control are already integrated into the electronic control systems provided by AGV manufacturers, we only

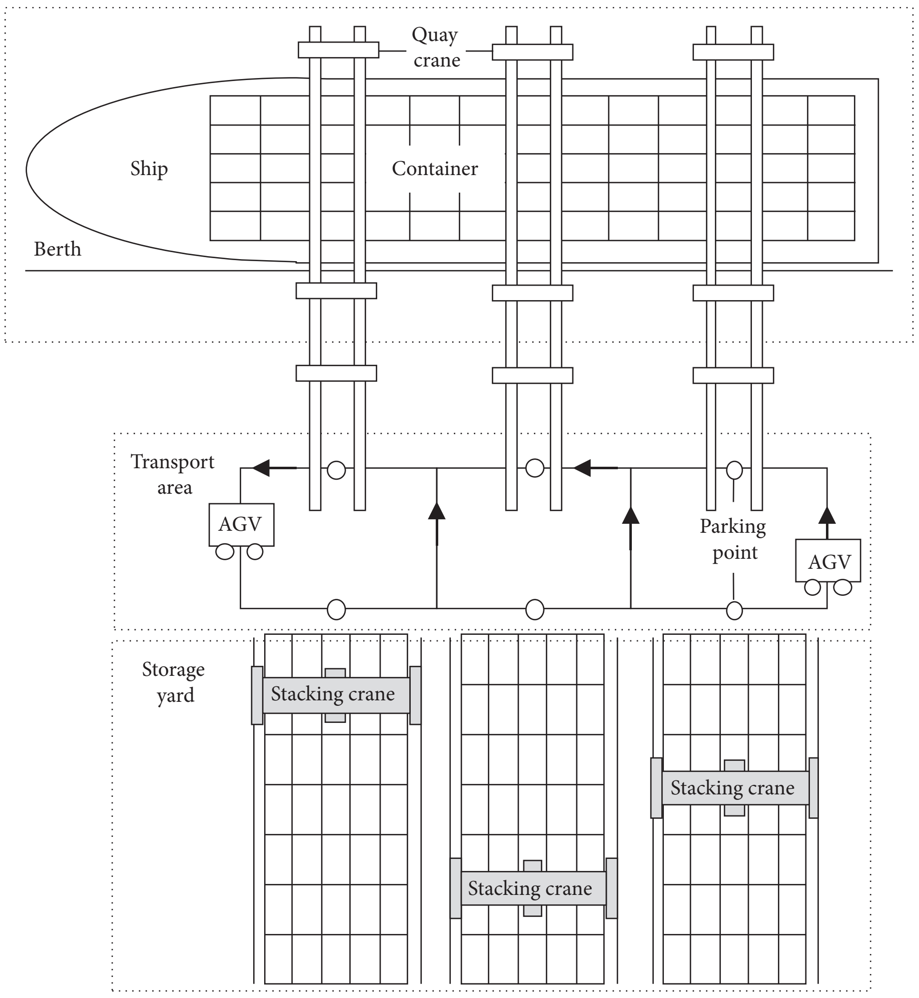

Figure 1: The layout of automated container terminals.

focus on algorithms for assigning tasks in automated container terminals. In the case of single-load AGVs, which can only carry one container at a time, the task assignment problem can be seen as the classical $m:n$ assignment problem without considering the sequence of the pickup and delivery operations. While referring to multiload AGVs, the task assignment problem is more complicated, as we may not only take the capacity of AGVs and the size of containers (20 ft or 40 ft) into consideration but also develop a pickup-delivery rule to determine the sequence of carrying out pickup tasks and delivery tasks, which dramatically increases the space complexity and time complexity of this problem.

In order to improve the operation efficiency further, we establish a new mathematical model to describe the multiload AGV dispatching problem in automated container terminals. Based on the new model, we propose an improved shuffled frog leaping algorithm, which can increase the diversity of the population and exploit the potential to find the globally optimal solution to the dispatching problem.

The rest of this article is presented as follows: in Section 2, we provide a comprehensive view of existing works. In Section 3, we describe the multiload AGV dispatching problem we try to address in detail and establish a mathematical model based on the dispatching process we developed. The improved shuffled frog leaping algorithm used to solve the model is presented in Section 4. Finally, the computational experimental results of different algorithms are compared in Section 5, and conclusions are made in Section 6.

2. Related Works

2.1. Research on AGV Dispatching Problems. AGV dispatching has been studied extensively in the past decades. However, most of the studies focused on single-load AGV dispatching, assuming that there is only one type of container in the terminal, which is not the case in real practice. Bish et al. [2] divided the AGV problem into three subproblems, including assignment, scheduling, and routing. Based on the subproblems Bish et al. raised, the following researchers had made a further study. Zaghdoud et al. [3] proposed a hybrid approach that combined the Dijkstra algorithm, genetic algorithm, and a heuristic method to solve the assignment of containers to AGVs problem. Angeloudis and Bell [4] proposed a flexible dispatching algorithm along with a new AGV dispatching approach for job assignments problems in container terminal settings under various conditions of uncertainty. Rashidi and Tsang [5] addressed a scheduling problem for automated guided vehicles in container terminals by designing an extended network simplex algorithm (NSA+); however, it was proved to be time-consuming when solving problems with a large number of tasks. Miyamoto and Inoue [6] formulated the dispatch and conflict-free routing problem of capacitated AGV systems (DCFRPC) as an integer program and proposed local/random search methods. A two-stage ant colony algorithm (ACA) was proposed by Saidi-Mehrabad et al. [7] to address the combinatorial problem of job shop scheduling and conflict-free routing.

2.2. Research on Multiload AGV Dispatching Problems. Recently, as the latest trend in transfer equipment, the multiload AGVs show enormous potential in improving the operation efficiency of automated container terminals. Grunow et al. [1] developed a flexible priority rule-based approach for both single and dual-load vehicles and analyzed the MILP model in different scenarios concerning the total lateness of the AGVs. By combining different pickup-dispatching rules, Azimi et al. [8] generated several control strategies and determined the best control strategy in terms of different criteria such as system throughput (ST), mean flow time of parts (MFTP), mean tardiness of parts (MTP), and AGV idle time (AGVIT). Fazlollahtabar et al. [9] proposed a mathematical program and an optimization method in two stages, namely, searching the solution space and finding optimal solutions, by minimizing the penalized earliness and tardiness. Chawla et al. [10] presented the modified memetic particle swarm optimization algorithm (MMPSO), which is a combination of particle swarm optimization (PSO) for global search and memetic algorithm (MA) for local search, for scheduling of multiload AGVs in the flexible manufacturing systems. Ho and Liu [11] addressed the problem of load selection and pickup-dispatching of multiple-load AGVs by proposing various load-selection rules and pickup-dispatching rules, aiming at understanding not only the performance of load-selection rules and pickup-dispatching rules in two performance criteria (throughput and tardiness) but also the effects that pickup-dispatching rules and load-selection rules have on each other's performance. Ho and Liu [12] also focused on the pickup-dispatching problem, for which nine pickup rules are proposed and studied in different performance measures, e.g., the system's throughput, the mean flow time of parts, and the mean tardiness of parts. Huo and Hu [13] established a mixed-integer linear programming model for multiload AGV dispatching and solved it by applying the genetic algorithm. Huo et al. [14] proposed GUROBI and the genetic algorithm to solve the single-load and multiload AGV dispatching problem and tested and verified the superiority of multiload AGV.

2.3. Intelligent Optimization Algorithms. In order to address the multiload AGV dispatching problem efficiently and effectively, developing an appropriate algorithm is necessary. Previous work usually concentrated on the genetic algorithm, which can fall into a local optimum easily, especially when solving complicated problems. Some research shows that intelligent optimization algorithms can address many complicated computing problems efficiently. Among them, shuffled frog leaping algorithm (SFLA) is one of the most commonly used algorithms, which can improve the searching ability theoretically. However, the iterative times will increase as the problem becomes more complicated when SFLA converges to the global optimum solution [15]. Later, Niknam et al. [16] integrated a mutant process into SFLA to increase the diversity of the frog population, which can improve the searching ability of the global optimum solution. However, improving the convergence rate of the problem with large scale and high complexity is still needed. Quantum computation has many advantages, such as diversity and parallelism. Therefore, the intelligent optimization algorithm and quantum computation are combined to generate the quantum-inspired algorithm which attracts extensive attention in the field. The quantum-inspired algorithm is mainly based on the advantages of quantum diversity, which has a relatively large searching space in a small-scale range. The authors of References [17, 18] proposed a quantum-inspired shuffled frog leaping algorithm (QSFLA) to improve the diversity further. However, it increases the randomness of the solution which leads to the slowness of convergence speed. To solve the randomness of the solution, Sun et al. [19] once proposed quantum-behaved particle swarm optimization (QPSO) based on delta potential well, the solution space of which is transformed from quantum space into real space. Subsequently, Li et al. [20] further proposed QPSO for continuous space optimization. Quantum bit's phase was used to describe the solution updated by the quantum rotation gate, which is used as a mutant process to avoid falling into the local optimum solution and improve the convergence speed.

From the study of previous works, it is learned that multiload AGV dispatching in automated container terminals is not fully investigated. Considering the research gap, we establish an improved mathematical model to describe the multiload AGV dispatching problem and propose a new algorithm to find a better solution to this model.

3. Problem Statement

In the case we consider, the multiload AGVs serving in container terminals can carry one 40 ft container or two 20 ft containers at one time, so we mainly focus on the dispatching problem of multiload AGVs which can carry up to two containers at a time and describe this kind of problem in details in this section.

A new assignment of tasks to available AGVs has to be determined, every moment a dispatching request is initiated. The definition of the multiload AGVs' availability is sparklingly important in this case. The concept of the availability of multiload AGVs has been introduced by Grunow et al. [1]. An AGV is considered unavailable if its capacity is fully utilized by the tasks assigned to it, otherwise available if an AGV can provide capacity partially or fully.

Based on the availability status of vehicles Grunow et al. proposed, we investigate the dispatching process and availability status of multiload AGVs further. An AGV is fully available after it unloads a 40 ft container or the second 20 ft container or two 20 ft containers simultaneously, and an AGV is partially available after loading the first 20 ft container. On the contrary, an AGV is unavailable after loading the second 20 ft container or a 40 ft container or two 20 ft containers simultaneously. As presented in Figure 2, different kinds of lines represent different kinds of status. The solid line means that the AGV is not available both during the trip to the next stop and at the next stop, while the dotted line and dashed line represent that the AGV is fully available or partially available, respectively. For an AGV with a different status, tasks at the next stop vary. A fully available AGV can carry a 40 ft container or a 20 ft container or two 20 ft containers simultaneously, and a partially available AGV is allowed to load a second 20 ft container or unload the first 20 ft container it carries. However, an unavailable AGV can only carry out the delivery task to unload the container after finishing the previous pickup task. In order to determine which type of container should be carried and whether the next task is a pickup task or a delivery task, we develop a priority rule presented in the subsequent sections.

Based on the multiload AGV dispatching process stated above, we formulate the mathematical model of the dispatching problem as follows, together with corresponding assumptions. The formulated dispatching problem is subjected to several constraints and optimized for minimum travel distance.

3.1. Basic Assumptions.

The following assumptions are considered in the present study:

(1) At the initial stage, all of the AGVs are empty and are in the waiting place
(2) The distance between two points is known in advance, and AGVs travel from one point to another point through the predetermined guide path
(3) The location and number of delivery points and pickup points are fixed during the whole dispatching process
(4) The number of AGVs is also fixed
(5) Quay cranes, yard cranes, and multiload AGVs are reliable
(6) There is no task preemption
(7) There is no traffic congestion
(8) After completing all the tasks, the AGVs will travel to the waiting place

3.2. Model Formulation.

Before introducing the newly proposed mathematical model, the following notation has to be defined.

Parameters include the number of tasks, the number of AGVs, the node-set of all task points for AGV$_k$, all available routes for AGV$_k$, the maximum capacity of AGV$_k$, and the traveling distance between different task points, which are known in advance according to the operation plan of automated container terminals. The detailed descriptions are shown below:

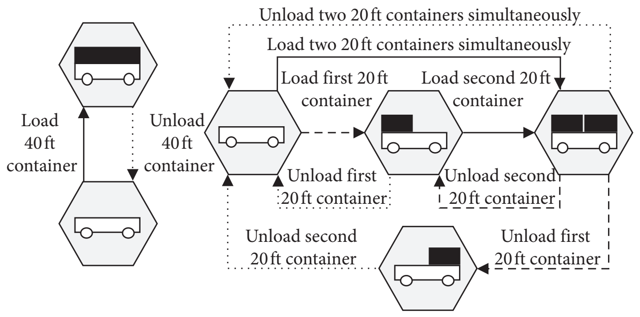
Unload two 20 ft containers simultaneously
Load two 20 ft containers simultaneously
Load first 20 ft container
Load second 20 ft container
Load 40 ft container
Unload 40 ft container
Unload first 20 ft container
Unload second 20 ft container
Unload second 20 ft container
Unload first 20 ft container

Figure 2: The dispatching process of a multiload AGV and its availability status.

3.2.1. Parameters

n: number of tasks

u: number of task points (including pickup points, delivery points, starting point, and termination point)

K: number of AGVs

P: node-set of pickup points

D: node-set of delivery points

$T = P \cup D$: set of nodes for all task points

P_k: node-set of pickup points for AGV$_k$

D_k: node-set of delivery points for AGV$_k$

$T_k = P_k \cup D_k$: node-set of pickup and delivery points for AGV$_k$

s_k: starting point of AGV$_k$

d_k: termination point of AGV$_k$

$V_k = T_k \cup \{s_k, d_k\}$: node-set of all task points for AGV$_k$

$A_k = V_k \times V_k$: all available routes for AGV$_k$

C_k: the maximum capacity of AGV$_k$

$d_{i,j,k}$: the traveling distance between points i and j

Nodes i and $n+i$: the pickup and delivery point of the ith task, respectively

n_i: the number of containers transported from point i to point $n+i$

Decision variables include the routes AGV$_k$ visits, the number of containers AGV$_k$ carries, and the type of container loaded by AGV$_k$, which are determined according to the shortest distance rule proposed in Section 4. The detailed descriptions are presented below:

3.2.2. Decision Variables

$x_{i,j,k}$: the route AGV$_k$ visits. Initially, $x_{i,j,k} = 0$. If AGV$_k$ moves from point i to point j $((i, j) \in A_k)$, $x_{i,j,k} = x_{i,j,k} + 1$.

$l_{i,k}$: the number of containers AGV$_k$ carries, after completing the task at point i. Initially, $l_{i,k} = 0$.

$s_{i,k}$: the type of container loaded at point i by AGV$_k$. If it is 40 ft, $s_{i,k} = 1$. If it is 20 ft, $s_{i,k} = 0$.

Applying the above notation, multiload AGV dispatching problem in container terminals can be formulated as an integer programming model. The objective function and constraints of the problem are presented as follows:

3.2.3. Objective Function

$$F = \text{minimize} \sum_{k \in K} \sum_{(i,j) \in A_k} d_{i,j,k} x_{i,j,k}. \tag{1}$$

3.2.4. Constraints

$$\sum_{j \in P_k \cup \{d_k\}} x_{s_k,j,k} = 1, \quad \forall k \in K, \tag{2}$$

$$\sum_{i \in D_k \cup \{s_k\}} x_{i,d_k,k} = 1, \quad \forall k \in K, \tag{3}$$

$$\sum_{k \in K} \sum_{j \in T_k \cup \{d_k\}} x_{i,j,k} = 1, \quad \forall i \in P_k, \tag{4}$$

$$\sum_{j \in T_k} x_{i,j,k} - \sum_{j \in T_k} x_{j,n+i,k} = 0, \quad \forall k \in K,\ i \in P_k, \tag{5}$$

$$\sum_{i \in T_k \cup \{s_k\}} x_{i,j,k} - \sum_{i \in T_k \cup \{d_k\}} x_{j,i,k} = 0, \quad \forall k \in K,\ j \in T_k, \tag{6}$$

$$\sum_{j=n+i} x_{i,j,k} = 1, \quad \forall k \in K,\ i \in \{P_k\colon s_{i,k} = 1\}, \tag{7}$$

$$\sum_{j \in P_k} x_{i,j,k} + \sum_{j=n+i} x_{i,j,k} = 1, \quad \forall k \in \{K\colon l_{i,k} = 1\},\\ i \in \{P_k\colon s_{i,k} = 0\}, \tag{8}$$

$$x_{i,j,k}\left(l_{i,k} - n_i - l_{j,k}\right) = 0, \quad \forall k \in K,\ (i, j) \in A_k, \tag{9}$$

$$n_i \le l_{i,k} \le C_k, \quad \forall k \in K,\ i \in P_k, \tag{10}$$

$$0 \le l_{n+i,k} \le C_k - n_i,\\ \forall k \in K,\ n+i \in D_k, \tag{11}$$

$$l_{s_k,k} = 0, \quad \forall k \in K, \tag{12}$$

$$l_{d_k,k} = 0, \quad \forall k \in K. \tag{13}$$

Equation (1) is the objective function that minimizes total travel distances. Equations (2) and (3) are the constraints on the beginning and the end of the stage, which indicates that multiload AGVs must start traveling from the starting point before performing any tasks and move to the termination point to wait for the new pickup request after completing all tasks. Equation (4) illustrates that every task can only be executed once by one AGV. Equation (5) ensures that the pickup and delivery task of a container must be carried out by one AGV. Equation (6) means that if an AGV arrives at a task point, it will exit that point. Equations (7) and (8) are constraints on the sequence of delivery tasks and pickup tasks, which indicates that an AGV must go to the associated delivery point to unload the container if it has just loaded a 40 ft container, and an AGV can visit the associated delivery point or any pickup point where 20 ft containers are waiting to be loaded after loading a 20 ft container. Equations (9)~(13) are constraints on the capacity of an AGV, which illustrate that the number of containers carried by the AGV will never exceed its capacity and the number of containers transported by the AGV from one point to another point will never exceed its load.

4. Shuffled Frog Leaping Algorithm with a Mutant Process (SFLAMUT)

Multiload AGVs are assigned several tasks in the very beginning, and then each AGV carries out the tasks assigned according to a predetermined AGV control process, which works as a rule giving multiload AGV guidance about task determination (whether a delivery task or a pickup task is performed). To solve the model established above, we propose a priority rule-based algorithm, called shuffled frog leaping algorithm with a mutant process (SFLAMUT), which conforms to the shortest distance rule and an improved shuffled frog leaping scheme.

The shortest distance rule, under which the multiload AGV will give priority to the containers in the nearest pickup point or delivery point, has been proved to have the best throughput performance in AGV dispatching problem [12]. Therefore, we adopt the shortest distance rule as our dispatching rule and design the multiload AGV control process based on this rule. When the operation begins, every AGV performs the pickup task or delivery task by following the control process illustrated in Figure 3.

SFLAMUT, based on the shuffled frog leaping scheme, is designed to address the integer programming problem as Section 3.2 stated, which has fast convergence speed and powerful searching ability to find the global optimum solution.

4.1. Population Initialization. Assuming that the number of tasks is l, the number of AGVs is k, the frog population size is F, and the number of subgroups is m. Set the state of the ith frog as $N_i = [n_{i1}, n_{i2}, \ldots, n_{il}]$, and the fitness value of its position as fitness$_i = f(N_i)$. The state of every frog in the frog population is initialized randomly. All frogs are sorted

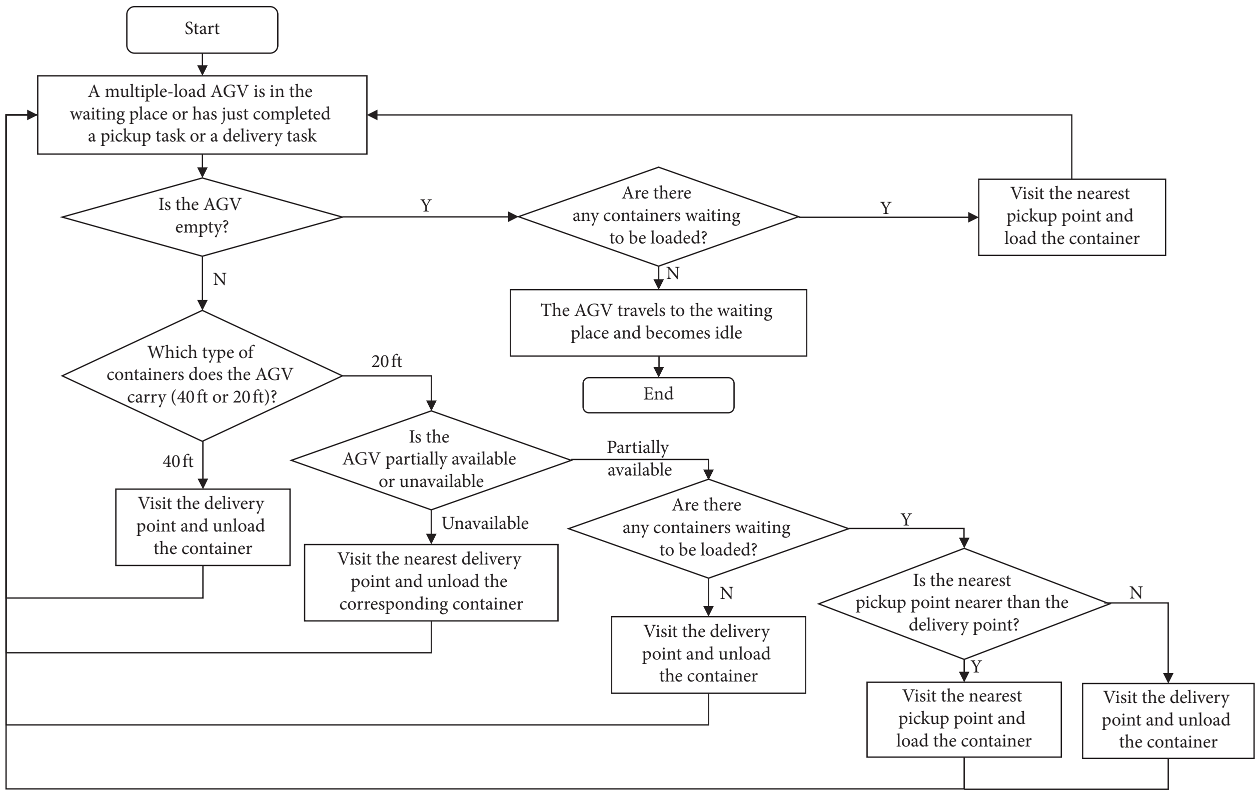
Start
A multiple-load AGV is in the waiting place or has just completed a pickup task or a delivery task
Is the AGV empty?
Y
Are there any containers waiting to be loaded?
Y
Visit the nearest pickup point and load the container
N
The AGV travels to the waiting place and becomes idle
End
N
Which type of containers does the AGV carry (40 ft or 20 ft)?
20 ft
Is the AGV partially available or unavailable
Partially available
40 ft
Visit the delivery point and unload the container
Unavailable
Visit the nearest delivery point and unload the corresponding container
Are there any containers waiting to be loaded?
Y
N
Is the nearest pickup point nearer than the delivery point?
N
Visit the delivery point and unload the container
Y
Visit the nearest pickup point and load the container
Visit the delivery point and unload the container

Figure 3: The multiload AGV control process.

in the ascending order according to the fitness value, before being divided into m subgroups containing s frogs, where $m \times s = F$. The grouping process is stated as follows: the first frog is assigned to the first subgroup, the second frog is assigned to the second subgroup, …, the mth frog is assigned to the $m$thsubgroup, the $(m+1)$th frog is assigned to the first subpopulation, …, and so on, until every frog individual in the frog population is assigned to a subgroup. The whole frog population can be expressed as follows:

$$P = \begin{bmatrix} n_{11} & n_{12} & \cdots & n_{1j} & \cdots & n_{1l} \\ n_{21} & n_{22} & \cdots & n_{2j} & \cdots & n_{2l} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ n_{i1} & n_{i2} & \cdots & n_{ij} & \cdots & n_{il} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ n_{F1} & n_{F2} & \cdots & n_{Fj} & \cdots & n_{Fl} \end{bmatrix}, \quad (14)$$

where $n_{ij} \in [1, k]$ is the serial number of an AGV, representing a specific AGV.

4.2. Fitness Function Calculation Based on the Shortest Distance Rule. The fitness function is mainly used to evaluate the frog position, which reflects the extent that the frog is near the best solution. The fitness value of a frog is derived from the objective function shown in equation (1), which is calculated by following the multiload AGV control process based on the short distance rule (illustrated in Figure 3). The relationship between the fitness function and objective function is shown below. F_{GA} represents the best solution the genetic algorithm generates:

$$\text{fitness} = \frac{F}{F_{\mathrm{GA}}}. \quad (15)$$

The pseudocode of the shortest distance rule and the calculation process of the objective function is presented in Pseudocode 1. Following parameters of this problem is predetermined: node-set of pickup points $P = [p_1, p_2, \ldots, p_j, \ldots, p_l]$; node-set of delivery points $D = [d_1, d_2, \ldots, d_j, \ldots, d_l]$; container size $S = [s_1, s_2, \ldots, s_j, \ldots, s_l]$; node-set of tasks $T = [t_1, t_2, \ldots, t_j, \ldots, t_l]$, where $t_j = [p_j, d_j, s_j]$, which means the jth task is to transport the container of size s_j from pickup point p_j to delivery point d_j; the traveling distance between different task points distance = $\begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1j} & \cdots & d_{1u} \\ d_{21} & d_{22} & \cdots & d_{2j} & \cdots & d_{2u} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ d_{i1} & d_{i2} & \cdots & d_{ij} & \cdots & d_{iu} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ d_{u1} & d_{u2} & \cdots & d_{uj} & \cdots & d_{uu} \end{bmatrix}$, where u represents the number of different task points and d_{ij} represents the distance between the ith task point and the jth task point.

4.3. Updating with a Mutant Process. Assuming that $\mathrm{ch}_u\,(1 \le u \le m)$ is each subgroup, $\mathrm{ch}_u^{lw} = [n_{u1}^{lw}, n_{u2}^{lw}, \ldots, n_{uj}^{lw}, \ldots, n_{ul}^{lw}]\ (1 \le j \le l)$ represents the worst frog in the subgroup. $\mathrm{ch}_u^{lb} = [n_{u1}^{lb}, n_{u2}^{lb}, \ldots, n_{uj}^{lb}, \ldots, n_{ul}^{lb}]$ represents the

```
Initialize:/* initialize tasks and AGVs
  T = InitializeTasks();/* initialize the pickup point, delivery point, and size of container of tasks
  AGVs = InitializeAGVs();/* initialize the status of AGV and assign tasks to AGVs randomly
GetRoutes:/* AGV route planning according to tasks assigned
  While there are tasks waiting or AGV is not in the waiting place:/* for every AGV in AGVs
    IF AGV is in the waiting place or has just completed a pickup task or a delivery task:
      IF AGV is empty:
        IF the number of containers waiting to be loaded != 0:
          AGV visits the nearest pickup point;
          $route_{p,d,k} = route_{p,d,k} + 1$;/* update the route of AGVs; $AGV_k$ travels from current point $p$ to pickup point $d$
        ELSE:
          AGV travels to the waiting place;
          $route_{p,d,k} = route_{p,d,k} + 1$;/* $AGV_k$ travels from current point $p$ to waiting point $d$
      ELSE:
        IF the size of the container the AGV is carrying == 40 ft:
            AGV visits the delivery point;
            $route_{p,d,k} = route_{p,d,k} + 1$;/* $AGV_k$ travels from current point $p$ to delivery point $d$
        ELSE:
            IF AGV is unavailable:
              AGV visits the nearest delivery point;
              $route_{p,d,k} = route_{p,d,k} + 1$; /* $AGV_k$ travels from current point $p$ to delivery point $d$
            ELSE:
              IF the number of containers waiting to be loaded != 0:
                IF the distance travel to the nearest pickup point < the distance travel to the delivery point:
                  AGV visits the nearest pickup point;
                  $route_{p,d,k} = route_{p,d,k} + 1$; /* $AGV_k$ travels from current point $p$ to pickup point $d$
              ELSE:
                  AGV visits the delivery point
                  $route_{p,d,k} = route_{p,d,k} + 1$/* $AGV_k$ travels from current point $p$ to the delivery point
              ELSE:
                  AGV visits the delivery point
                  $route_{p,d,k} = route_{p,d,k} + 1$/* $AGV_k$ travels from current point $p$ to the delivery point $d$
  RETURN route
Calculate the objective function:
  $F = \sum\sum route_{p,d,k} distance_{p,d,k}$/* $distance_{p,d,k}$ represents the distance between task point $p$ and task point $d$
```

Pseudocode 1: The shortest distance rule and the calculation process of the objective function.

best frog in the subgroup, and $G = [n_1^{gb}, n_2^{gb}, \ldots, n_j^{gb}, \ldots n_l^{gb}]$ is the global best frog in the frog population. The specific update strategy is expressed as follows:

$$d = \begin{cases} \text{randint}(0, \Delta(n)), & \text{if } \text{ch}_u^{lb}(t) > \text{ch}_u^{lw}(t), \\ \text{randint}(\Delta(n), 0), & \text{if } \text{ch}_u^{lb}(t) < \text{ch}_u^{lw}(t), \end{cases} \tag{16}$$

$$\text{ch}_u^{lw}(t+1) = \text{ch}_u^{lw}(t) + d, \tag{17}$$

where randint is a random integer ranging from 0 to $\Delta(n)$, or $\Delta(n)$ to 0, where $\Delta(n) = \text{ch}_u^{lb}(t) - \text{ch}_u^{lw}(t)$, and t is the iterative time.

In each subgroup, ch_u^{lw} is updated by equation (17). During the updating process, a temporary variable new $\text{ch}_u^{lw} = [n_{u1}^{\text{new}}, n_{u2}^{\text{new}}, \ldots, n_{uj}^{\text{new}}, \ldots, n_{ul}^{\text{new}}]$ is used to store the updated solution. If new ch_u^{lw} is better than ch_u^{lw}, which means that the fitness value of new ch_u^{lw} is lower than the fitness value of ch_u^{lw}, new ch_u^{lw} will replace ch_u^{lw}. Otherwise, G will take the place of ch_u^{lb} in equation (16), and new ch_u^{lw} is recalculated. After completing the above process, if ch_u^{lw} is still not updated, a randomly generated new ch_u^{lw} will replace ch_u^{lw}. The previous process is repeated before reaching the local maximum iterative times. After the local search, all frogs are mixed and regrouped into m subgroups, and the above process of local search is repeated until reaching the boundary limits.

The above update strategy may lead to convergence to the local optimal solution. To avoid falling into the local optimum and improve the diversity of the frog population, introducing a mutant process to this algorithm is necessary. The mutant process is developed according to the fitness value after all frogs being regrouped into subgroups, which can guarantee the convergence speed and avoid falling into the local optimum.

The process is expressed as follows:

$$r_j = \text{randint}(1, k), \quad j = [1, 2, \ldots, k], \tag{18}$$

$$\text{rand ch}_u^l = [r_1, r_2, \ldots, r_j, \ldots, r_l], \tag{19}$$

$$d = \begin{cases} \text{randint}\left(0, \text{ch}_u^{lb}(t) - \text{rand ch}_u^{l}(t)\right), & \text{if } \text{ch}_u^{lb}(t) > \text{ch}_u^{lw}(t), \\ \text{randint}\left(\text{ch}_u^{lb}(t) - \text{rand ch}_u^{l}(t), 0\right), & \text{if } \text{ch}_u^{lb}(t) < \text{ch}_u^{lw}(t), \end{cases} \tag{20}$$

$$\text{mut ch}_u^l = \text{rand ch}_u^l + d. \tag{21}$$

If the fitness values of rand ch_u^l is better than ch_u^{lb}, ch_u^{lb} will be substitute by rand ch_u^l; Otherwise, if mut ch_u^l is better than ch_u^{lb}, ch_u^{lb} will be substitute by mut ch_u^l; otherwise, ch_u^{lb} will remain the same, and ch_u^{lw} will be substitute by rand ch_u^l.

4.4. Algorithm Process. In summary, the algorithm process of SFLAMUT is stated as follows:

(1) Initialize population P, and set iteration times $t = 0$
(2) Calculate the fitness value of every frog by equation (15), keep the best frog
(3) Sort the frogs in population according to their fitness value in the ascending order and group the frogs into several subgroups
(4) Update the worst frog in every subgroup by equations (16) and (17)
(5) Divide the updated population into new subgroups
(6) Apply the mutant process by equations (18)~(21)
(7) Judge whether the termination criteria is met; if so, the current best frog is the optimum result yielded; otherwise, set $t = t + 1$ and turn to Step (2)

4.5. Convergence Analysis

4.5.1. Local Convergence Analysis. To facilitate the convergence analysis, we assume that the number of subgroup $m = 1$, indicating that the global best frog G equals the best frog ch_u^{lb} in the subgroup at the beginning. Describe equations (16) and (17) as follows.

The tth iteration:

$$\text{ch}_u^{lw}(t) = \text{Round}\left[r\left(\text{ch}_u^{lb} - \text{ch}_u^{lw}(t-1)\right)\right] + \text{ch}_u^{lw}(t-1), \quad 0 < r < 1, \tag{22}$$

The $(t + 1)$th iteration:

$$\text{ch}_u^{lw}(t+1) = \text{Round}\left[r\left(\text{ch}_u^{lb} - \text{ch}_u^{lw}(t)\right)\right] + \text{ch}_u^{lw}(t), \quad 0 < r < 1, \tag{23}$$

where $\text{Round}(x)$ is the integer number of x through rounding.

Let e_i be equal to the difference between the integer number of $r(\text{ch}_u^{lb} - \text{ch}_u^{lw}(i-1))$ through rounding and $r(\text{ch}_u^{lb} - \text{ch}_u^{lw}(i-1))$, equations (24) and (25) can be derived as follows:

$$\begin{aligned} e_t = {} & \text{Round}\left[r\left(\text{ch}_u^{lb} - \text{ch}_u^{lw}(t-1)\right)\right] \\ & - r\left(\text{ch}_u^{lb} - \text{ch}_u^{lw}(t-1)\right), \end{aligned} \tag{24}$$

$$\begin{aligned} e_{t+1} = {} & \text{Round}\left[r\left(\text{ch}_u^{lb} - \text{ch}_u^{lw}(t)\right)\right] \\ & - r\left(\text{ch}_u^{lb} - \text{ch}_u^{lw}(t)\right). \end{aligned} \tag{25}$$

Substitute equation (24) into equation (22) and equation (25) into equation (23), and let $\Delta\text{ch}_u^l(t+1) = \text{ch}_u^l(t+1) - \text{ch}_u^l(t)$, then equation (26) can be derived from (equation (23) − equation (22)):

$$\Delta\text{ch}_u^{lw}(t+1) = (1-r)\Delta\text{ch}_u^{lw}(t) + e_{t+1} - e_t. \tag{26}$$

Substitute equations (24) and (25) into equation (26):

$$\Delta\text{ch}_u^{lw}(t+1) - \Delta\text{ch}_u^{lw}(t) = \Delta\text{Round}, \tag{27}$$

where $\Delta\text{Round} = \text{Round}[r(\text{ch}_u^{lb} - \text{ch}_u^{lw}(t) + \Delta\text{ch}_u^{lw}(t))] - \text{Round}[r(\text{ch}_u^{lb} - \text{ch}_u^{lw}(t))]$.

If $\text{ch}_u^{lb} - \text{ch}_u^{lw}(t) \geq 0$, we can yield that $\Delta\text{ch}_u^{lw}(t+1) \geq 0$ and $\Delta\text{ch}_u^{lw}(t) \geq 0$. In this case, $-\Delta\text{ch}_u^{lw}(t) \leq \Delta\text{Round} \leq 0$. If $\Delta\text{ch}_u^{lw}(t) \neq 0$, equation (28) can be derived:

$$\frac{\Delta\text{ch}_u^{lw}(t+1)}{\Delta\text{ch}_u^{lw}(t)} = 1 + \frac{\Delta\text{Round}}{\Delta\text{ch}_u^{lw}(t)}, \quad -\Delta\text{ch}_u^{lw}(t) \leq \Delta\text{Round} < 0. \tag{28}$$

Let $q(t)$ be equal to $1 + (\Delta\text{Round}/\Delta\text{ch}_u^{lw}(t))$. From equation (28), we can identify that $0 < q(t) < 1$. In this case, $\lim_{t\longrightarrow\infty}\Delta\text{ch}_u^{lw}(t) = 0$ and $\lim_{t\longrightarrow\infty}\text{ch}_u^{lw}(t) = \text{ch}_u^{lb}$, indicating that the result converges to the local optimal solution.

If $\text{ch}_u^{lb} - \text{ch}_u^{lw}(t) \leq 0$, we can yield the same result according to the above derivation similarly.

4.5.2. Global Convergence Analysis. As SFLAMUT is a kind of random search techniques, we determine whether SFLAMUT can get the global optimization solution through the convergence criteria of the stochastic algorithm proposed by Solis and Wets [21] as follows.

For optimization problem $\langle A, f\rangle$, we seek a point x in A which minimizes f on A. D is a random search algorithm, X_k is the solution of the kth iteration, and the solution of the next iteration is $X_{k+1} = D(X_k, \alpha)$, where A is the feasible solution space, f is the fitness function, α is a solution algorithm D once found.

Condition H1: $f(D(x,\alpha)) \leq f(x)$, and if $\alpha \in A$, $f(D(x,\alpha)) \leq f(\alpha)$.

Condition H2: for any subset B of A with $v(B) > 0$, we have that $\prod_{k=0}^{\infty}(1 - \mu_k(B)) = 0$, where $v(B)$ is the Lebesgue measure on the set A, and $\mu_k(B)$ is the probability measure of solution at iteration k on the set B.

Convergence Theory (Global Search) [21]: suppose that f is a measurable function, A is a measurable subset of R^n

and Conditions H1 and H2 are satisfied. Let $\{x_k\}_{k=0}^{\infty}$ be a sequence generated by algorithm D. Then,

$$\lim_{k\longrightarrow\infty} P(x_k \in R_\varepsilon) = 1, \tag{29}$$

where R_ε is the set of global best solutions and $P(x_k \in R_\varepsilon)$ is the probability that the point x_k generated by the algorithm is in R_ε at step k.

To verify whether SFLAMUT can get the global optimization solution or not, we just need to verify whether SFLAMUT satisfies Conditions H1 and H2. For SFLAMUT, algorithm D can be defined as follows:

$$D(X_b(k), X_i(k)) = \begin{cases} X_b(k), & f(X_b(k)) \le f(X_i(k)), \\ X_i(k), & f(X_b(k)) > f(X_i(k)). \end{cases} \tag{30}$$

It is obvious that algorithm D satisfies Condition H1. If algorithm satisfies Condition H2, the union of all solutions algorithm D once found can consist of the feasible solution space, that is,

$$\bigcup_{i=1}^{F} R_{i,k} \supseteq A, \tag{31}$$

where $R_{i,k}$ is the solution algorithm D found at iteration times k and A is the feasible solution space. From equation (22), we have that

$$R_{i,t} = \text{Round}\left[r\left(X_b - X_i(t-1)\right)\right] + X_i(t-1), \quad 0 < r < 1. \tag{32}$$

From local convergence analysis, we have proved that $\lim_{t\longrightarrow\infty}\text{ch}_u^{lw}(t) = \text{ch}_u^{lb}$, so when $t \longrightarrow \infty$, $R_{i,t} \longrightarrow X_b$. In this case, $v(\cup_{i=1}^{F} R_{i,t} \cap A) < v(A)$, which means that the algorithm cannot ensure the global convergence if just updated through equations (16) and (17). Therefore, we introduce the mutant process (equations (18)~(21)) to the updating process. $X_i(t)$ will be generated randomly through the mutant process after every iteration. In this case, if $t \longrightarrow \infty$, we have that $\cup_{i=1}^{F} R_{i,t} = A$, indicating that equation (31) is satisfied. In this case, for any subset B of A, we have that $0 < \mu_k(B) = \sum_{i=0}^{F}\mu_{i,k}(B) \le 1$, $\prod_{k=0}^{\infty}(1-\mu_k(B)) \longrightarrow \lim_{k\longrightarrow\infty}(1-\sum_{i=0}^{F}\mu_{i,k}(B))^k = 0$, which means the probability that algorithm D cannot find the solution in B is zero, and therefore, Condition H2 is satisfied.

In conclusion, the updating process of equations (16) and (17) can ensure that the algorithm converges to the current optimal position; however, it cannot guarantee the global convergence. When equations (16) and (17) cannot generate a better solution, equations (18)~(21) randomly generate a solution to substitute a current solution, which can expand the sample space. In this case, as iteration times $k \longrightarrow \infty$, the probability that SFLAMUT converges to the global optimum solution is 1, which indicates the global convergence of SFLAMUT.

5. Experimental Results

5.1. Description of the Experiments. The proposed mathematical model and SFLAMUT were tested by computational experiments on hypothetical automated container terminals, in which multiload AGVs are employed. The computational experiment was performed by the Python program. We choose algorithms including genetic algorithm (GA), shuffled frog leaping algorithm (SFLA), and quantum-inspired shuffled frog leaping algorithm (QSFLA) as our benchmarks. The difference between the given algorithms is presented as follows:

(1) GA: traditional genetic algorithm with roulette wheel selection process, single-point crossover process, and a mutant process

(2) SFLA: traditional shuffled frog leaping algorithm with improved updating process suitable for solving integer programming problems

(3) QSFLA: shuffled frog leaping algorithm and quantum computation are combined to generate QSFLA, the solution space of which is transformed from quantum space into real space

(4) SFLAMUT: an improved shuffled frog leaping algorithm with a mutant process as stated in Section 4

5.2. Parameter Settings. The parameters in the mathematical model we proposed were set as follows: The number of pickup points was set to 5, the number of delivery points to 5, and the number of multiload AGVs to 5. The parameters of different algorithms are presented in Table 1. The sign "—" represents that the parameter of the algorithm does not exist.

We used GA, QSFLA, SFLA, and SFLAMUT to solve the mathematical model, respectively, and evaluated the performance of the algorithms by comparing the convergence speed, optimal results, and stability at a fixed hybrid iteration number of 500.

For tasks of different numbers ranging from 10 to 80, each algorithm was executed 50 times independently in order to reduce the impact of random factors. We calculated the best value, worst value, and mean value of the optimal results generated by 50 experimental runs of each algorithm. Details are shown in Table 2.

5.3. Results and Discussion. From Table 2, we can figure out that when it comes to searching for a better optimum solution, intelligent optimization algorithms (SFLA, QSFLA, and SFLAMUT) perform better than the traditional heuristic algorithm (GA) in terms of the best value, worst value, and mean value in most cases. This is because GA with selection and crossover process will retain the genes of the best individuals to the next generation, leading to falling into the local optimum especially when the population size is small. However, SFLA, QSFLA, and SFLAMUT with the updating process of global information exchange can avoid falling into the local optimum to some extent. Although SFLA, QSFLA, and SFLAMUT perform better with respect to optimal solutions, they have longer running time than GA due to a more complex searching process. This limitation can be accepted in the

Table 1: Parameter settings of GA, SFLA, QSFLA, and SFLAMUT.

Number of tasks	Algorithm	Population size	Mutation rate	Number of subgroup	Number of internal interaction
10	GA	10	0.1	—	—
	SFLA	10	—	5	2
	QSFLA	10	—	5	2
	SFLAMUT	10	—	5	2
20	GA	20	0.1	—	—
	SFLA	20	—	10	2
	QSFLA	20	—	10	2
	SFLAMUT	20	—	10	2
30	GA	30	0.1	—	—
	SFLA	30	—	10	3
	QSFLA	30	—	10	3
	SFLAMUT	30	—	10	3
40	GA	40	0.1	—	—
	SFLA	40	—	20	2
	QSFLA	40	—	20	2
	SFLAMUT	40	—	20	2
50	GA	50	0.1	—	—
	SFLA	50	—	25	2
	QSFLA	50	—	25	2
	SFLAMUT	50	—	25	2
60	GA	60	0.1	—	—
	SFLA	60	—	30	2
	QSFLA	60	—	30	2
	SFLAMUT	60	—	30	2
70	GA	70	0.1	—	—
	SFLA	70	—	35	2
	QSFLA	70	—	35	2
	SFLAMUT	70	—	35	2
80	GA	80	0.1	—	—
	SFLA	80	—	40	2
	QSFLA	80	—	40	2
	SFLAMUT	80	—	40	2

Table 2: The results of GA, SFLA, QSFLA, and SFLAMUT.

Number of task	Algorithm	Best value	Worst value	Mean value	Standard deviation	Running time (s)
10	GA	274	278	276	1.84	13.29
	SFLA	274	274	274	0.00	48.66
	QSFLA	274	274	274	0.00	65.31
	SFLAMUT	274	274	274	0.00	50.85
20	GA	340	376	358	12.80	66.62
	SFLA	328	348	336	6.85	256.19
	QSFLA	336	352	344	5.33	352.88
	SFLAMUT	328	340	332	5.03	273.24
30	GA	428	474	452	14.80	42.38
	SFLA	416	440	426	7.23	179.25
	QSFLA	428	452	439	6.88	248.90
	SFLAMUT	406	434	425	8.85	187.57
40	GA	632	666	646	10.48	93.99
	SFLA	608	636	625	9.80	352.66
	QSFLA	612	658	636	14.42	493.33
	SFLAMUT	608	628	620	5.85	381.65
50	GA	780	808	790	9.35	88.59
	SFLA	772	786	778	5.32	335.90
	QSFLA	772	794	783	7.19	471.73
	SFLAMUT	760	784	774	7.15	358.49
60	GA	926	974	948	12.78	141.50
	SFLA	914	950	932	11.37	542.50
	QSFLA	928	960	948	9.23	758.42
	SFLAMUT	910	946	927	12.62	582.16

Table 2: Continued.

Number of task	Algorithm	Best value	Worst value	Mean value	Standard deviation	Running time (s)
70	GA	1138	1166	1150	10.46	153.68
	SFLA	1108	1148	1126	9.91	592.01
	QSFLA	1134	1164	1145	11.12	827.36
	SFLAMUT	1106	1148	1126	12.54	639.29
80	GA	1192	1224	1214	9.32	205.87
	SFLA	1176	1224	1197	15.70	800.00
	QSFLA	1196	1228	1214	10.49	1090.99
	SFLAMUT	1168	1218	1194	15.78	851.53

case we consider because the task assignment process should be completed in advance, not in operation, which means there is plenty of time to execute each algorithm. Among intelligent optimization algorithms (SFLA, QSFLA, and SFLAMUT), SFLAMUT we proposed yielded the minimum best value, the minimum worst value, and the minimum mean value, indicating far better optimal results than the other three algorithms in all experiments we carried out. That means SFLAMUT has the potential to find the global optimum solution, despite a slightly longer running time. Besides, SFLAMUT has a relatively low value with respect to the standard deviation, which indicates better search accuracy and credibility.

The optimization process of these four algorithms is compared in Figures 4–11. As depicted in each figure, the vertical axis is the objective function, whose value is obtained from the mean value of 50 experimental runs of each algorithm, and the horizontal axis is the evolutionary generation (iteration times).

As Figures 4–11 show, intelligent optimization algorithms (SFLA, QSFLA, and SFLAMUT) have a faster convergence speed than the traditional heuristic algorithm (GA) in most cases. However, as the complexity of the problem increases, QSFLA and GA have almost similar performance concerning convergence speed and optimum results, which indicates QSFLA performs worse when solving complex problems. This is reasonable because QSFLA transforms the solution space from real space into quantum space, which improves the diversity but increases the randomness of the solution, leading to the slowness of convergence speed, especially when solving complicated problems. Therefore, when the iteration times are fixed, the results QSFLA yielded become worse and worse as problems become more and more complex (as shown in Figures 9–11). SFLA and SFLAMUT always perform better than QSFLA and GA because they have similar improved updating process as equations (16) and (17) show. Such an updating process maintains the advantages (avoiding falling into the local optimum and a faster convergence speed) of traditional SFLA and is suitable for solving the integer programming problem in the meantime. Compared with SFLA, SFLAMUT has the best performance among the four algorithms. We add the mutant process (equations (18)~(21)) into the updating process of SFLAMUT,

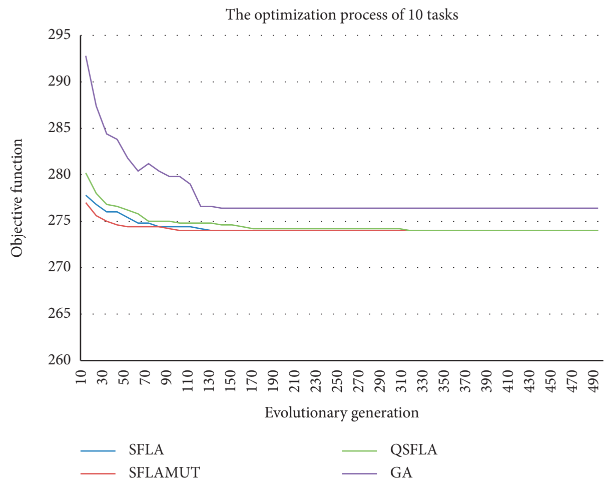

Figure 4: The optimization process of 10 tasks.

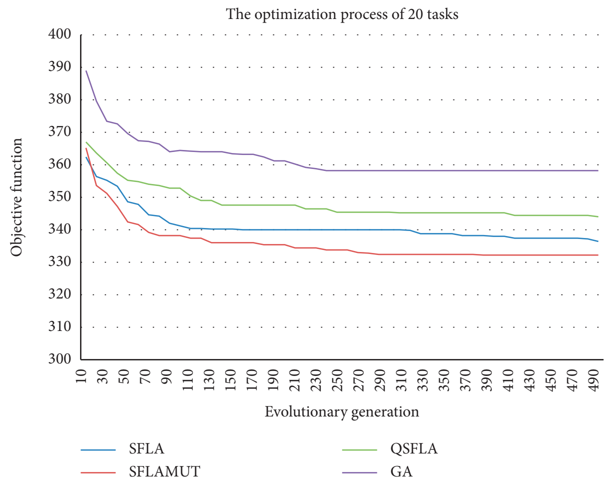

Figure 5: The optimization process of 20 tasks.

which enables SFLAMUT to randomly generate a solution compared with the local best solution, leading to more chances to find the global optimum solution when iteration times are fixed.

Based on previous analysis, we can conclude that the SFLAMUT algorithm has a better global searching ability and a faster convergence speed.

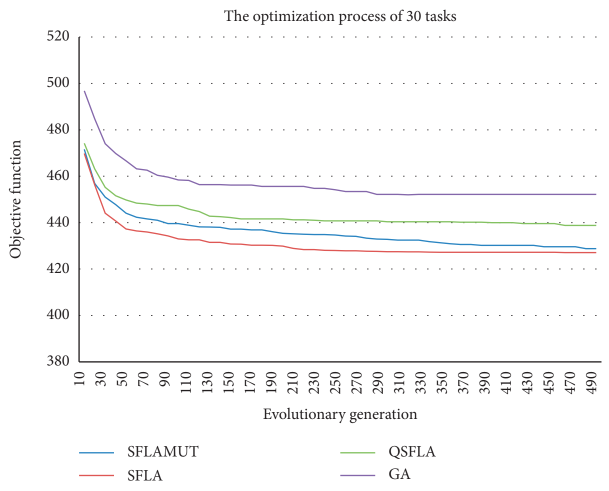

Figure 6: The optimization process of 30 tasks.

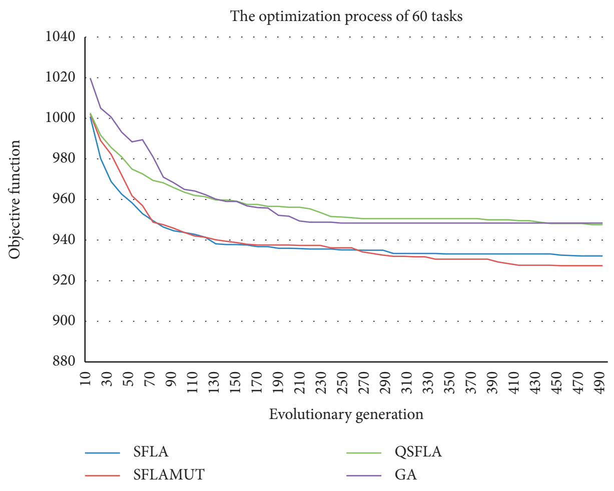

Figure 9: The optimization process of 60 tasks.

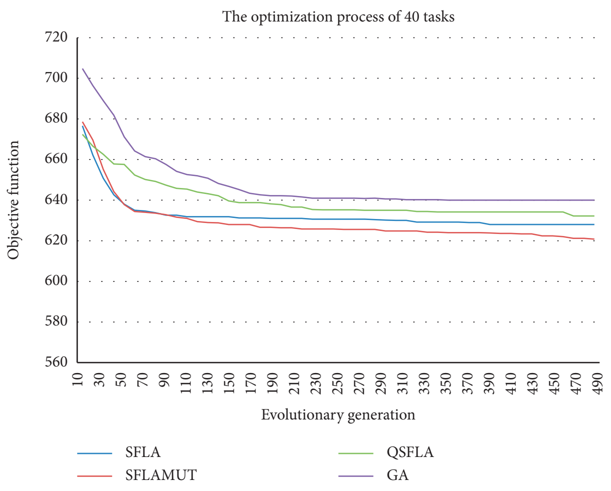

Figure 7: The optimization process of 40 tasks.

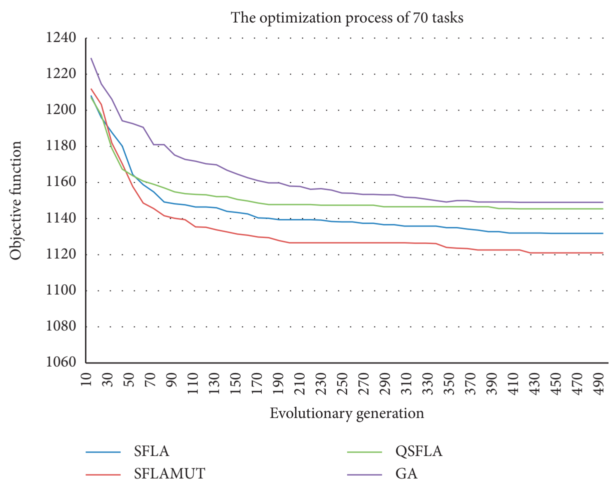

Figure 10: The optimization process of 70 tasks.

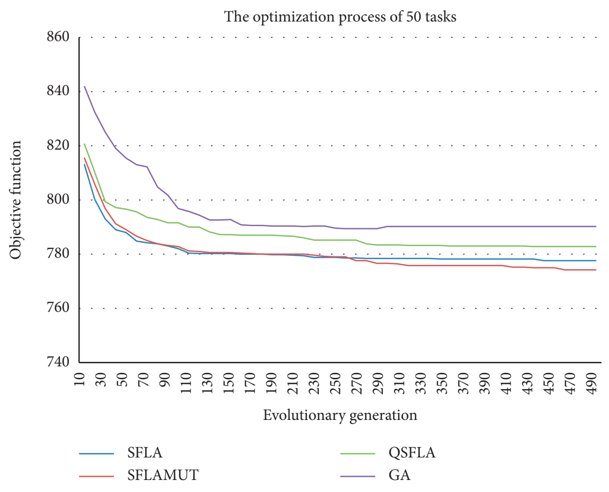

Figure 8: The optimization process of 50 tasks.

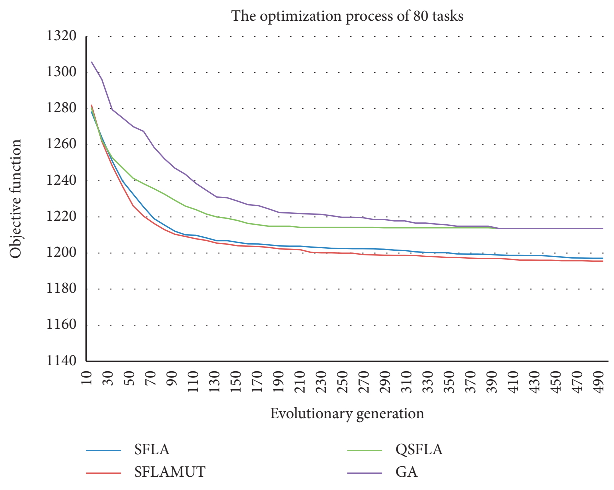

Figure 11: The optimization process of 80 tasks.

6. Conclusions

In this paper, we establish an improved mathematical model for multiload AGV dispatching in automated container terminals, aiming at fully utilizing the capacity of multiload AGVs and improving its operation efficiency further. To find the global optimal solution to this model, we propose the shortest distance rule-based shuffled frog leaping algorithm with a mutant process (SFLAMUT). The experimental results of SFLAMUT outperform the other three algorithms (GA, SFLA, and QSFLA).

Future research work can be extended to analyze the proposed model by considering other objective functions with respect to maximizing capacity utilization or minimizing total travel time. More in-depth research of other efficient algorithms to address the dispatching problem, such as machine learning algorithms, can be conducted based on the mathematical model proposed in this paper.

Data Availability

The data used in the experiments are generated randomly based on the actual flow of automated container terminals. All data included in this study are available upon request by contact with the corresponding author.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

The authors would like to acknowledge the support of the Fundamental Research Funds for the Central Universities.

References

[1] M. Grunow, H.-O. Günther, and M. Lehmann, “Dispatching multi-load AGVs highly automated seaport container terminals,” *Container Terminals and Automated Transport Systems*, pp. 231–255, Springer, Berlin, Germany, 2005.

[2] E. K. Bish, F. Y. Chen, Y. T. Leong, B. L. Nelson, J. W. C. Ng, and D. Simchi-Levi, “Dispatching vehicles in a mega container terminal,” *Container Terminals and Cargo Systems*, pp. 179–194, Springer, Berlin, Germany, 2007.

[3] R. Zaghdoud, K. Mesghouni, S. C. Dutilleul, K. Zidi, and K. Ghedira, “A hybrid method for assigning containers to AGVs in container terminal,” *IFAC-PapersOnLine*, vol. 49, no. 3, pp. 96–103, 2016.

[4] P. Angeloudis and M. G. H. Bell, “An uncertainty-aware AGV assignment algorithm for automated container terminals,” *Transportation Research Part E: Logistics and Transportation Review*, vol. 46, no. 3, pp. 354–366, 2010.

[5] H. Rashidi and E. P. K. Tsang, “A complete and an incomplete algorithm for automated guided vehicle scheduling in container terminals,” *Computers & Mathematics with Applications*, vol. 61, no. 3, pp. 630–641, 2011.

[6] T. Miyamoto and K. Inoue, “Local and random searches for dispatch and conflict-free routing problem of capacitated AGV systems,” *Computers & Industrial Engineering*, vol. 91, pp. 1–9, 2016.

[7] M. Saidi-Mehrabad, S. Dehnavi-Arani, F. Evazabadian, and V. Mahmoodian, “An Ant Colony Algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs,” *Computers & Industrial Engineering*, vol. 86, pp. 2–13, 2015.

[8] P. Azimi, H. Haleh, and M. Alidoost, “The selection of the best control rule for a multiple-load AGV system using simulation and fuzzy MADM in a flexible manufacturing system,” *Modelling and Simulation in Engineering*, vol. 2010, Article ID 821701, 7 pages, 2010.

[9] H. Fazlollahtabar, M. Saidi-Mehrabad, and J. Balakrishnan, “Mathematical optimization for earliness/tardiness minimization in a multiple automated guided vehicle manufacturing system via integrated heuristic algorithms,” *Robotics and Autonomous Systems*, vol. 72, pp. 131–138, 2015.

[10] V. K. Chawla, A. K. Chanda, and S. Angra, “Multi-load AGVs scheduling by application of modified memetic particle swarm optimization algorithm,” *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 40, no. 3, p. 436, 2018.

[11] Y.-C. Ho and H.-C. Liu, “The performance of load-selection rules and pickup-dispatching rules for multiple-load AGVs,” *Journal of Manufacturing Systems*, vol. 28, no. 1, pp. 1–10, 2009.

[12] Y.-C. Ho and H.-C. Liu, “A simulation study on the performance of pickup-dispatching rules for multiple-load AGVs,” *Computers & Industrial Engineering*, vol. 51, no. 3, pp. 445–463, 2006.

[13] K. G. Huo and Z. H. Hu, “Multi-load AGV scheduling in automated container terminals based on genetic algorithm,” *Journal of Shanghai Maritime University*, vol. 37, no. 3, pp. 7–12, 2016.

[14] K. G. Huo, Y. Q. Zhang, and Z. H. Hu, “Research on scheduling problem of multi-load AGV at automated container terminal,” *Journal of Dalian University of Technology*, vol. 56, no. 3, pp. 244–251, 2016.

[15] M. M. Eusuff and K. E. Lansey, “Optimization of water distribution Network design using the shuffled frog leaping algorithm,” *Journal of Water Resources Planning and Management*, vol. 129, no. 3, pp. 210–225, 2003.

[16] T. Niknam, M. R. Narimani, M. Jabbari, and A. R. Malekpour, “A modified shuffle frog leaping algorithm for multi-objective optimal power flow,” *Energy*, vol. 36, no. 11, pp. 6420–6432, 2011.

[17] W. Ding and J. Wang, “A novel approach to minimum attribute reduction based on quantum-inspired self-adaptive cooperative co-evolution,” *Knowledge-Based Systems*, vol. 50, no. 3, pp. 1–13, 2013.

[18] L. Wang and Y. Gong, “Quantum binary shuffled frog leaping algorithm,” in *Proceedings of the 2013 Third International Conference on Instrumentation, Measurement, Computer, Communication and Control*, pp. 1655–1659, Shenyang, China, September 2013.

[19] J. Sun, B. Feng, and W. Xu, “Particle swarm optimization with particles having quantum behavior,” in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, vol. 1, pp. 325–331, Portland, OR, USA, June 2004.

[20] S. Y. Li and P. C. Li, “Quantum particle swarms algorithm for continuous space optimization,” *Chinese Journal of Quantum Electronics*, vol. 24, no. 5, pp. 569–574, 2007.

[21] F. J. Solis and R. J.-B. Wets, “Minimization by random search techniques,” *Mathematics of Operations Research*, vol. 6, no. 1, pp. 19–30, 1981.