sustainability

MDPI

Article

A Genetic Algorithm for Integrated Scheduling of Container Handing Systems at Container Terminals from a Low-Carbon Operations Perspective

Yan Zheng [1], Meixian Xu [1], Zhaohu Wang [2,*] and Yujie Xiao [3,*]

[1] College of Automobile and Traffic Engineering, Nanjing Forestry University, Nanjing 210037, China
[2] School of Marketing and Logistics Management, Nanjing University of Finance and Economics, Nanjing 210046, China
[3] Business School, Nanjing University, Nanjing 210093, China
* Correspondence: 1120200836@stu.nufe.edu.cn (Z.W.); yujiexiao@nufe.edu.cn (Y.X.); Tel.: +86-180-1292-9107 (Z.W.); +86-132-2205-3281 (Y.X.)

Abstract: At container terminals, quay cranes, yard trucks, and yard cranes are mainly used to transfer containers. Driven by the demand for a green and low-carbon economy, an integrated scheduling problem considering three types of handling equipment of container handling systems is studied. As the task of transferring each container is completed by the three handling equipment sequentially, the optimal solution may not be found by only studying one type of equipment separately from a green operations perspective. The inter-dependency of different equipment should be considered to guarantee the overall performance of container handling systems with low-carbon operations so as to reduce energy consumption. In this paper, this integrated problem is formulated as a mixed integer linear programming (MILP). Since the MILP cannot be applied to solve large-sized practical problems, a genetic algorithm (GA) is developed. In the proposed GA, a three-dimension chromosome representation is proposed, which integrates the coordination of three handling equipment. A new mechanism including three pairs of crossover and mutation is used in parallel in GA with the aim of enhancing the efficiency of searching for good solutions. Each pair of crossover and mutation is specific to one dimension of a solution. Moreover, a novel heuristic mutation is developed to diversify solutions. The computational results indicate that the developed solution method for the integrated scheduling problem is promising and the heuristic mutation can highly improve the solution quality.

Keywords: integrated scheduling; low-carbon; container handling systems; mixed integer linear programming; genetic algorithm

Citation: Zheng, Y.; Xu, M.; Wang, Z.; Xiao, Y. A Genetic Algorithm for Integrated Scheduling of Container Handing Systems at Container Terminals from a Low-Carbon Operations Perspective. *Sustainability* **2023**, *15*, 6035. https://doi.org/10.3390/su15076035

Academic Editors: Antonio Comi and Tamás Bányai

Received: 9 November 2022
Revised: 26 January 2023
Accepted: 20 March 2023
Published: 30 March 2023

1. Introduction

The efficiency and low-carbon operations of container terminals (CTs) have a direct impact on the success of international trade and a nation's green economy [1]. Meanwhile, governments introduce policies and regulations for organizations such as CTs to reduce carbon emissions [2–4]. The efficiency of CTs is usually indicated by the ship turnaround time. Terminals with shorter ship turnaround times and low-carbon operations can keep pace with the world trade competition. Thus, the cooperation of different activities with respect to three types of integrated handling equipment in CTs is needed to be considered.

Containers are boxes in a standard size (i.e., forty-foot-equivalent unit, twenty-foot-equivalent unit) in most CTs [5]. They are mainly handled and transferred by three types of equipment which are quay cranes (QCs), yard trucks (YTs), and yard cranes (YCs). In the studies of Abdelmaguid et al. [6], Deroussi et al. [7], and Zheng et al. [8], the importance of the integrated scheduling of cooperated resources in flexible manufacturing systems (FMS) has been addressed. The key difference in the operations of material handling equipment between FMS and CTs lies in the buffer space. There is buffer space between machines and

material handling equipment but no buffer space between container handling equipment in CTs. Due to the absence of buffer space between cranes and YTs, a blocking situation may happen during the process of transferring containers. When a QC or YC lifts a container, it must place the container onto an available YT instead of placing it on the ground. As such, QCs and YCs cannot continue their next task until an empty YT has arrived. In another aspect, containers have to remain on YTs until QCs or YCs lift the containers off YTs. Such a blocking situation happens because of the poor coordination between cranes and YTs. The ignorance of interaction between cranes and YTs would lead to local optimality and consequently result in the loss of performance. Therefore, the overall performance with good efficiency and low-carbon emission can be guaranteed when QCs, YTs, and YCs are simultaneously considered in an integrated model.

This paper studies an integrated problem of concurrently scheduling QCs, YTs, and YCs with low-carbon operations, named P_QYY. The objective is to minimize the ship turnaround time (i.e., makespan) from the low-carbon perspective. As introduced by Zeng & Yang [9], the solution methodologies were applied to schedule the loading operations for outbound containers and the unloading operations for inbound containers, respectively. In this paper, the process of loading operations is studied. However, the proposed solution approach is not limited to loading operations, also valid for unloading operations. In our problem, QCs assignment to ships and locations of containers in ships are known in advance. The sequence of handling containers by each QC is to be determined. Meanwhile, the determination on YTs assignment to containers and YCs assignment to blocks is to be made. YCs shared by different blocks could enhance the utilization of YCs and then reduce potential investment costs [10]. Thus, the purpose of P_QYY is to resolve three sub-problems: the sequence of loading containers on ships by each QC, YTs assignment to containers, and YCs assignment to blocks.

A mixed integer linear programming (MILP) model is formulated to determine the optimal solution for P_QYY. However, since even the sub-problem YT scheduling problem is of NP-hard class [11], P_QYY is also an NP-hard problem as it can be reduced to a YT scheduling problem when the other two sub-problems are treated as the input data. To solve large-sized practical problems, a solution method based on a genetic algorithm (GA) is developed. The experimental results reveal that the proposed algorithm is efficient to find good solutions for P_QYY.

2. Literature Review

Considerable research has been conducted in the field of CTs as a result of the increasing importance of port container terminal systems. Previous research mainly focused on only one aspect of terminal scheduling problems, such as QC scheduling problems, YT scheduling problems, and YC scheduling problems.

Meisel & Bierwirth [12] stated diverse models for QCs scheduling problems. A unified approach and a new platform were presented for the evaluation of distinct models and solution procedures. Liu et al. [13] proposed the QC scheduling problem with non-crossing constraints for QCs and precedence constraints for container tasks. The authors also studied the computational complexity of the problem and developed a GA to get near-optimal solutions. AI-Dhaheri et al. [14] studied the QC scheduling problem while considering the dynamics and the uncertainty inherent to the container handling process. A simulation-based GA was presented to solve this problem.

Ng et al. [11] studied a fleet of trucks scheduling problem with the constraints of different ready times and sequence-dependent processing times with the goal of minimizing the makespan. A GA with a greedy crossover was proposed to search the effective schedules. He et al. [15] addressed the scheduling problem for the internal trucks (IT) in CTs with the consideration of IT assignment strategy. Specially, an approach that dealt with the sharing of ITs among multiple CTs were designed and investigated. To efficiently solve this problem, the rolling-horizon approach together with a simulation optimization method was presented and verified.

Ng & Mak [16] made a study on a single YC scheduling problem with the objective of minimizing the sum of job waiting times. In their problem, both inbound containers and outbound containers were involved. In the study of Jung & Kim [17], a single YC scheduling problem was extended to multiple YC scheduling problems. Multiple YCs operating in the same block were addressed and only loading operation was considered. He et al. [10] proposed a hybrid GA to deal with the YCs scheduling problem. The necessity of YC movement among blocks was introduced. Wu et al. [18] studied the multiple YC scheduling problems while considering the operational constraints in practice such as the crane non-crossing constraint. A clustering-reassigning approach was developed to find satisfactory near-optimal solutions.

The studies addressed above focused on the scheduling problem of one single-handling equipment. The interrelation between cranes and trucks/vehicles was not involved by these authors, which resulted in poor performance of CTs with much energy consumption and low working efficiency. In recent years, integrated problems of concurrently studying the interrelated decision-making processes in CTs are considered.

Bierwirth & Meisel [19] made a survey of berth allocation and QC scheduling problems in CTs. They remarked that the integration models would receive much attention and be a new trend of future research in CTs. Rodrigues & Agra [20] surveyed the integrated problem of berth allocation and QCs assignment problem. They addressed the research trends and future study direction together with the limitation of published papers. In the study of Tang et al. [21], a solution method incorporating the discretization strategy and a MILP was presented to deal with the continuous berth allocation and QCs assignment problem. Cho et al. [22] developed an integrated method for berth allocation and QCs assignment problems, of which the reassignment of the vessel to other terminals was allowed. In Kaveshgar & Huynh [23], an integrated model for jointly scheduling QCs and YTs was proposed. A GA combined with a greedy algorithm was designed to solve this integrated scheduling problem. He [24] addressed an integrated berth allocation and QC assignment problem, in which the trade-off between time-saving and energy-saving at the operational level was studied. An integrated simulation and optimization method was used to solve the problem. Luo et al. [25] integrated vehicle scheduling and container storage problems in the unloading process in an automated container terminal. A MIP model and a GA were employed to solve this problem with the objective of minimizing the ship's berth time.

The importance of integrating various types of handling equipment in CTs has been stated in the literature ([1,26]). A tabu search algorithm was developed in the paper of Chen et al. [1] to solve the integrated scheduling problem of QCs, yard vehicles, and YCs. In their problem, each YC served at only one block. In the work of Lau & Zhao [26], an integrated model of scheduling QCs, automated guided vehicles, and automated YCs was introduced. In their model, each block was assumed to have a specified YC. These above-mentioned two models improved the coordination of different handling equipment in CTs.

The classification of scheduling problems in CTs are shown in Table 1.

Table 1. Classification of scheduling problems in CTs.

Scheduling Problems in CTs	Related Literature
QC scheduling problem	Meisel & Bierwirth [12], Liu et al. [13], AI-Dhaheri et al. [14]
YT scheduling problem	Ng et al. [11], He et al. [15]
YC scheduling problem	Ng & Mak [16], Jung & Kim [17], He et al. [10], Wu et al. [18]
Berth allocation and QC scheduling problem	Bierwirth & Meisel [19], Rodrigues & Agra [20], Tang et al. [21], Cho et al. [22], He [24]

Table 1. *Cont.*

Scheduling Problems in CTs	Related Literature
Integrated QC and YT scheduling problem	Kaveshgar & Huynh [23]
Integrated vehicles scheduling and containers storage problem	Luo et al. [25]
Integrated QC, yard vehicles, and YC scheduling problem	Chen et al. [1], Lau & Zhao [26]

This paper addresses the integrated problem of scheduling QCs, YTs, and YCs, where YCs assignment to different blocks is considered. The proposed model is a generalization of real practice in CTs where a dedicated YC is used for each block—a special case of P_QYY. For this special case, a YCs assignment is not needed. This model can also be applied to CTs where dedicated YCs are used if a restriction is given on the selection of YCs. In some CTs, not every block has a specified YC and thus YCs need to be shared among blocks. One such widely used YCs is the rubber-tired gantry cranes (RTGC). It moves on rubber-tired wheels and needs to move to another block after it finishes the task in the current block to well utilize RTGCs [27]. Due to its mobility and flexibility, RTGC is the one of most commonly used handling equipment in the port of Hong Kong and other CTs all over the world [28]. To fully utilize YCs, the movement of YCs among different blocks is necessary. This is also helpful to relax workload congestion and imbalance [10]. Moreover, the schedule of YCs is generally made by port managers based on their experience before yard operations, which cannot guarantee the efficiency of the YC schedule. As an inefficient YCs schedule could result in the loss of productivity in the storage yard thus affecting the performance of CTs, YCs movement among blocks and YCs assignment are considered in our integrated model.

3. MILP Model for P_QYY

The typical layout of most CTs is given in Figure 1. In a CT, there are mainly two working areas: the ship's operation area and the storage yard. QCs are located in the ship operation area and YCs serve at blocks in the storage yard. In front of each QC and each block, a transfer point is set for YTs to load/unload containers. For outbound containers, the task is to load containers in the storage yard onto ships along the seaside. Three phases are successively included to complete the loading process. First, YCs move into the storage yard to get containers and place them onto the assigned YTs. Then, YTs transfer containers to the transfer points of QCs. Finally, QCs are responsible for loading containers onto ships.

In P_QYY, a job is referred to as the complete process of transferring a container from a block to a QC. For each container, the block and QC are assigned in advance. The distance between any two transfer points is also known. In the integrated model, the following assumptions for P_QYY are adopted:

1. YTs/YCs employed in CTs are identical.
2. YTs/YCs are driven at a specific speed.
3. Each YT has a unit capacity, i.e., only one container can be moved by a YT for a job.
4. The average handling time is set to be the handling time for each QC/YC. The time spent in transferring process from YC/YT to YT/QC is included in the handling time of YC/QC.
5. Containers assigned to one QC are loaded onto ships according to the first come first served rule. Special containers are not considered in this study.
6. Traffic congestion or conflict of YTs/YCs is not taken into consideration.

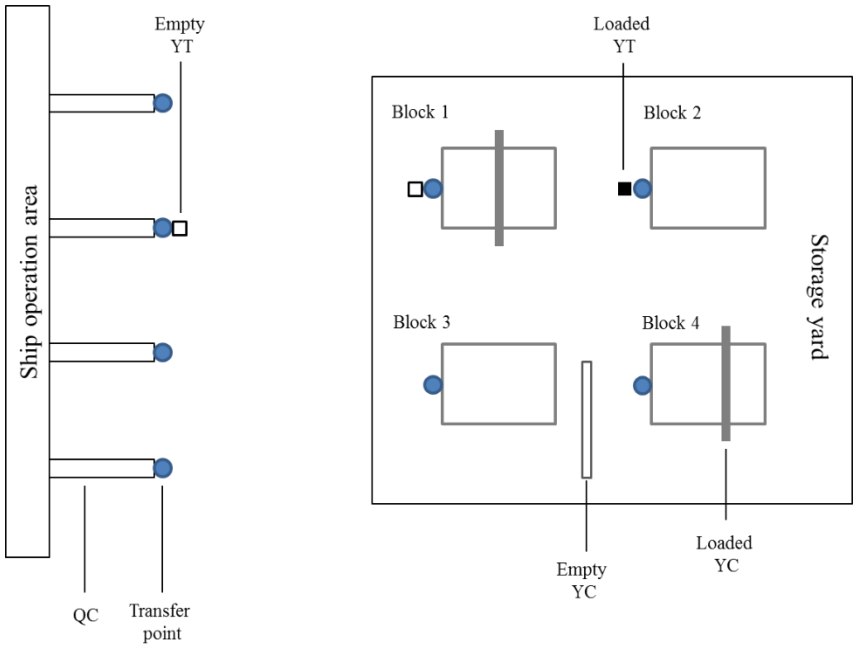

Figure 1. The layout of a CT.

The processing time of each job is the sum of the handling times of QC and YC, the travel time of YT from the origin to the destination of the job, and the waiting time. A job is in the condition of waiting when the required handling equipment is not available or in a blocking situation. The efficiency can be improved by reducing the waiting time of each job and the empty trips of YTs and YCs. Appropriate loading sequences of QCs as well as appropriately dispatching YTs and YCs to containers can significantly decrease the empty trips and the waiting time.

To describe P_QYY, notations are demonstrated as follows:

Indices:

i, *j:* Jobs, $i, j = 1, 2, \ldots, Njob$, *Njob* is the total number of jobs
q: QCs, $q = 1, 2, \ldots, Nqc$, *Nqc* is the total number of QCs
k: YTs, $k = 1, 2, \ldots, Nyt$, *Nyt* is the total number of YTs
e: YCs, $e = 1, 2, \ldots, Nyc$, *Nyc* is the total number of YCs
m, *n:* Locations of QCs and blocks

Parameters:

$QC(i)$: QC preassigned the container of job *i*
$B(i)$: Block where the container of job *i* is placed
ipt_k: The initial position of YT *k*
ipc_e: The initial position of YC *e*
D_{mn}: Distance between *m* and *n*
hqc: Handling time spent by a QC
hyc: Handling time spent by a YC
syt: Driving speed of YTs
syc: Driving speed of YCs
M: A very large positive constant

Sets:

L: Set of outbound containers
SQ: Set of QCs
ST: Set of YTs
SC: Set of YCs
SP_q: Set of pairs of jobs (i,j) where $QC(i) = QC(j) = q$

Decision variables:

C_{max}: Makespan
Tyt_i: Arrival time of the container of job i at $QC(i)$ by YT
Tyc_i: Arrival time of the container of job i at $B(i)$ by YC
c_i: Completion time of job i

$$u_{ik} = \begin{cases} 1, \text{ if job } i \text{ is assigned to YT } k \\ 0, \text{ otherwise.} \end{cases}$$

$$v_{ie} = \begin{cases} 1, \text{ if job } i \text{ is assigned to YC } e \\ 0, \text{ otherwise.} \end{cases}$$

$$x_{ijq} = \begin{cases} 1, \text{ if job } i \text{ precedes job } j \text{ on } QC\ q \text{ and } (i,j) \in SP_q \\ 0, \text{ otherwise.} \end{cases}$$

$$y_{ijk} = \begin{cases} 1, \text{ if job } i \text{ is immediately before job } j \text{ on YT } k \\ 0, \text{ otherwise.} \end{cases}$$

$$z_{ije} = \begin{cases} 1, \text{ if job } i \text{ is immediately before job } j \text{ on YC } e \\ 0, \text{ otherwise.} \end{cases}$$

$$f1_{ik} = \begin{cases} 1, \text{ if job } i \text{ is the first task of YT } k \\ 0, \text{ otherwise.} \end{cases}$$

$$f2_{ie} = \begin{cases} 1, \text{ if job } i \text{ is the first task of YC } e \\ 0, \text{ otherwise.} \end{cases}$$

The objective of P_QYY is the minimization of the makespan described in Constraint (1).

$$\text{Minimize } C_{max} \tag{1}$$

Constraint (2) defines the makespan.

$$C_{\max} \geq c_i,\ \forall i \in L \tag{2}$$

In Constraint (3), the completion time of each job is calculated. The container is handled by its preassigned QC after the container arrives at this QC.

$$c_i - hqc \geq Tyt_i,\ \forall i \in L \tag{3}$$

Constraint (4) implies that the container of each job is handled by the assigned YT after the container arrives at the transfer part of the block by its assigned YC. $Tyt_i - D_{B(i)QC(i)}/syt$ is the beginning time of YT's working on this container.

$$Tyt_i - D_{B(i)QC(i)}/syt \geq Tyc_i,\ \forall i \in L \tag{4}$$

Constraints (5a), and (5b) define the blocking situation. Constraint (5a) describes the situation where job i is handled before job j on YC e. Constraint (5b) describes the situation where the container of job i is handled before the container of job j on YT k. The handling equipment (YC or YT) can handle its next task job j after this equipment is released by the next handling equipment of job i. The container of job i will have to remain on the current handling equipment until the next equipment of job i takes this container off the current handling equipment.

$$Tyc_j - hyc + M(1 - z_{ije}) \geq Tyt_i - D_{B(i)QC(i)}/syt + D_{B(i)B(j)}/syc,\ \forall i,j \in L, \forall e \in SC \tag{5a}$$

$$Tyt_j - D_{B(j)QC(j)}/syt + M(1 - y_{ijk}) \geq c_i - hqc + D_{QC(i)B(j)}/syt,\ \forall i,j \in L, \forall k \in ST \tag{5b}$$

Constraint (6a)/(6b) ensures that YTs/YCs begin their first task from the initial position.

$$Tyt_i + M(1 - f1_{ik}) \geq (D_{ipt_kB(i)} + D_{B(i)QC(i)})/syt,\ \forall i \in L, \forall k \in ST \tag{6a}$$

$$Tyc_i + M(1 - f2_{ie}) \geq D_{ipc_eB(i)}/syc + hyc,\ \forall i \in L, \forall e \in SC \tag{6b}$$

Constraint (7a)/(7b) forces the containers of job (i, j) on QC q cannot be handled at the same time.

$$c_j - c_i + M(1 - x_{ijq}) \geq hqc,\ \forall (i,j) \in SP_q, \forall q \in SQ \tag{7a}$$

$$c_i - c_j + Mx_{ijq} \geq hqc,\ \forall (i,j) \in SP_q, \forall q \in SQ \tag{7b}$$

Constraint (8a)/(8b) makes sure that the container of each job must be assigned to one YT/YC.

$$\sum_{k=1}^{Nyt} u_{ik} = 1,\ \forall i \in L \tag{8a}$$

$$\sum_{e=1}^{Nyc} v_{ie} = 1,\ \forall i \in L \tag{8b}$$

Constraint (9a)/(9b) describes the relationship between the YT k/YC e assignment and the handling sequence on YT k/YC e.

$$u_{ik} + u_{jk} - 2y_{ijk} \geq 0,\ \forall i, j \in L, \forall k \in ST \tag{9a}$$

$$v_{ie} + v_{je} - 2z_{ije} \geq 0,\ \forall i, j \in L, \forall e \in SC \tag{9b}$$

Constraint (10a)/(10b) determines the first task for each YT/YC.

$$\sum_{i=1}^{Njob} f1_{ik} = 1,\ \forall k \in ST \tag{10a}$$

$$\sum_{i=1}^{Njob} f2_{ie} = 1,\ \forall e \in SC \tag{10b}$$

Constraint (11a)/(11b) describes the relationship between the YT k/YC e assignment and the first task of job i.

$$f1_{ik} - u_{ik} \leq 0,\ \forall i \in L, \forall k \in ST \tag{11a}$$

$$f2_{ie} - v_{ie} \leq 0,\ \forall i \in L, \forall e \in SC \tag{11b}$$

In Constraints (12a)–(12c), the handling sequence of containers on each YT is determined.

$$\sum_{k=1}^{Nyt} y_{ijk} + \sum_{k=1}^{Nyt} y_{jik} \leq 1,\ \forall i, j \in L \tag{12a}$$

$$\sum_{k=1}^{Nyt} (f1_{ik} + \sum_{\substack{\forall j \in L \\ i \neq j}} y_{jik}) = 1,\ \forall i \in L \tag{12b}$$

$$\sum_{k=1}^{Nyt} \sum_{\substack{\forall j \in L \\ i \neq j}} y_{ijk} \leq 1,\ \forall i \in L \tag{12c}$$

In Constraints (13a)–(13c), the handling sequence of containers on each YC is determined.

$$\sum_{e=1}^{Nyc} z_{ije} + \sum_{e=1}^{Nyc} z_{jie} \leq 1,\ \forall i, j \in L \tag{13a}$$

$$\sum_{e=1}^{Nyc} (f2_{ie} + \sum_{\substack{\forall j \in L \\ i \neq j}} z_{jie}) = 1, \ \forall i \in L \tag{13b}$$

$$\sum_{e=1}^{Nyc} \sum_{\substack{\forall j \in L \\ i \neq j}} z_{ije} \leq 1, \ \forall i \in L \tag{13c}$$

Constraint (14) defines the binary variables.

$$x_{ijq}, u_{ik}, v_{ie}, y_{ijk}, z_{ije}, f1_{ik}, f2_{ie} \in \{0, 1\}, \ \forall i, j \in L, \forall q \in SQ, \forall k \in ST, \forall e \in SC \tag{14}$$

4. Solution Method

Due to the NP-hardness of P_QYY, solving large-sized problems by exact methods is intractable. In this paper, a solution method based on a GA is designed to find the optimal or satisfactory solutions. The problem P_QYY is to simultaneously solve the three integrated sub-problems: the handling sequence of containers on QCs/YTs assignment and YCs assignment. To match this problem structure, a solution representation with three dimensions in GA is designed. It integrates that three sub-solution by considering the cohesive relation of QCs, YTs, and YCs. Accordingly, the three sub-solutions are searched by the three pairs of crossover and mutations. Furthermore, a novel heuristic mutation is designed to improve the possibility of finding high-quality solutions. The effect of the proposed algorithm is to shorten the empty ships and reduce the waiting time to improve the utilization and coordination of the handling equipment, and thereby finish all jobs as soon as possible.

The flowchart of the proposed GA is depicted in Figure 2. GA starts with reading the input data. The initial population is generated according to the given input data. In each generation, three pairs of crossover and mutation are implemented in parallel to generate offspring which are then evaluated. The new population is formed using a mixed selection strategy. GA searches solutions iteratively and is stopped when the stopping criterion is satisfied.

4.1. Chromosome Representation

For P_QYY, the chromosome is represented as a $3 \times Njob$ array, named π, which has 3 rows and *Njob* columns. The rows from top to bottom are named $\pi(1)$, $\pi(2)$, and $\pi(3)$, respectively. $\pi(h)$ is set as $\pi(h) = \{\pi(h)(1), \pi(h)(2), \ldots, \pi(h)(Njob)\}(h = 1, 2, 3)$. $\pi(1)$gives the loading sequence of containers on QCs. $\pi(2)$ shows the assignment of YTs to jobs in $\pi(1)$. $\pi(3)$ demonstrates the assignment of YCs to jobs in $\pi(1)$. The loading sequence of containers on YTs and YCs are determined from $\pi(2)$ and $\pi(3)$. The gene in each column of π describes YT/YC assignment to the job.

The potential benefit of this three-dimension representation is that it builds the links of the handling process of QC, YT, and YC for each job by combining them in the same column (as each job has a predefined QC). These links may be helpful for maintaining the highly fit chromosome patterns after performing crossover and mutation operators. The survival of these good chromosome patterns more facilitates the convergence of GA than an independent single-dimension solution representation.

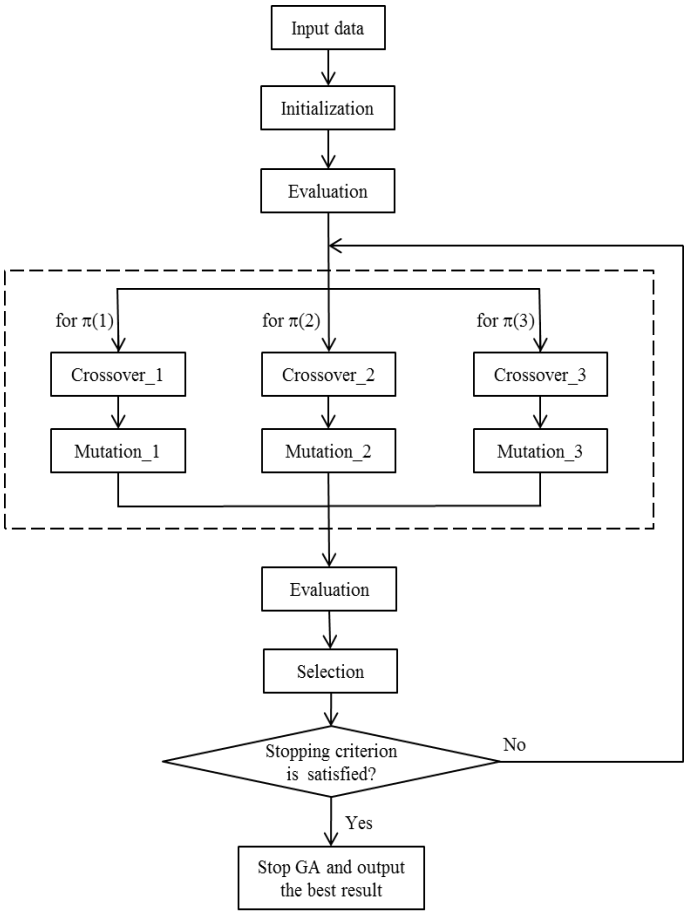

Figure 2. Flowchart of the proposed GA.

An example problem is given below to illustrate the chromosome representation. Consider a CT with three QCs and a storage yard consisting of three blocks. There are 10 jobs each of which has a predefined QC and block in Table 2. Numbers 1–10 in the first row indicate the jobs. The second and third rows give the information on the assigned QCs and blocks, with numbers 1–3 standing for QCs and 4–6 for blocks. There are a total of three YTs and two YCs. The initial position of each YT and YC is given in Table 3, where the numbers indicate QCs (1–3) or blocks (4–6).

Table 2. Assigned QCs and blocks for 10 jobs in the example problem.

Job i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$QC(i)$	2	3	2	1	3	3	2	2	3	1
$B(i)$	4	6	5	5	6	4	4	5	6	4

Table 3. The initial positions for YTs and YCs in the example problem.

	YTs			**YCs**	
	YT 1	**YT 2**	**YT 3**	**YC 1**	**YC 2**
Initial position	5	4	6	4	5

For this example problem, a feasible chromosome π can be defined as a 3×10 array with $\pi(1)$ = {2,4,1,6,3,8,9,7,10,5}, $\pi(2)$ = {2,1,1,3,2,3,1,2,2,3}, and $\pi(3)$ = {1,1,2,2,1,2,1,2,2,1}, which is shown in Figure 3a. $\pi(1)$ is the loading sequence of jobs, $\pi(2)$ represents YTs assignment to jobs in $\pi(1)$, and $\pi(3)$ represents YCs assignment to jobs in $\pi(1)$. For example,

in the first column, YT 2 and YC 1 are assigned to job 2. According to the given information on the preassigned QC and block for each job, this chromosome gives the loading sequences of jobs on QCs, YTs, and YCs in Figure 3b.

$\pi(1)$	2	4	1	6	3	8	9	7	10	5
$\pi(2)$	2	1	1	3	2	3	1	2	2	3
$\pi(3)$	1	1	2	2	1	2	1	2	2	1

(**a**)

Sequence of jobs on each QC	Sequence of jobs on each YT	Sequence of jobs on each YC
QC 1: 4,10	YT 1: 4,1,9	YC 1: 2,4,3,9,5
QC 2: 1,3,8,7	YT 2: 2,3,7,10	YC 2: 1,6,8,7,10
QC 3: 2,6,9,5	YT 3: 6,8,5	

(**b**)

Trips of YTs	Trips of YCs
YT 1: ipt1→B(4) →QC(4) →B(1) →QC(1) →B(9) →QC(9) i.e., 5→1→4→2→6→3	YC 1: ipc1→B(2) →B(4) →B(3) →B(9) →B(5) i.e., 4→6→5→5→6→6
YT 2: ipt2→B(2) →QC(2) →B(3) →QC(3) →B(7) →QC(7) →B(10) →QC(10) i.e., 4→6→3→5→2→4→2→4→1	YC 2: ipc2→B(1) →B(6) →B(8) →B(7) →B(10) i.e., 5→4→4→5→4→4
YT 3: ipt3→B(6) →QC(6) →B(8) →QC(8) →B(5) →QC(5) i.e., 6→4→3→5→2→6→3	

The numbers represent QCs and blocks in storage yard 1–3 for QCs and 4–6 for blocks.

(**c**)

Figure 3. An example problem. (**a**) A feasible chromosome for the example problem. (**b**) Sequences of jobs on QCs, YTs, and YCs retrieved from the solution in Figure 3a. (**c**) Trips of YTs and YCs were generated from the solution in Figure 3a.

Figure 3c shows the trips of YTs and YCs needed to complete 10 jobs. It can be seen that empty trips occur when YTs head for a new task or YCs move between different blocks to pick up the required containers.

4.2. Initial Population

In this paper, the initialization of each chromosome in the population includes three parts: $\pi(1)$ initialization, $\pi(2)$ initialization, and $\pi(3)$ initialization. $\pi(1)$ is randomly initialized as a loading sequence of *Njob* jobs. For each gene in $\pi(2)$, a YT k ($k \in ST$) is randomly created. For each gene in $\pi(3)$, a YC e ($e \in SC$) is randomly created.

4.3. Fitness Evaluation

For each chromosome, the objective function value is the completion time of the last job. Since P_QYY is a minimization problem and the objective function value (i.e., makespan) is always positive, the fitness value $fitness(n)$ associated with each chromosome π_n is the reciprocal of makespan, i.e., $fitness(n) = 1/C_{max}$.

The procedure of calculating the completion time of job $\pi(1)(i)$ (i = 1,2, . . . , *Njob*) in solution π is described below. The notations are first introduced as follows.

$pyc[\pi(3)(i)]$: Current position of YC $\pi(3)(i)$
$pyt[\pi(2)(i)]$: Current position of YT $\pi(2)(i)$
$ayc[\pi(3)(i)]$: Arrival time of YC $\pi(3)(i)$ for job $\pi(1)(i)$
$ayt[\pi(2)(i)]$: Arrival time of YT $\pi(2)(i)$ for job $\pi(1)(i)$
$ryc[\pi(3)(i)]$: Released time of YC $\pi(3)(i)$ for job $\pi(1)(i)$, i.e., the
$ryt[\pi(2)(i)]$: Released time of YT $\pi(2)(i)$ for job $\pi(1)(i)$
$fyc[\pi(3)(i)]$: Finish time of job $\pi(1)(i)$ on YC $\pi(3)(i)$
$fyt[\pi(2)(i)]$: Finish time of job $\pi(1)(i)$ on YT $\pi(2)(i)$
$fqc[QC(\pi(1)(i))]$: Finish time of job $\pi(1)(i)$ on $QC(\pi(1)(i))$

As a generated chromosome π satisfies the constraints (7a)–(13c) of the MILP model in Section 3, the completion times of jobs $\pi(1)(i)$ (i = 1, 2, . . . , *Njob*) are calculated one by one according to the loading sequence of jobs in $\pi(1)$ based on the constraints (2)–(6b). Since each job is successively handled by three handling equipment YC, YT, and QC, the core procedure of calculating the completion time consists of three steps: (1) the calculation of the finish time on YC, (2) the calculation of the finish time on YT, and (3) the calculation of the finish time on QC. The completion time of each job is equal to the finish time on its assigned QC. The blocking situation can be circumvented by the procedure of calculating the mekespan of each three-dimension solution representation. The containers in $\pi(1)$ are handled one by one. The released time together with the finish time of the handling equipment for its processing container is calculated. The released time of YTs/YCs means the unloading time of the current container away from YTs/YCs. It is influenced by two factors: one is the finish time to complete the current job of YT/YC, and the other is the time when next handling equipment in secession QC/YT can take the job away from this YT/YC. After the handling equipment is released, its operation on the next container will start. A small example is given to show the calculation procedure in Appendix A. The calculation procedure is described as follows.

Step 0 Initialization.
0.1 Set $i = 1$.
0.2 Set $pyc[e] = ipc_e$ for $\forall e \in SC$, and $pyt[k] = ipt_k$ for $\forall k \in ST$.
0.3 Set $fyc[e] = fyt[k] = fqc[q] = ryc[e] = ryt[k] = 0$ for $\forall e \in SC$, $\forall\ k \in ST$, $\forall q \in SQ$.

Step 1 Calculation of the finish time of job $\pi(1)(i)$ on YC $\pi(3)(i)$.
1.1 The arrival time and finish time of YC $\pi(3)(i)$ are calculated.
$ayc[\pi(3)(i)] = ryc[\pi(3)(i)] + D_{pyc[\pi(3)(i)],B(\pi(1)(i))}/syc$
$fyc[\pi(3)(i)] = ayc[\pi(3)(i)] + hyc$
1.2 The new position of YC $\pi(3)(i)$ is updated as $B(\pi(1)(i))$.
$pyc[\pi(3)(i)] = B(\pi(1)(i))$

Step 2 Calculation of the finish time of job $\pi(1)(i)$ on YT $\pi(2)(i)$.
2.1 The arrival time YT $\pi(2)(i)$ is calculated.
$ayt[\pi(2)(i)] = ryt[\pi(2)(i)] + D_{pyt[\pi(2)(i)],B(\pi(1)(i))}/syt$
2.2 YC π(3)(i) is released when job π(1)(i) is loaded onto YT π(2)(i). Thus, the released time of YC π(3)(i) is calculated below.
$ryc[\pi(3)(i)] = \max\{ayt[\pi(2)(i)], fyc[\pi(3)(i)]\}$
The finish time of YT $\pi(2)(i)$ is then calculated.
$fyt[\pi(2)(i)] = ryc[\pi(3)(i)] + D_{B(\pi(1)(i))QC(\pi(1)(i))}/syt$
2.3 The new position of YT $\pi(2)(i)$ is updated as $QC(\pi(1)(i))$.
$pyt[\pi(2)(i)] = QC(\pi(1)(i))$
2.4 YT $\pi(2)(i)$ is released when the assigned QC is available to pick up job $\pi(1)(i)$ away from this YT. Thus, the released time of YT $\pi(2)(i)$ is calculated below.
$ryt[\pi(2)(i)] = \max\{fyt[\pi(2)(i)], fqc[QC(\pi(1)(i))]\}$

Step 3 Calculation of the finish time of job $\pi(1)(i)$ on QC.

$$c_{\pi(1)(i)} = fqc[QC(\pi(1)(i))] = ryt[\pi(2)(i)] + hqc$$

Step 4 Calculation of C_{max}. If $i = Njob$, then the makespan is calculated as $C_{max} = \max\{c_{\pi(1)(i)} \mid i = 1,2, \ldots, Njob\}$, otherwise, $i = i + 1$ and go to Step 1.

4.4. Generation of New Population

In the proposed GA, based on the three-dimension chromosome, three pairs of crossover and mutation (Crossover_1 & Mutation_1, Crossover_2 & Mutation_2, Crossover_3 & Mutation_3) are accordingly designed to wildly search the solution space. Crossover_1 and Mutation_1 are used for $\pi(1)$ while keeping the assignment of YTs and YCs to each job unchanged. Crossover_2 and Mutation_2 are performed on $\pi(2)$ while keeping $\pi(1)$ and $\pi(3)$ unchanged. Crossover_3 and Mutation_3 are performed on $\pi(3)$ while keeping $\pi(1)$ and $\pi(2)$ unchanged. In each generation, the three pairs of crossover and mutation are applied in parallel to generate the offspring. By doing so, the exploration of GA is improved and the possibility of missing the solutions with good attributes is decreased.

4.4.1. Crossover

Based on the characteristics of $\pi(a)$ ($a = 1, 2, 3$), three crossover operators are developed for different purposes. For $\pi(1)$, as the simple crossover fails to generate valid permutation-structured offspring, Crossover_1 is implemented on the basis of order crossover to guarantee that the loading sequence of jobs remains feasible. For $\pi(2)$ and $\pi(3)$, Crossover_2 and Crossover_3 adopt simple two-point crossover. The detail of order crossover and two-point crossover is introduced in [29].

Figure 4 illustrates the order crossover for $\pi(1)$. Based on Parent 1, Child 1 is produced by performing Crossover_1 on two selected parent chromosomes Parent 1 and Parent 2. In Figure 4a, a sub-sequence of $\pi(1)$ in Parent 1 is randomly selected and copied to the corresponding positions in Child 1. The rest positions of Child 1 are filled with the remaining jobs of $\pi(1)$ in Parent 2. The filling jobs in Parent 2 are copied to Child 1 in accordance with the sequence of jobs in Parent 2. Note that Child 1 inherits YTs and YCs assignments to jobs from Parent 1. For example, in Parent 1, YT 1 and YC 2 are assigned to job 4. Thus, job 4 of Child 1 has the same assignments YT 1 and YC 2. In the same way, Child 2 is generated based on Parent 2 by performing Crossover_1 on Parent 1 and Parent 2 as shown in Figure 4b. Crossover_1 keeps the links of YT and YC assignments to each job when changing the loading sequence in $\pi(1)$.

Figure 5 illustrates the two-point crossover for $\pi(2)$. Taking Parent 1 as a base, Child 1 is produced by performing Crossover_2 on two selected parent chromosomes Parent 1 and Parent 2. In Figure 5a, two cut points: cut point 1 and cut point 2 are randomly determined. For the jobs between these two cut points, the YTs assignment in $\pi(2)$ of Parent 1 is copied to Child 1. YTs assignment to other jobs in Child 1 is the same as that of Parent 2. Child 1 inherits $\pi(1)$ and $\pi(3)$ directly from Parent 1. In the same way, Child 2 is produced based on Parent 2 by performing Crossover_2 on Parent 1 and Parent 2 as shown in Figure 5b.

For $\pi(3)$, Crossover_3 is the same as Crossover_2 using the two-point crossover but keeping $\pi(1)$ and $\pi(2)$ unchanged.

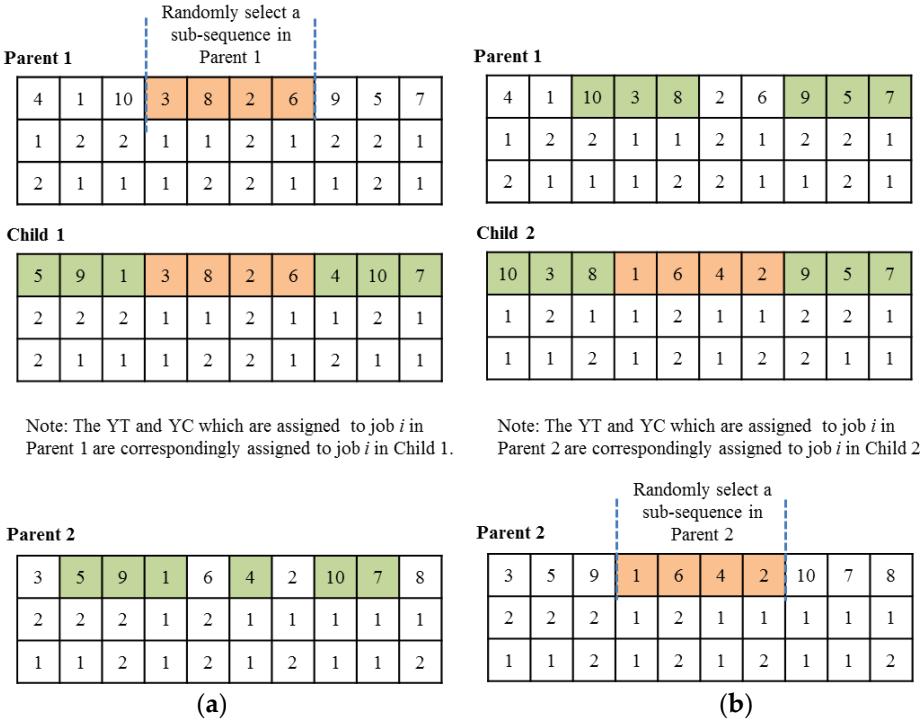

Figure 4. Illustration of the order crossover for $\pi(1)$. (**a**) Generation of Child 1, (**b**) Generation of Child 2.

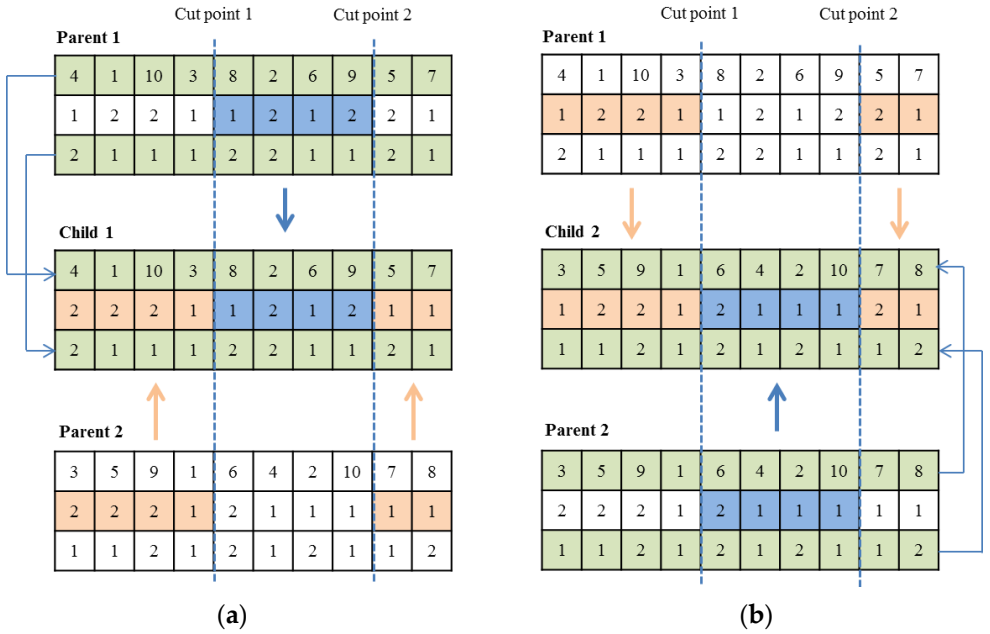

Figure 5. Illustration of the two-point crossover for $\pi(2)$. (**a**) Generation of Child 1, (**b**) Generation of Child 2.

4.4.2. Mutation

Mutation ensures that GA will have a chance to search other regions of the solution space and thus maintain genetic diversity from one generation to the next. Here different mutation operators are designed for $\pi(a)$ ($a = 1, 2, 3$). For $\pi(1)$, Mutation_1 is implemented based on the swap mutation (Goldberg 1989) [30] which mutates the selected parent chromosome by exchanging two randomly chosen jobs in $\pi(1)$ as well as their assigned YTs and YCs. Figure 6 shows Mutation_1 for $\pi(1)$.

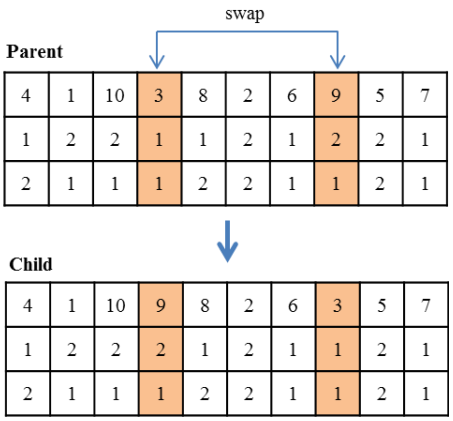

Figure 6. Illustration of Mutation_1 for $\pi(1)$.

Mutation_2 and Mutation_3 are specifically designed for $\pi(2)$ and $\pi(3)$, respectively. These two operators are based on a new heuristic mutation that seeks improvement and avoids local optimum by selecting the best-mutated chromosome as the offspring among all mutated neighbor chromosomes. In the proposed heuristic mutation, a specified number of neighbors are generated based on one chosen parent chromosome to find a more suitable schedule through the following procedure. In Mutation_2 (or Mutation_3), fixing $\pi(1)$ and $\pi(3)$ (or $\pi(1)$ and $\pi(2)$) of the parent chromosome, neighbors are created by changing a YT (or YC) assigned to job $\pi(1)(i)$ ($i = 1,2, \ldots Njob$) to any one of other ($Nyt - 1$) YTs (or ($Nyc - 1$) YCs). In this way, ($Nyt - 1$) neighbors are created by changing the YT assignment of one job in Mutation_2 (or ($Nyc - 1$) neighbors in Mutation_3). Thus, $Njob \times (Nyt - 1)$ neighbors are totally generated in Mutation_2 and $Njob \times (Nyc - 1)$ neighbors in Mutation_3. After implementing Mutation_2 or Mutation_3, all obtained neighbors are evaluated and the neighbor with the minimum makespan is chosen as the mutated offspring chromosome. It can be seen that the number of generated neighbors depends on the size of integrated problems, i.e., the total number of jobs and the total number of YT or YC.

The reason for developing this heuristic mutation is that the quality of the chromosome in each generation has an important impact on the overall performance of the next generation. This operator behaves as a local search to improve the chromosome quality and to avoid the local optimum by exploiting better-mutated chromosomes in the way of searching neighbors of parent chromosomes in each generation.

4.4.3. Selection

A mixed selection strategy including the deterministic selection and roulette wheel selection is used. The last population and all generated offspring are ranked in non-ascending order in terms of fitness value. Part of the new population is formed by deterministic selection which selects the best K chromosomes with different fitness values. Here *popsize* denotes the population size in GA. The remaining *popsize*–K chromosomes are generated using one of the most commonly used selection rules roulette wheel selection.

Based on the preliminary experiments, the drawback of only using the deterministic selection (i.e., $K = popsize$) is that the disappearance of worse individuals may result in losing the useful attributes and thereby cause premature convergence. In another aspect, by only using roulette wheel selection, the elite individuals may be lost or destroyed by crossover and mutation operators. The deterministic selection forces GA to retain K elite individuals in each generation. The significant role of elitism in improving GA performance has been verified by many researchers [31]. Another shortcoming of only using roulette wheel selection is that many same chromosomes may be selected in the next generation. The phenomenon that too many same chromosomes exist in the population is another cause of premature convergence [32]. Therefore, this mixed selection strategy overcomes the shortcoming of only using the deterministic selection or roulette wheel selection.

4.5. Stopping Criterion

In this paper, the stopping criterion is the allowed maximum number of generations (*Max_Gen*). When *Max_Gen* is reached, GA is stopped.

5. Computational Experiments and Analysis

The MILP model and the solution method described in the above sections have been applied to 25 (*ex*1–*ex*25) problem instances. The experimental data was generated based on the published studies [26,32]. In the CT, 6 QCs berth along the seaside and 20 blocks are set in the storage yard. For each container, the preassigned QC and block are created in a random way. YTs and YCs start at their initial positions which are set randomly. YTs drive at the speed of 4 m/s, and YCs at 3 m/s. The handling time of QCs is 60 s, and the handling time of YCs is 100 s. Other information about 25 problem instances including the total number of jobs, the total time of QCs, the total number of YTs, and the total number of YCs are given in Table 4.

Table 4. Details of 25 problem instances.

Problem Instances	*Njob*	*Nqc*	*Nyt*	*Nyc*
*ex*1	6	1	2	2
*ex*2	6	2	2	2
*ex*3	8	1	2	2
*ex*4	8	2	2	2
*ex*5	8	2	3	2
*ex*6	10	2	2	2
*ex*7	10	2	3	2
*ex*8	20	2	3	2
*ex*9	20	3	4	3
*ex*10	30	3	6	3
*ex*11	30	3	8	4
*ex*12	50	3	8	4
*ex*13	50	4	10	4
*ex*14	80	3	8	4
*ex*15	80	4	10	5
*ex*16	100	4	10	6
*ex*17	100	5	14	6
*ex*18	200	4	16	6
*ex*19	200	5	18	8
*ex*20	200	6	20	8
*ex*21	400	4	20	8
*ex*22	400	5	24	10
*ex*23	500	5	20	10
*ex*24	500	6	24	12
*ex*25	500	6	28	12

The proposed algorithm has been implemented using C++ language. The MILP model has been solved using ILOG CPLEX 12.2. All computation examples have been carried out on the personal computer with Inter Core 2, 3.30 GHz CPU. The parameters of the proposed algorithm include the population size *popsize*, probability of crossover P_c, probability of mutation P_m, and the number of elite chromosomes to be chosen in each generation K and *Max_Gen*. To determine the values of parameters, preliminary experiments were conducted. The best set of parameters chosen for the computational experiments is *popsize* = 100, P_c = 0.8, P_m = 0.2, K = 50, and *Max_Gen* = 1000 for *ex*1–*ex*15; *Max_Gen* = 2000 for *ex*16–*ex*25.

To assess the performance of the solution method, the results found by GA are compared with the results from MILP in terms of makespan and CPU time. For each instance, the algorithm has been conducted ten times. The best result obtained together with its CPU time was set to be the final result, which was reported in Table 5. For *ex*1 and *ex*2, the

optimal solution has been found by MILP, since the problem size of these two instances are small. For other instances, the running time of CPLEX was set to 36 h. The best result obtained during that time period was recorded for MILP. As seen from Table 5, the computational effort required for MILP to solve even small-sized problem instances is prohibitively large. For large-sized problems, the feasible solutions cannot be found by MILP within 36 h. MILP could find the optimal or feasible solutions for seven out of 25 problem instances. For *ex*3–*ex*7, the solutions found by GA in a few seconds are better than the solutions found by MILP within 36 h. Gaps between the results from MILP and GA are also calculated. Results in Table 5 indicate that MILP cannot solve medium-sized or large-sized P_QYY within reasonable time periods. The proposed GA can efficiently find good solutions for P_QYY.

Table 5. Comparison of results between MILP and the proposed GA.

Problem Instances	MILP		GA		Gap (%)
	Makespan (A1)	CPU Time	Makespan (A2)	CPU Time (Seconds)	(A1–A2)/A2
*ex*1	1328.33	64.45 s	1328.33	0.80	0.00
*ex*2	1335.33	38.36 s	1335.33	0.91	0.00
*ex*3	1408.00	36 h	1356.00	1.57	3.83
*ex*4	1363.00	36 h	1363.00	1.56	0.00
*ex*5	1374.33	36 h	1359.00	1.61	1.13
*ex*6	1988.33	36 h	1798.67	1.75	10.54
*ex*7	1661.00	36 h	1547.33	1.83	7.35
*ex*8	–	36 h	1565.67	2.11	–
*ex*9	–	36 h	1928.00	3.49	–
*ex*10	–	36 h	1575.00	4.18	–
*ex*11	–	36 h	1813.13	5.63	–
*ex*12	–	36 h	3149.67	11.84	–
*ex*13	–	36 h	2669.67	12.40	–
*ex*14	–	36 h	4809.67	26.08	–
*ex*15	–	36 h	3898.33	31.28	–
*ex*16	–	36 h	4412.33	105.44	–
*ex*17	–	36 h	3975.00	127.12	–
*ex*18	–	36 h	7881.67	547.49	–
*ex*19	–	36 h	6136.00	635.98	–
*ex*20	–	36 h	6011.33	662.59	–
*ex*21	–	36 h	12,447.00	2680.81	–
*ex*22	–	36 h	9843.00	3103.88	–
*ex*23	–	36 h	13,613.00	4304.06	–
*ex*24	–	36 h	10,305.33	5157.29	–
*ex*25	–	36 h	9685.00	5704.66	–

"–" indicates that the feasible solution is not obtained.

To validate the effectiveness of the proposed heuristic mutations, comparative experiments are carried out between the proposed GA with heuristic mutations (GA_HM) and GA with simple mutations (GA_SM). In GA_SM, heuristic mutations are replaced with simple mutations, which mutate the selected parent chromosome by randomly changing

the assigned YT (or YC) of a randomly chosen job to a different YT (or YC). Thus, GA_HM can be considered an enhanced version of GA_SM. The same parameter set is used for two versions of GA. Table 6 shows the results obtained from GA_HM and GA_SM. Each problem instance is executed ten times and the best results are reported for comparison. The gaps between the results of the two GAs are given in Table 6.

Table 6. Comparison of results between GA_HM and GA_SM.

Problem Instances	GA_HM		GA_SM		Gap (%)
	Makespan (B1)	CPU Time (Seconds)	Makespan (B2)	CPU Time (Seconds)	(B2–B1)/B1
*ex*1	1328.33	0.80	1328.33	0.72	0.00
*ex*2	1335.33	0.91	1335.33	0.85	0.00
*ex*3	1356.00	1.57	1356.00	1.66	0.00
*ex*4	1363.00	1.56	1363.00	1.63	0.00
*ex*5	1359.00	1.61	1359.00	1.38	0.00
*ex*6	1798.67	1.75	1798.67	2.08	0.00
*ex*7	1547.33	1.83	1547.33	2.22	0.00
*ex*8	1565.67	2.11	1565.67	2.42	0.00
*ex*9	1928.00	3.49	1931.33	4.06	0.17
*ex*10	1575.00	4.18	1592.33	3.21	1.10
*ex*11	1813.13	5.63	1875.00	3.70	3.41
*ex*12	3149.67	11.84	3258.00	4.94	3.44
*ex*13	2669.67	12.40	2877.33	5.13	7.78
*ex*14	4809.67	26.08	5294.00	7.03	10.07
*ex*15	3898.33	31.28	4214.67	7.20	8.11
*ex*16	4412.33	105.44	4704.33	17.06	6.62
*ex*17	3975.00	127.12	4222.33	13.26	6.22
*ex*18	7881.67	547.49	8854.67	25.43	12.35
*ex*19	6136.00	635.98	7480.33	24.27	21.91
*ex*20	6011.33	662.59	7269.67	23.89	20.93
*ex*21	12,447.00	2680.81	15,674.00	56.09	25.93
*ex*22	9843.00	3103.88	13,254.00	57.25	34.65
*ex*23	13,613.00	4304.06	17,192.67	80.53	26.30
*ex*24	10,305.33	5157.29	13,287.67	79.29	28.94
*ex*25	9685.00	5704.66	13,202.33	80.92	36.32

From Table 6, it is shown that GA_HM and GA_SM can find the same solutions for the small-sized problem instances (*ex*1–*ex*8) in a short time period. However, the gaps become bigger from 0.17% to 36.32% as the problem size increases. GA_HM provides better results than GA_SM, especially for very large-sized problems. The improved results do come at the cost of CPU time. This is because the number of neighbor chromosomes generated in the heuristic mutations is enlarged as the problem size increases. For the largest problems (*ex*23–*ex*25), it takes more than 1 h to get the solutions with approximately 30% improvement. Given that a detailed loading schedule should be made a few hours before the ship arrives at the terminal, such CPU time is still justified in practical situations.

The convergence speed of GA_HM and GA_SM is examined by revealing the value of makespan changing with each generation for the largest problem instance *ex*25 in Figure 7. Both GA_HM and GA_SM could reach convergence within 1000 generations. Nevertheless, GA_HM could find much better solutions than GA_SM. This is because the heuristic mutations improve solutions quality by finding the best neighbor chromosome that possesses potential good patterns. In conclusion, the comparison results indicate that two GAs are efficient for solving P_QYY and GA_HM can efficiently find better solutions than GA_SM for large-sized practical problems.

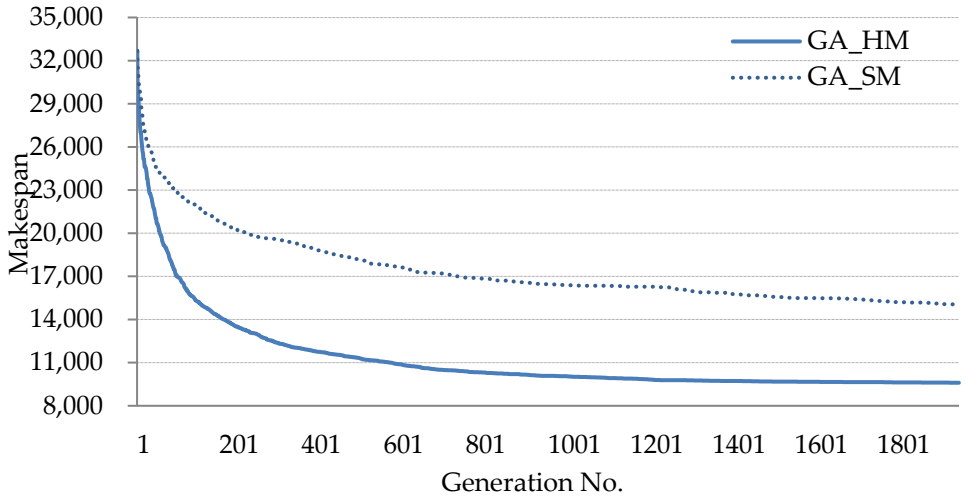

Figure 7. The convergence of GA_HM and GA_SM.

6. Conclusions

From a low-carbon operations perspective, this paper studies the integrated scheduling problem of QCs, YTs, and YCs in CTs by considering the interconnections between the three types of handling equipment. To solve this integrated model, a MILP is formulated in this paper. Due to large numbers of variables and constraints, the MILP model is intractable for large-sized practical problems. A GA with three-dimension chromosome representation is developed, which links the interrelation of the three sub-problems. Based on the chromosome representation, a mechanism with three pairs of crossover and mutation is introduced. Each pair of crossover and mutation is specific to one dimension of a chromosome. This mechanism in GA well matches the property of the integrated problem. Moreover, to enhance the performance of the proposed solution method, a new heuristic mutation is proposed. Better solutions are found by applying the three pairs of crossover and mutation in parallel. The efficiency of the proposed algorithm as well as the heuristic mutation is shown by the computational results.

The developed solution method is not limited to the integrated model presented in this paper. Moreover, the proposed algorithm with a multi-dimension solution representation and a combined crossovers and mutations mechanism can be flexibly applied to other integrated problems with some modifications. In our future research, more practical constraints will be taken into consideration to consummate the P_QYY model, such as YTs/YCs conflicts, the allocation of storage space, the due date or the ready time of containers, etc. In addition, the effort on extending our proposed solution method for integrating problems in other related research fields will be made.

Author Contributions: Y.Z. and M.X. conceived and designed the experiments; Y.X. and Z.W. performed the experiments; Z.W. analyzed the data; Y.Z. and Y.X. wrote the paper. All authors have read and agreed to the published version of the manuscript.

Funding: The research has been supported by the National Natural Science Foundation of China (Grant No. 71701099, 71871111), Social Science Foundation of Jiangsu Province (No. 19GLC011), the Research Foundation of the Jiangsu (No. KJB17580008) and the Natural Science Foundation of Jiangsu Higher Education Institutions of China (Grant No. 18KJA410001,17KJB580008).

Acknowledgments: The authors are grateful to the editors and anonymous reviewers for providing the valuable comments.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

Appendix A

To be simplicity, a small example is given of which the input data is described in Tables A1–A3. In this example, 2 QCs, 2 blocks, 2 YTs and 2 YCs are used to finish 4 jobs and the speeds of YTs and YCs are set to be 1. The average handling time of QCs and YCs are 60s and 100s, respectively. QCs are numbered as 1, 2 and blocks as 3, 4. Assume that the chromosome π to be evaluated is the one as shown in Figure A1.

Table A1. Distance matrix for the example problem.

D_{mn}	1	2	3	4
1	–	100	150	200
2	100	–	200	150
3	150	200	–	100
4	200	150	100	–

Table A2. Assignment of QC and block for each job.

Job i	1	2	3	4
$QC(i)$	2	1	2	1
$B(i)$	3	3	4	4

Table A3. The initial position of each YT/YC.

	YT 1	YT 2	YC 1	YC 2
Initial position	3	4	4	3

$\pi(1)$	2	4	1	3
$\pi(2)$	2	1	1	2
$\pi(3)$	1	1	2	2

Figure A1. Chromosome π of the example problem.

Based on the fitness evaluation in Section 4.3, the procedure of calculating C_{max} is detailed below.

(1) Initialization

According to Table 3, set $pyc[1] = ipc_1 = 3$, $pyc[2] = ipc_2 = 4$ and $pyt[1] = ipt_1 = 4$, $pyt[2] = ipt_2 = 3$.

Set $fyc[1] = fyc[2] = fyt[1] = fyt[2] = fqc[1] = fqc[2] = ryc[1] = ryc[2] = ryt[1] = ryt[2] = 0$

(2) $i = 1$: $\pi(1)(1) = 2$, $\pi(2)(1) = 2$, $\pi(3)(1) = 1$.

The arrival time of YC 1: $ayc[1] = ryc[1] + D_{pyc}[1]_{,B(2)}/1 = 0 + D_{33} = 0 + 0 = 0$
The finish time of YC 1: $fyc[1] = ayc[1] + hyc = 0 + 100 = 100$
The new position of YC 1: $pyc[1] = B(2) = 3$
The arrival time of YT 2: $ayt[2] = ryt[2] + D_{pyt}[2]_{,B(2)}/1 = 0 + D_{43} = 0 + 100 = 100$
The released time of YC 1: $ryc[1] = \max\{ayt[2], fyc[1]\} = \max\{100, 100\} = 100$
The finish time of YT 2: $fyt[2] = ryc[1] + D_{B(2)QC(2)}/1 = 100 + D_{31} = 100 + 150 = 250$
The new position of YT 2: $pyt[2] = QC(2) = 1$
The released time of YT 2: $ryt[2] = \max\{ fyt[2], fqc[QC(2)]\} = \max\{ fyt[2], fqc[1]\} = \max\{250, 0\} = 250$
The completion time of job 2 and the finish time of $QC(2)$:
$c_2 = fqc[QC(2)] = fqc[1] = ryt[2] + hqc = 250 + 60 = 310$

(3) $i = 2$: $\pi(1)(2) = 4$, $\pi(2)(2) = 1$, $\pi(3)(2) = 1$.

The arrival time of YC 1: $ayc[1] = ryc[1] + D_{pyc}[1]_{,B(4)}/1 = 100 + D_{34} = 100 + 100 = 200$
The finish time of YC 1: $fyc[1] = ayc[1] + hyc = 200 + 100 = 300$
The new position of YC 1: $pyc[1] = B(4) = 4$
The arrival time of YT 1: $ayt[1] = ryt[1] + D_{pyt}[1]_{,B(4)}/1 = 0 + D_{34} = 0 + 100 = 100$
The released time of YC 1: $ryc[1] = \max\{ayt[1], fyc[1]\} = \max\{100, 300\} = 300$
The finish time of YT 1: $fyt[1] = ryc[1] + D_{B(4)QC(4)}/1 = 300 + D_{41} = 300 + 250 = 550$
The new position of YT 1: $pyt[1] = QC(4) = 1$
The released time of YT 1: $ryt[1] = \max\{ fyt[1], fqc[QC(4)]\} = \max\{ fyt[1], fqc[1]\} = \max\{550, 310\} = 550$
The completion time of job 4 and the finish time of $QC(4)$:
$c_4 = fqc[QC(4)] = fqc[1] = ryt[1] + hqc = 550 + 60 = 610$

(4) $i = 3$: $\pi(1)(3) = 1$, $\pi(2)(3) = 1$, $\pi(3)(3) = 2$.

The arrival time of YC 2: $ayc[2] = ryc[2] + D_{pyc}[2]_{,B(1)}/1 = 0 + D_{33} = 0 + 0 = 0$
The finish time of YC 2: $fyc[2] = ayc[2] + hyc = 0 + 100 = 100$
The new position of YC 2: $pyc[2] = B(1) = 3$
The arrival time of YT 1: $ayt[1] = ryt[1] + D_{pyt}[1]_{,B(1)}/1 = 550 + D_{13} = 550 + 150 = 700$
The released time of YC 2: $ryc[2] = \max\{ayt[1], fyc[2]\} = \max\{700, 100\} = 700$
The finish time of YT 1: $fyt[1] = ryc[2] + D_{B(1)QC(1)}/1 = 700 + D_{32} = 700 + 200 = 900$
The new position of YT 1: $pyt[1] = QC(1) = 2$
The released time of YT 1: $ryt[1] = \max\{ fyt[1], fqc[QC(1)]\} = \max\{ fyt[1], fqc[2]\} = \max\{900, 0\} = 900$
The completion time of job 1 and the finish time of $QC(1)$:
$c_1 = fqc[QC(1)] = fqc[2] = ryt[1] + hqc = 900 + 60 = 960$

(5) $i = 4$: $\pi(1)(4) = 3$, $\pi(2)(4) = 2$, $\pi(3)(4) = 2$.

The arrival time of YC 2: $ayc[2] = ryc[2] + D_{pyc}[2]_{,B(3)}/1 = 700 + D_{34} = 700 + 100 = 800$
The finish time of YC 2: $fyc[2] = ayc[2] + hyc = 800 + 100 = 900$
The new position of YC 2: $pyc[2] = B(3) = 4$
The arrival time of YT 2: $ayt[2] = ryt[2] + D_{pyt}[2]_{,B(3)}/1 = 800 + D_{14} = 800 + 200 = 1000$
The released time of YC 2: $ryc[2] = \max\{ayt[2], fyc[2]\} = \max\{1000, 900\} = 1000$
The finish time of YT 2: $fyt[2] = ryc[2] + D_{B(3)QC(3)}/1 = 1000 + D_{42} = 1000 + 150 = 1150$
The new position of YT 2: $pyt[2] = QC(3) = 2$
The released time of YT 2: $ryt[2] = \max\{ fyt[2], fqc[QC(3)]\} = \max\{ fyt[2], fqc[2]\} = \max\{1150, 960\} = 1150$
The completion time of job 3 and the finish time of $QC(3)$:
$c_3 = fqc[QC(3)] = fqc[2] = ryt[2] + hqc = 1150 + 60 = 2210$

(6) Makespan $C_{max} = \max\{c_{\pi(1)(1)}, c_{\pi(1)(2)}, c_{\pi(1)(3)}, c_{\pi(1)(4)}\} = \max\{960, 310, 2210, 610\} = 2210$

References

1. Chen, L.; Bostel, N.; Dejax, P.; Cai, J.; Xi, L. A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. *Eur. J. Oper. Res.* **2007**, *181*, 40–58. [CrossRef]
2. Tang, W.; Du, S.; Hu, L.; Wang, B.; Zhu, Y. The effects of leadership in Clean Development Mechanism low-carbon operations. *Transport. Res. Part E Logist. Transp. Rev.* **2022**, *158*, 102575.
3. Du, S.; Wang, L.; Hu, L.; Zhu, Y. Platform-led green advertising: Promote the best or promote by performance. *Transport. Res. Part E Logist. Transp. Rev.* **2019**, *128*, 115–131. [CrossRef]
4. Huang, C.; Du, S.; Wang, B.; Tang, W. Accelerate or hinder it? Manufacturer transformation under competition and carbon emission trading. *Int. J. Prod. Res.* **2022**, 1–21. [CrossRef]
5. Carlo, H.J.; Vis, I.F.A.; Roodbergen, K.J. Transport operations in container terminals: Literature overview, trends, research directions and classification scheme. *Eur. J. Oper. Res.* **2014**, *236*, 1–13. [CrossRef]
6. Abdelmaguid, T.F.; Nassef, A.O.; Kamal, B.A.; Hassan, M.F. A hybrid GA/heuristic approach to the simultaneous scheduling of machines and automated guided vehicles. *Int. J. Prod. Res.* **2004**, *42*, 267–281. [CrossRef]
7. Deroussi, L.; Gourgand, M.; Tchernev, N. A simple metaheuristic approach to the simultaneous scheduling of machines and automated guided vehicles. *Int. J. Prod. Res.* **2008**, *46*, 2143–2164. [CrossRef]
8. Zheng, Y.; Xiao, Y.; Seo, Y. A tabu search algorithm for simultaneous machine/AGV scheduling problem. *Int. J. Prod. Res.* **2014**, *52*, 5748–5763. [CrossRef]

9. Zeng, Q.; Yang, Z. Integrating simulation and optimization to schedule loading operations in container terminals. *Comput. Oper. Res.* **2009**, *36*, 1935–1944. [CrossRef]
10. He, J.; Chang, D.; Mi, W.; Yan, W. A hybrid parallel genetic algorithm for yard crane scheduling. *Transport. Res. Part E Logist. Transp. Rev.* **2010**, *46*, 136–155. [CrossRef]
11. Ng, W.C.; Mak, K.L.; Zhang, Y.X. Scheduling trucks in container terminals using a genetic algorithm. *Eng. Optimiz.* **2007**, *39*, 33–47. [CrossRef]
12. Meisel, F.; Bierwirth, C. A unified approach for the evaluation of quay crane scheduling models and algorithms. *Comput. Oper. Res.* **2011**, *38*, 683–693. [CrossRef]
13. Liu, M.; Wang, S.; Chu, F.; Xu, Y. Some complexity results and an efficient algorithm for quay crane scheduling problem. *Discret. Math. Algorithms Appl.* **2016**, *8*, 1650058. [CrossRef]
14. AI-Dhaheri, N.; Jebali, A.; Diabat, A. A simulation-based Genetic Algorithm approach for the quay crane scheduling under uncertainty. *Simul. Model. Pract. Theory* **2016**, *66*, 122–138. [CrossRef]
15. He, J.; Zhang, W.; Huang, Y. A simulation optimization method for internal trucks sharing assignment among multiple container terminals. *Adv. Eng. Inform.* **2013**, *27*, 598–614. [CrossRef]
16. Ng, W.C.; Mak, K.L. Yard crane scheduling in port container terminals. *Appl. Math. Model.* **2005**, *29*, 263–276. [CrossRef]
17. Jung, S.H.; Kim, K.H. Loading scheduling for multiple quay cranes in port container terminals. *J. Intell. Manuf.* **2006**, *17*, 479–492. [CrossRef]
18. Wu, Y.; Li, W.; Petering, M.E.H.; Goh, M.; Souza, R. Scheduling multiple yard cranes with crane interference and safety distance requirement. *Transport. Sci.* **2015**, *49*, 990–1005. [CrossRef]
19. Bierwirth, C.; Meisel, F. A survey of berth allocation and quay crane scheduling problems in container terminals. *Eur. J. Oper. Res.* **2010**, *202*, 615–627. [CrossRef]
20. Rodrigues, F.; Agra, A. Berth allocation and quay crane assignment/scheduling problem under uncertainty: A survey. *Eur. J. Oper. Res.* **2022**, *303*, 501–524. [CrossRef]
21. Tang, M.; Ji, B.; Fang, X.; Yu, S.S. Discretization-Strategy-Based Solution for Berth Allocation and Quay Crane Assignment Problem. *J. Mar. Sci. Eng.* **2022**, *10*, 495. [CrossRef]
22. Cho, S.W.; Park, H.J.; Lee, C. An integrated method for berth allocation and quay crane assignment to allow for reassignment of vessels to other terminals. *Marit. Econ. Logist.* **2021**, *23*, 123–153. [CrossRef]
23. Kaveshgar, N.; Huynh, N. Integrated quay crane and yard truck scheduling for unloading inbound containers. *Int. J. Prod. Econ.* **2015**, *159*, 168–177. [CrossRef]
24. He, J. Berth allocation and quay crane assignment in a container terminal for the trade-off between time-saving and energy-saving. *Adv. Eng. Inform.* **2016**, *30*, 390–405. [CrossRef]
25. Luo, J.; Wu, Y.; Mendes, A.B. Modelling of integrated vehicle scheduling and container storage problems in unloading process at an automated container terminal. *Comput. Ind. Eng.* **2016**, *94*, 32–44. [CrossRef]
26. Lau, H.Y.K.; Zhao, Y. Integrated scheduling of handling equipment at automated container terminals. *Ann. Oper. Res.* **2008**, *159*, 373–394. [CrossRef]
27. Linn, R.; Liu, J.; Wan, Y.; Zhang, C.; Murty, K.G. Rubber tired gantry crane deployment for container yard operation. *Comput. Ind. Eng.* **2003**, *45*, 429–442. [CrossRef]
28. Murty, K.G.; Liu, J.; Wan, Y.; Linn, R. A decision support system for operations in a container terminal. *Decis. Support Syst.* **2005**, *39*, 309–332. [CrossRef]
29. Gen, M.; Cheng, R. *Genetic Algorithms and Engineering Optimization*; Wiley: New York, NY, USA, 1999.
30. Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*; Addison Wesley Longman: Boston, MA, USA, 1989.
31. Mitchell, M. *An Introduction to Genetic Algorithms*; MIT Press: Cambridge, MA, UK, 1999.
32. Cao, J.X.; Lee, D.-H.; Chen, J.H.; Shi, Q. The integrated yard truck and yard crane scheduling problem: Benders' decomposition-based methods. *Transport. Res. Part E Logist. Transp. Rev.* **2010**, *46*, 344–353. [CrossRef]