Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 548762, 14 pages
http://dx.doi.org/10.1155/2015/548762

Research Article

Simulation-Based Optimization for Storage Allocation Problem of Outbound Containers in Automated Container Terminals

Ning Zhao,[1] Mengjue Xia,[1] Chao Mi,[2] Zhicheng Bian,[3] and Jian Jin[3,4]

[1]*Logistics Engineering College, Shanghai Maritime University, 1550 Haigang Avenue, Shanghai 201306, China*
[2]*Container Supply Chain Technology Engineering Research Center, Shanghai Maritime University, 1550 Haigang Avenue, Shanghai 201306, China*
[3]*Logistics Research Center, Shanghai Maritime University, 1550 Haigang Avenue, Shanghai 201306, China*
[4]*Taicang Port SIPG ZHENGHE Container Terminal Co., Ltd., 8 North Circular Road, Taicang Port Area, Jiangsu 215438, China*

Correspondence should be addressed to Chao Mi; chaomi@shmtu.edu.cn

Received 11 June 2015; Revised 25 August 2015; Accepted 2 September 2015

Academic Editor: Alessandro Gasparetto

Storage allocation of outbound containers is a key factor of the performance of container handling system in automated container terminals. Improper storage plans of outbound containers make QC waiting inevitable; hence, the vessel handling time will be lengthened. A simulation-based optimization method is proposed in this paper for the storage allocation problem of outbound containers in automated container terminals (SAPOBA). A simulation model is built up by Timed-Colored-Petri-Net (TCPN), used to evaluate the QC waiting time of storage plans. Two optimization approaches, based on Particle Swarm Optimization (PSO) and Genetic Algorithm (GA), are proposed to form the complete simulation-based optimization method. Effectiveness of this method is verified by experiment, as the comparison of the two optimization approaches.

1. Introduction

Automated container terminals (ACT) are a category of container terminals with unmanned equipment, which detect their surroundings on their own, and execute the working actions in the right moment [1, 2]. In such terminal, containers are loaded onto or unloaded from vessels at quayside by Quay Cranes (QC) with double trolley, stacked into or retrieved out of blocks in storage yard by Automated Stacking Cranes (ASC), and transported between quayside and storage yard by Automatic Guided Vehicles (AGV). The QC, ASC, and AGV, all operated with no manpower, make up the container handling system of ACT, with a typical layout as shown in Figure 1. In this figure, QC are represented by two crossing rectangles, ASC are represented by rectangles with two edges popping out, and AGV are represented by small rectangles. Large rectangles are for blocks, strips of storage spaces laid perpendicular to the shoreline. Two ASC are bound to one block, one for the container stacking and retrieving at the seaside of the block and the other for those at the land side. Grey rectangles at both ends of the blocks are I/O points, in which containers could be stored temporarily.

Outbound containers are containers brought from inland area and are to be loaded onto vessels. Prior to vessel arrival, information of the outbound containers is sent to the terminal in EDI, including container numbers and stowage positions on the vessel. In the following days, these containers are brought in by container trucks, one after another in a random order. On the arrival of every outbound container, a block is allocated to it, and the landside ASC of that block is to stack the container into the block. Within vessel handling period, outbound containers are retrieved from block to the seaside I/O point by seaside ASC, picked up and transported to quayside by AGV, and loaded onto the vessel by QC, placed in their stowage positions according to EDI information.

Average QC efficiency, which means the average number of containers handled per hour per QC, is a main measurement for the performance of container handling system of ACT, while the upper bound of this measurement depends on QC capacity. Higher average QC efficiency leads to shorter container handling time; hence, vessels could depart earlier and finish their voyages in fewer days. The efficiency of a single QC reaches the maximum when it concentrates on

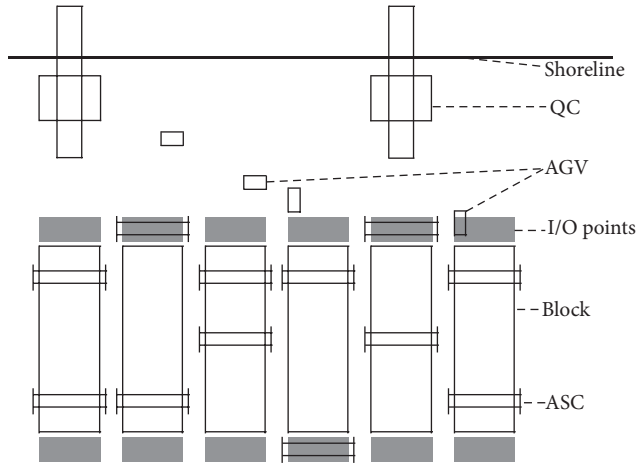

Figure 1: A schematic diagram of an ACT.

container handling, with no time wasted in other actions such as moving itself or waiting for an overdue AGV. Consequently, to optimize the performance of container handling system of ACT, a key factor is to prevent QC from waiting; hence, the average QC efficiency could be maximized.

Unreasonable distribution of outbound containers among blocks is a fundamental reason for QC waiting. For every outbound container stacked in some block, the QC and ASC to handle it could be predetermined: the QC is determined since the stowage position of the container on the vessel is already known, while there is only one ASC in a block retrieving containers to the seaside I/O point. In handling containers, a QC works continuously on condition that there is always some loaded AGV at its foot every time the trolley moves back empty to the quayside. Owing to the fact that capacity of ASC is commonly lower than that of QC, the condition for continuous QC working is kept by receiving containers from multiple ASC. However, this condition could be failed in case that outbound containers of one QC are all stacked in the same block, and QC waiting is unavoidable. Still, time of QC waiting could be even longer in case that the ASC of that block is retrieving outbound containers for another QC at the same time.

This paper presents a research on the *storage allocation problem for outbound containers in ACT* (SAPOBA), which outputs a storage plan allocating storage spaces to outbound containers before they arrive at the terminal. It is intended that outbound containers are stacked following the storage plan, resulting in a container distribution that leads to minimal QC waiting; hence, the performance of the container handling system could be optimized. The rest of the paper is organized as follows. Section 2 presents a review of the related works and an introduction to the methodology used in this paper. A Timed-Colored-Petri-Net (TCPN) model is proposed in Section 3 for simulation of container handling system in ACT, and two optimization approaches, Particle Swarm Optimization (PSO) and Genetic Algorithm (GA), are proposed in Section 4. Experiments are conducted in Section 5, to verify the effectiveness of the simulation-based optimization method and to compare the two optimization approaches. A conclusion is driven in the last section.

2. Related Works and Methodology

2.1. Related Works. SAPOBA has received hardly any attention for the past few years. In the field of ACT, existing research is mainly on problems of inbound containers, which pass through container terminals in a reversed direction. Some works are on block selection problem, in which every inbound container is assigned to a block to when they are unloaded from vessel [3, 4]. Some others are on container stacking problem, in which every inbound container is assigned to a stacking position in a block [5–8]. Different from outbound containers that are already stacked in some block, in container handling inbound containers could be placed in either block in the storage yard; hence, the candidate ASC to handle an inbound container is not unique. Therefore, these research results are not applicable to SAPOBA.

As a counterpart problem of SAPOBA in another kind of container terminals in which cranes and vehicles are operated by human labor, storage allocation problem of outbound containers in manual container terminals (SAPOBM) is a hotspot in recent years [9]. The idea of SAPOBM is quite similar to that of SAPOBA. With storage space properly assigned to outbound containers before they arrive at the terminal, cycle time of storage yard operations could be reduced. Howbeit in manual container terminals yard cranes are not bound to blocks, and multiple yard cranes are allowed to work in the same block; hence, the SAPOBM is a bit different from SAPOBA. Moreover, a general defect of current research on this problem is that no clear description could be given of the impact of outbound containers' storage plan on the performance of container handling system. Existing research propose mathematical models for their problem, in which objective functions are constructed for optimal storage plan. The aim of these functions could be minimizing total travel distance of vehicles [10–19], maximizing the balance level of workload among blocks [11, 14–17, 20–27], or maximizing the utilization of storage space [13, 15, 19, 24, 26]. Since none of

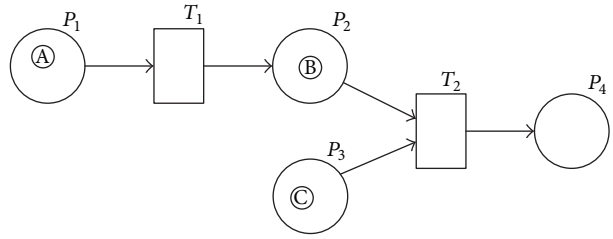

Figure 2: A simple Petri Net example.

these models is to minimize the QC waiting time, it could not be guaranteed that the optimal storage plans of these models will lead to best performances of container handling systems in the terminals.

It is not an easy work to describe the numerical relationship between QC waiting time and storage plan of outbound containers using mathematical models, while discrete event simulation (simulation hereinafter) is a suitable tool to achieve this purpose. In addition to the fact that QC waiting is a result of many factors related to QC, AGV, and ASC activities, AGV are dispatched in real-time and the AGV to transport each container is not determined before vessel handling starts. For the two reasons above, evaluation of storage plans by mathematical modeling in SAPOBA is quite a difficult task. In contrast, simulation is a process-based method which could go through the working process of the system to be modeled. Using this method, QC, AGV, and ASC in the container handling system are treated as individual objects, and the working process could be described as state changes in events. In this way, QC waiting could be expressed by time lags of some events, and AGV dispatching could be executed following dispatching rules implemented in the simulation model.

In consideration of the facts mentioned above, this paper is to solve the SAPOBA using simulation-based optimization method. This method is hybrid of simulation and optimization, in which simulation is used to evaluate candidates generated during the optimization procedure, while optimization is used to search for a best solution for the problem. The only difference between simulation-based optimization and pure optimization is the evaluation method of candidates. In simulation-based optimization, candidates are evaluated on the basis of simulation results, while in pure optimization, these evaluations are done by math expressions. Just recently, simulation-based optimization method has been successfully used to optimize ambulance fleet allocation and base station locations [28], to determine the best design and control factors of a paint shop production line in an automotive company [29], and to decide the best vehicle schedules for housekeeping in a container transshipment terminal [30]. To the extent of our knowledge, this method has not been used in ACT.

2.2. Methodology. Container handling systems in ACT are full of concurrency and asynchrony. Concurrency is a concept meaning that multiple independent processes in this system may go on simultaneously. In the container handling system consisting of multiple QC, ASC, and AGV, any two objects of them could operate concurrently, as long as they are not handling or transporting the same container. Asynchrony means that processes of different objects upon the same activity may occur in different times. For example, when transferring a container from an ASC to an AGV, the ASC just move to the I/O point and put the container down there, while the AGV just pick the container up once the container and the AGV itself are both at the I/O point. In this example, the ASC and AGV are for the same container (activity); however, they do not have to be at the I/O point at the same time.

Flowchart is a traditional formalism to describe the processes of a simulation model in concept, yet it is not suitable for simulation modeling of the container handling system in ACT. A flowchart describes a process by dividing it into subprocesses and connecting them with certain order. When a subprocess ends, the next subprocess according to order starts immediately. Thus, flowchart could only be used to describe sequential motions of one object as processes but not to express the concurrency and asynchrony among these processes and make up a model for the whole container handling system.

Known for the applicability to express asynchrony and concurrency, Petri Net is a modeling formalism for distributed parallel systems [31]. Basic Petri Nets consist of places, transitions, and directed arcs, signified by circles, rectangles, and arrows, respectively. An arc may run from a place to a transaction, or in a reversed direction form a transaction to a place. In the former case, the place is called the input place of the transaction, while, in the latter case, the place is called the output place. A number of tokens, signified by dots, may be located in some places, and move to some other places following regulations of the Petri Net. Suppose that there is a transaction and some input and output places connected to it. In case that there are enough tokens located in the input places of a transaction, this transaction fires, consuming a required number of tokens in the input places and creating new tokens in the output places. When used to build up a simulation model, places in a Petri Net indicate states of an object, while transactions indicate events of the system.

Petri Net is a formalism which is able to express the concurrency and asynchrony of a system in nature. Figure 2 is a simple example of Petri Net, consisting of five places (P_1, P_2, P_3, P_4, and P_5), two transactions (T_1 and T_2), and five arcs. Three token A, B, and C are now located in P_1, P_2, and P_3, respectively. In this figure, places P_1 and P_2 and transaction T_1 are in the same process, while places P_3 and P_4 are in two processes each. Tokens A and B are independent of each other

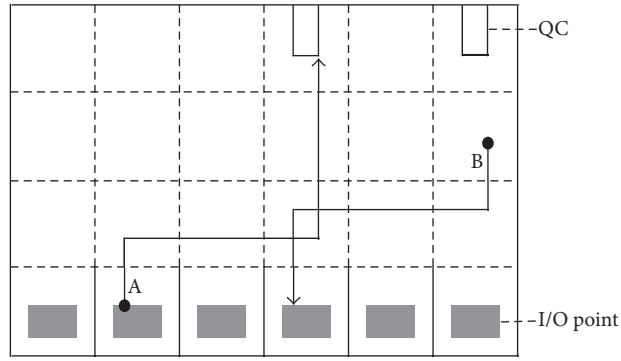

Figure 3: Illustration on grids and AGV routing.

in the process between P_1 and P_2; hence, the processes of the two tokens could go on concurrently. Tokens B and C are in two asynchronous processes, since it is not required that these processes should start or end in the same time. However, the following process of them (P_4) will not start till the two processes both end and the two tokens are in the input places of transaction T_2.

Colored Petri Net (CPN) [30] and Timed Petri Net (TPN) [32] are two extensions of Petri Net. In a CPN, tokens have properties and data values attached to them, called token color, after which distinction between tokens is made possible [33–35]. In a TPN, transactions are no longer finished instantaneously. On firing of a transaction, required tokens in the input places are consumed immediately, while tokens in the output places are generated some time later [36, 37]. In simulating the container handling system in ACT, token colors could be used to distinguish different QC, ASC, AGV, and containers, while transaction times could be used to indicate the time length of equipment motions.

The possible contributions of this paper are listed as follows:

(1) A simulation-based optimization method is proposed to solve the SAPOBA, which outputs a storage plan of outbound containers that leads to a best performance of the container handling system in ACT.

(2) TCPN is used to build up the simulation model, which establishes directly the relationship between storage plans of outbound containers and QC waiting time of container handling system.

(3) Two optimization methods, PSO and GA, are compared when used in a simulation-based optimization for optimal solution.

3. TCPN Model for Simulation

3.1. Model Assumptions. Assumptions of the model are listed below:

(1) Only outbound containers are considered in this model.

(2) Only 40 ft container are considered in this model. 20 ft containers could be converted into 40 ft ones in pairs.

(3) The capacity of I/O points, as buffers between ASC and AGV, is fixed and is the same. No such buffer exists between QC and AGV.

(4) Moving range of AGV could be divided into two-dimensional grids of equal square zones. Positions of AGV could be represented by coordinate of the grid which they are in, as positions of QC and I/O points.

(5) Capacity of each grid is large enough to hold all the AGV at the same time, and traffic congestions are not considered in this model. All AGV move in a constant speed, and they always spend the same time in crossing a current grid.

(6) AGV routes are decided step by step following rules. When travelling to a destination grid, at first an AGV tries to run into a next grid nearer the destination grid in a direction parallel to the shoreline. In case that no such grid exists, the AGV runs into a next grid nearer to the destination grid in a direction perpendicular to the shoreline.

Assumptions (4) to (6) could be illustrated in Figure 3, which shows a two-dimensional grid of 6×4 square zones. AGV could run into a neighboring grid crossing a dash line, while full lines are unbridgeable. There are 6 I/O points and 2 QC in these grids. Two AGV A and B are traveling to their destinations. Routes of the two AGV are shown as directed fold lines, intersecting at the same grid. Following the routes, the two AGV will run into the same grid simultaneously; however, their travelling times will never be lengthened:

(7) Outbound containers are allowed to be retrieved in batches. Containers in the same batch get the same earliest time for retrieval, before which they are not allowed to be retrieved.

(8) Travelling speeds of ASC and AGV are treated as constants in this model, while QC are assumed to stay in a fixed grid during handling process.

(9) An AGV is always bound to some ASC. In case that an AGV gets no task at the moment, it runs back to the ASC it is bound to and waits there for a next task.

(10) Storage spaces are allocated in slots (spaces in the same bay and same row of a block), and outbound

containers are divided into stacks (a group of containers that could be placed in one slot). On this basis, storage plans are treated as one-to-one matching of slots and stacks.

3.2. Constraints of Storage Plans. Storage plan is an allocation of storage spaces to containers, a one-to-one matching plan in nature. Definition and constraints of these plans could be expressed by math formulas. Notations used in the formulas are listed below:

I: total number of stacks, indexed by i or i', $1 \le i,\ i' \le I$.

J: total number of blocks in the storage yard, indexed by j, $1 \le j \le J$.

K: total number of bays in a block, indexed by k, $1 \le k \le K$. The smaller the bay number is, the closer this bay is to the shoreline

C_{jk}: number of free slots in the kth bay of block j.

x_{ijk}: if stack i is allocated to the slot in the kth bay of block j then $x_{ijk} = 1$; otherwise, $x_{ijk} = 0$.

A storage plan could be described by a set of variables x_{ijk}:

$$\left\{x_{ijk} \mid i \in \{1, 2, \ldots, I\},\ j \in \{1, 2, \ldots, J\},\ k \in \{1, 2, \ldots, K\}\right\}. \tag{1}$$

The constraints are shown in the equations below

$$\sum_{i} x_{ijk} \le C_{jk}, \quad \forall j, k. \tag{2}$$

Constraint (3) means that the total number of stacks allocated to the same bay in the same block should never exceeds the capacity of that bay:

$$\sum_{j,k} x_{ijk} = 1, \quad \forall i. \tag{3}$$

Equation (4) means that each stack could be allocated only once. Outbound containers are to be stacked on their arrivals in locations as determined in the storage plan.

3.3. Overall Model. Notations used in the simulation model are listed below:

Q: total number of QC.

A: total number of ASC.

C: total number of containers.

V: total number of AGV.

B: capacity of an I/O point.

G: total number of zones in the moving range of AGV.

P: total number of places, indexed by p. In this model $P = 20$.

T: total number of transactions, indexed by t. In this model, $T = 13$.

H: total number of containers to be handled.

d_p: a token in place p.

$d_{p,t}$: a token that is fired from place p to transaction t.

$d'_{t,p}$: a token that is fired out of transaction t into place p.

CS: color set of tokens, an 8-tuple of integers as $\langle cid, bid, aid, qid, aim, pos, az, qz\rangle$.

cid: attribute of tokens indicating the container number.

bid: attribute of tokens indicating the batch number.

aid: attribute of tokens indicating the ASC number.

qid: attribute of tokens indicating the QC number.

aim: attribute of tokens indicating the destination zone number of an AGV.

pos: attribute of tokens indicating the current zone number of an AGV.

az: attribute of tokens indicating the zone number of I/O point of an ASC to which an AGV is bound. An AGV moves back to the zone *az* when it is free.

qz: attribute of tokens indicating the zone number of QC joint point.

$L(z_1, z_2)$: travel distance between two zones, written as z_1 and z_2.

$M(p, \exp)$: number of tokens in places p as defined by expressions exp in the bracket. If there is no expression in the brackets, this natation means the total number of tokens in places p.

$N(z_1, z_2)$: the next zone of z_1 when running to zone z_2.

δ: average QC waiting time, as output of the model.

CN_q: total number of containers handled by QC q, indexed by n.

ts_{qn}: start time of the nth handling of QC q, according to simulation.

te_{qn}: end time of the nth handling of QC q, according to simulation.

The overall TCPN model is shown in Figure 4. Circles in this figure are for places, in which dashed circles indicate boundaries between different kinds of equipment. As for places with limited capacity, the maximal number of tokens allowed is marked at the upper left of the circle. Meanings of places in Figure 1 are listed in Table 1. Squares and rectangles are for transactions; the former fire instantaneously while the latter fire with some time. Square brackets are for firing conditions of transactions. In case that there is a square bracket over or under a square/rectangle, the transaction will not fire till the conditions are met. Arrows are for arcs, each with the number of tokens to or from a transaction marked. Marks of arrows could be divided into two parts. For arrows from a place to a transaction, the first part of the mark indicates number of tokens consumed by the transaction, and

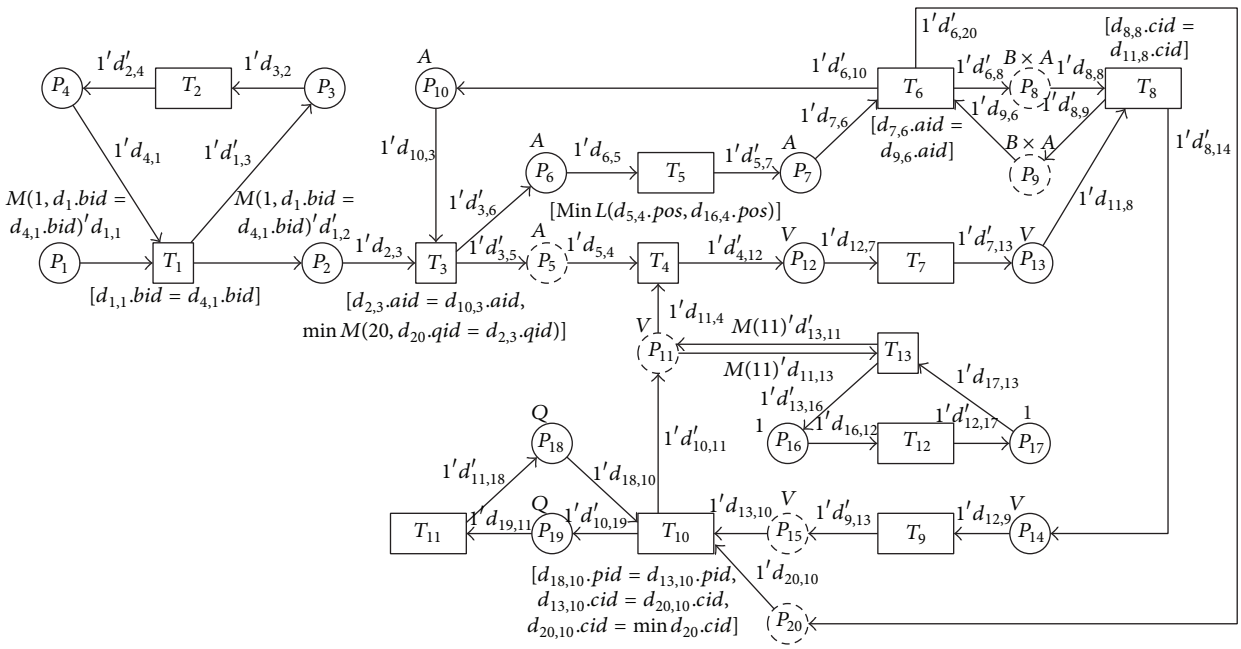

Figure 4: TCPN model for a handling system in ACT.

the second part is written as "$d_{p,t}$," with p for the origin place and t for the destination transaction, indicating every token going through the arc. The two parts are put together to be a whole mark, with a back single quotation set between them. Marks of arcs in an opposed direction are formed in a similar way, while in these marks the tokens are written as "$d'_{t,p}$" for distinction.

Meanings of places in Figure 1 are listed in Table 1, while the transactions and arcs are to be explained in partial models.

In the beginning, some colored tokens are scattered initially in some places in the TCPN model, which is called the initial marking of the model, as is shown in Table 2. Tokens located in P_1 are for containers to be handled. Each container gets a unique container number, and the batch numbers are known as well. In addition, the ASC and QC to handle the containers are known as well (*.aid* and *.qid*), as the zone number of QC joint point (*.qz*). The token in P_2 is used to allow container retrievals in batches, and no color is assigned to it. Tokens in P_{10} are for ASC; hence, the ASC number of each token (*.aid*) is known and unique, as the zone number of I/O points (*.az*). Tokens in P_9 are used to indicate the residual capacity of I/O points; hence, tokens in this place all get an ASC number (*.aid*), and the amount of tokens that have the same ASC number is fixed. Tokens in P_{18} are for the QC; hence, each of these tokens gets a unique QC number (*.qid*). Tokens in P_{11} are for the AGV, which are always located in some zone (*.pos*) and bound to some ASC (*.az*). There is only one token in P_{17} with no color, used to update the positions of AGV periodically. The TCPN model ends with a similar marking as in the beginning, yet no container is left in place P_1.

When simulation ends, the total QC waiting time could be calculated as sum of time intervals between the time a QC finishes a current container handling and the time the QC starts a next container handling. As a result, the average QC waiting time, as the output of the simulation runs and the optimization objective of the SAPOBA, could be calculated following (1) as follows:

$$\text{Min } \delta = \text{Min } \frac{1}{Q} \sum_{q} \left(\sum_{n=2}^{CN_q} \left(ts_{qn} - te_{q(n-1)} \right) \right). \tag{4}$$

3.4. Partial Models. The overall TCPN model could be divided into three parts, with those dashed circles in Figure 4 as boundaries, corresponding to ASC, AGV, and QC, respectively. Partial model of ASC is shown in Figure 5. Notice that the position of P_{20} is adjusted.

There are 5 transactions in this partial model. Transition T_1 is to allow a new batch of outbound containers to be retrieved by ASC, and transaction T_2 is the cool-down of this allowance. In case that the cool-down ends and a token d is placed in P_4, all the tokens (containers) in P_1 with the same batch number (*.bid*) as token d are fired ($d_{1,1}$), as token d itself ($d_{4,1}$). At the same time, one new tokens ($d'_{1,2}$) are created immediately into P_2 for every $d_{1,1}$, and a new token ($d'_{1,3}$) is also created to P_3. On this creation, transactions T_2 are fired, creating a new token ($d'_{2,4}$) into P_4 after a fixed interval. With the help of T_2, transaction T_1 brings periodically batches of containers that are allowed to be retrieved by ASC, following the handling schedule in a real ACT.

Transaction T_3 is fired when an ASC is ready for a next container (P_{10} not empty) and there is some container allowed to be retrieved by it already (token exists in P_2 with the same *.aid*). Considering that there may be multiple tokens in P_2 and they could go to different QC, an ASC always retrieve a container to the QC with minimum inventory,

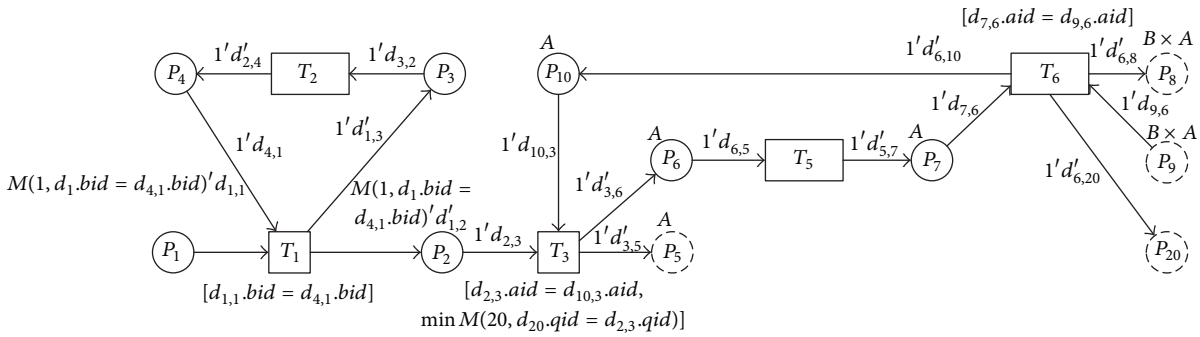

Figure 5: Partial model of ASC.

Table 1: Meanings of places.

Place	Practical meaning
P_1	Outbound containers in blocks not allowed to be retrieved
P_2	Outbound containers in blocks allowed to be retrieved
P_3	A batch of containers that has been allowed to be retrieved
P_4	Time to allow a new batch of containers to be retrieved
P_5	Information of containers that ASCs have decided to retrieve but no AGV is allocated
P_6	Containers that ASCs start to retrieve
P_7	Loaded ASCs at I/O points
P_8	Containers stacked at I/O points
P_9	Residual capacities of I/O points
P_{10}	ASCs ready for a next container
P_{11}	Unloaded free AGVs
P_{12}	Unloaded AGVs allocated to some container
P_{13}	Unloaded AGVs allocated to some container at I/O point
P_{14}	Loaded AGVs at I/O point
P_{15}	Loaded AGVs at the foot of QC
P_{16}	Ready to update positions of AGVs
P_{17}	Position of AGVs updated
P_{18}	Unloaded QCs at landside
P_{19}	Loaded QCs at landside
P_{20}	Containers that have been placed into I/O point but not handled by QCs

as shown in the firing condition. QC inventory means the number of containers which have been placed in the I/O point but not handled by the QC, equal to the number of tokens to the QC in P_{20}. This strategy is also used in [38], in which it is believed that this strategy helps in preventing QC from waiting for AGV, leading to a good productivity of the terminal. On firing of this transaction, a token ($d'_{3,5}$) is created into P_5 immediately, as an AGV task to be allocated.

Table 2: Initial marking of the TCPN.

Place	Number of tokens	*.cid*	*.bid*	*.aid*	*.qid*	*.aim*	*.pos*	*.az*	*.qz*
P_1	C	√	√	√	√	—	—	—	√
P_4	1	—	—	—	—	—	—	—	—
P_{10}	A	—	—	√	—	—	—	√	—
P_9	$B \times A$	—	—	√	—	—	—	—	—
P_{18}	Q	—	—	—	√	—	—	—	—
P_{11}	V	—	—	—	—	—	√	√	—
P_{17}	1	—	—	—	—	—	—	—	—

Moreover, another token ($d'_{3,6}$) is also created into P_6 at the same time, as the next container to be handled.

Transaction T_5 is for the cycle time of ASC. When deciding to retrieve a container (P_6 not empty), an ASC moves to its stacking location, pick it up, and carry it to the I/O point. Time length of this process could be calculated, since the stacking position of container and the moving speed of ASC are both known. On the arrival of ASC to the I/O point, a new token is created to P_7. Notice that since stacking positions of containers are all different, firing time of T_5 is not fixed.

Transaction T_6 is to add a new container to the I/O point of some ASC. In case that an ASC is loaded at the I/O point (P_7 not empty) and the I/O point is not filled (at least one token exists in P_8 with the same ASC number *.aid*), this transaction consumes a token in P_7 and P_8, respectively. After a time, a token is created to P_8 as a container stacked in I/O point, and another token is also created to P_{20} as QC inventory. Firing time of T_6 is fixed, which equals the time an ASC spends in putting a container down to the I/O point.

Color relations of tokens to and from the same transaction in this partial model are listed in Table 3. Notice that tokens from or to the same place always get the same color, while the color of tokens created in a firing is not always inherited from the consumed tokens. Attributes not involved in this partial model are excluded.

Partial model of AGV is shown in Figure 6. Positions of P_8, P_9, P_{13}, and T_8 are adjusted.

There are in total 6 transactions in the partial model of AGV. Transaction T_4 is to allocate a container task to an AGV with no task. In case that there is some tasks to be

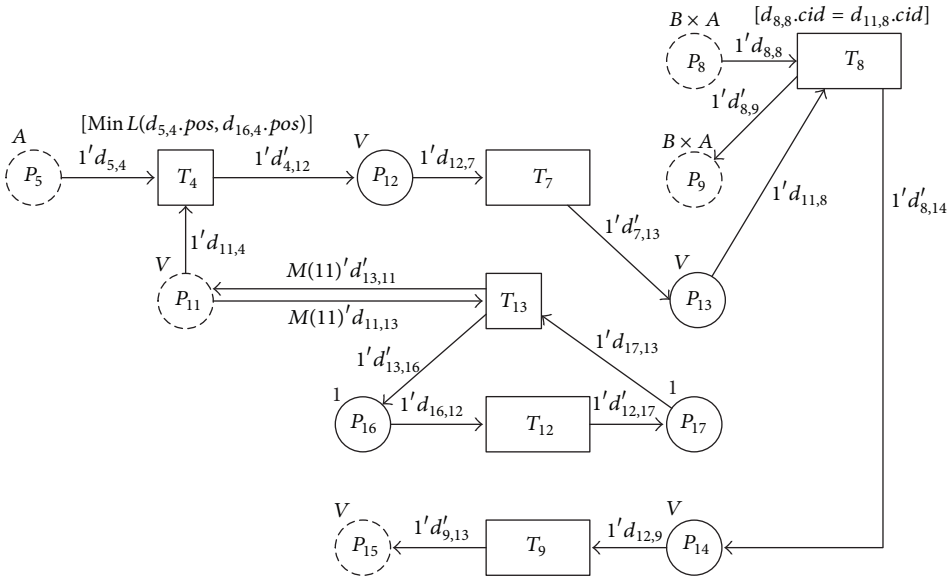

Figure 6: Partial model of AGV.

Table 3: Color relations of tokens in partial model of ASCs.

Token	.cid	.bid	.aid	.qid	.az	.qz
$d'_{1,2}$	$d_{1,1}.cid$	—	$d_{1,1}.aid$	$d_{1,1}.qid$	—	$d_{1,1}.qz$
$d'_{1,3}$	—	$d_{4,1}.bid + 1$	—	—	—	—
$d'_{2,4}$	—	$d_{3,2}.bid$	—	—	—	—
$d'_{3,5}$	$d_{2,3}.cid$	—	$d_{2,3}.aid$	$d_{2,3}.qid$	$d_{2,3}.az$	$d_{2,3}.qz$
$d'_{3,6}$	$d_{2,3}.cid$	—	$d_{2,3}.aid$	$d_{2,3}.qid$	—	—
$d'_{5,7}$	$d_{6,5}.cid$	—	$d_{6,5}.aid$	$d_{6,5}.qid$	—	—
$d'_{6,8}$	$d_{7,6}.cid$	—	$d_{7,6}.aid$	—	—	—
$d'_{6,10}$	—	—	$d_{7,6}.aid$	—	—	—
$d'_{6,20}$	$d_{7,6}.cid$	—	$d_{7,6}.aid$	$d_{7,6}.qid$	—	—

allocated (P_5 not empty) and some AGV are with no tasks (P_{11} not empty), T_4 fires to allocate a task to a nearest AGV, by consuming corresponding tokens in P_5 and P_{11} and creating a new token into P_{12} at once.

Transaction T_7 is used to simulate the travel time of an unloaded AGV with some task to the I/O point of the container. Firing time of the transaction equals travel time of AGV, which could be determined since the current zone and destination zone are both known for the AGV.

Transaction T_8 is for the process in which an AGV picks up a container at I/O point. In case that an AGV has reached the I/O point (P_{13}) and the container is placed there as well (P_8), this transaction fires, consuming the tokens in P_8 and P_{13}, and creating one new token in P_9 and P_{14} after a fixed time interval, respectively; hence, the AGV receives the container (P_{14}), and the number of containers that could be placed in the I/O point is added by 1 (P_8).

Transaction T_9 is used to simulate the travel time of a loaded AGV to the foot of the destination QC. Firing time of this transaction is calculated as in transaction T_7, after which a new token is created into P_{15}.

Transactions T_{12} and T_{13} are used to update the locations of the AGV with no tasks periodically. They work in a similar way as Transactions T_1 and T_2. For all the AGV with no task, they are moved in a fixed interval to a neighboring zone nearer to the ASC which they are bound to, if possible.

Color relations of tokens in this partial model are listed in Table 4. Attributes not involved in this partial model are excluded.

Partial model of QC is shown in Figure 7. Positions of P_{11} are adjusted.

There are only 2 transactions in this partial model. Transaction T_{10} is for the process that a QC picks up a container from an AGV. In case that a QC is ready for a next container loading (P_{18} not empty), and a loaded AGV has reached the foot of that QC (token with the same QC number in P_{15}), this transaction fires, consuming one token in P_{18}, P_{15}, and P_{20}, following firing conditions as shown in Figure 7. In case that there are multiple loaded AGV at the foot of the same QC, the AGV carrying the container with a minimal container number will be consumed. Firing time of transaction T_{10} is fixed. When the firing ends, a new token is created into P_{11} as an AGV with no task allocated to it, and another new token is created into P_{19} indicating that the QC has received the container.

Transaction T_{11} is used to simulate the time a QC trolley spends in putting a container onto a vessel and turn back empty to the quayside. Firing time of T_{11} is also fixed. When the firing ends, a new token is created to P_{18}; hence, the QC is ready for a next load.

Color relations of tokens in this partial model are listed in Table 5. Also, attributes not involved in this partial model are excluded.

4. Optimization Approaches

The TCPN model proposed in the last section could be used as fitness functions of optimization methods, evaluating

Table 4: Color relations of tokens in partial model of AGVs.

Token	.cid	.aid	.qid	.aim	.pos	.az	.qz
$d'_{4,12}$	$d_{5,4}.cid$	$d_{5,4}.aid$	$d_{5,4}.qid$	$d_{5,4}.az$	$d_{11,4}.pos$	$d_{11,4}.az$	$d_{5,4}.qz$
$d'_{7,13}$	$d_{12,7}.cid$	$d_{12,7}.aid$	$d_{12,7}.qid$	$d_{12,7}.aim$	$d_{12,7}.aim$	$d_{12,7}.az$	$d_{12,7}.qz$
$d'_{8,9}$	—	$d_{8,8}.aid$	—	—	—	—	—
$d'_{8,14}$	$d_{11,8}.cid$	$d_{11,8}.aid$	$d_{11,8}.qid$	$d_{11,8}.qz$	$d_{8,8}.pos$	$d_{11,8}.az$	—
$d'_{9,13}$	$d_{12,9}.cid$	—	$d_{12,9}.qid$	$d_{12,9}.aim$	$d_{12,9}.aim$	$d_{12,9}.az$	$d_{12,9}.qz$
$d'_{13,11}$	—	—	—	—	$N(d_{11,13}.pos, d_{11,13}.az)$	$d_{11,13}.az$	—
$d'_{13,16}$	—	—	—	—	—	—	—
$d'_{12,17}$	—	—	—	—	—	—	—

Table 5: Color relations of tokens in partial model of QCs.

Token	.cid	.aid	.qid	.aim	.pos	.az	.qz
$d'_{10,11}$	—	—	—	$d_{13,10}.az$	$d_{13,10}.qz$	—	—
$d'_{10,19}$	—	—	$d_{18,10}.qid$	—	—	—	—
$d'_{11,18}$			$d_{19,11}.qid$				

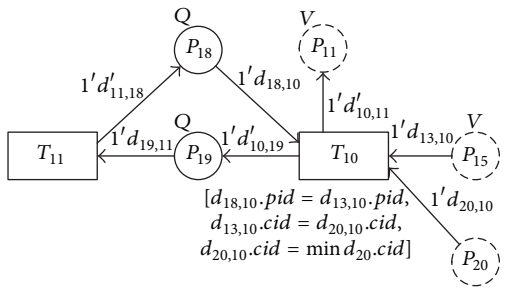

Figure 7: Partial model of QC.

the influences of candidate storage plans on the performance of container handling system. In this section, two intelligent optimization approaches, based on Particle Swarm Optimization (PSO) and Genetic Algorithm (GA), are proposed to complete the simulation-based optimization method.

4.1. Encoding. Intelligent optimization is a category of methods used to search for a solution of a problem as good as possible. These methods go through new candidate solutions continuously in a self-defined direction, till an end condition is fulfilled, and the best candidate solution found is output as the result. Encoding is a first step of intelligent optimization. Using some set of regulations, solutions of a problem could be expressed in a form convenient for optimization operations.

Storage plans are encoded as arrays of real numbers, as candidate solutions of the SAPOBA. Each number in the array is for a stack to be allocated, while the value of this number indicates a slot for the stack. For every real number in a code array, the integer part is for the number of block that the stack is to be allocated in, while the decimal part is for the priority of this stack to get a slot in the block. Stacks with a smaller decimal part tends to get a slot nearer to the ASC; hence, the ASC cycle time of containers in this stack could be shorter. In case that there is no slot left in a block, stacks to be allocated in this block will be reallocated to some other block, and the number of these stacks will also be changed.

Table 6: A coding example.

Stack number	1	2	3	4	5	6	7
Value	1.33	1.24	1.08	1.66	1.59	2.27	2.95

Table 7: Storage spaces available for the code.

Block number	Bay number	Number of slots available
1	1	2
1	2	2
2	1	2
2	2	2

Table 6 is a coding example of 7 stacks. Storage spaces are scattered in 2 blocks, as shown in Table 7. In this example, stacks 2 and 3 are allocated to the 1st bay in block 1, stacks 1 and 5 are allocated to the 2nd bay in block 1, and stacks 6 and 7 are allocated to the 1st bay in block 2. There are no slot left for stack 4; hence, it is allocated to the 2nd bay in block 2, and the value of this stack is to be changed into 2.66 accordingly.

4.2. PSO Approach. Particle Swarm Optimization works with a fixed number of particles (called swarm) iteratively. A particle moves to a possible solution in each iteration, while in a next iteration this particle moves to another. The best solution of the swarm and best solutions of particles are kept over iterations, which are used to guide the searching direction of particles. Notations used in the PSO approach are listed below:

G: scale of the swarm, total number of particles, indexed by g, $1 \le g \le G$.

R: maximum number of iterations, indexed by r, $1 \le r \le M$.

S: length of a particle, equal to the number of stacks to be allocated.

e_{rgs}: the sth element of gth particle in generation r, a real number.

p_{rg}: the gth particle in generation r, an array of I elements.

v_{rg}: speed of the gth particle in generation r, an array of I real numbers.

pb_g: best historical solution of particle g.

sb: best historical solution of the swarm.

δ_{rg}: fitness value of candidate solution of particle g in generation r.

δ'_g: fitness value of best historical candidate solution of particle g.

δ'': fitness value of best historical candidate solution of the swarm.

δ^E: maximal fitness value of a satisfactory candidate solution.

N^E: maximum allowable number of successive iterations in which best solution is not changed.

Since candidate solutions are encoded as one-dimensional arrays of real numbers, a particle g in iteration r could be defined as in the following equation:

$$p_{rg} = \left\{ e_{rgs} \mid 1 \le s \le S,\ s \in N^*,\ 1 < e_{rgs} < A + 1,\ e_{rgi} \in R \right\}. \tag{5}$$

Notation A is for the total number of ASC in ACT, which is already defined in Section 3.2. In the beginning of optimization, particles and their speeds are initialized in random. In each iteration, the current candidate solution of each particle is first evaluated by simulation output δ; then the best solutions pb_g and sb are updated in case that better solutions are found in the particles, as shown in the following two equations:

$$pb_g = \begin{cases} p_{rg}, & r = 1 \text{ or } \delta_{rg} < \delta'_g, \\ pb_g, & \text{otherwise}, \end{cases}$$
$$sb = \begin{cases} p_{rg}, & r = 1 \text{ or } \left(\delta_{rg} < \delta'',\ \delta_{rg} = \operatorname{Min}_{g'} \left(\delta_{rg'} \right) \right) \\ sb, & \text{otherwise}. \end{cases} \tag{6}$$

Iterations are used to generate new the candidate solution of all the particles. This operation is conducted following equations below:

$$v_{rg} = \omega_v v_{rg} + \omega_p \left(pb_g - p_{rg} \right) + \omega_s \left(sb - p_{rg} \right), \tag{7}$$

$$p_{rg} = p_{rg} + v_{rg}. \tag{8}$$

Notations ω_v, ω_p, and ω_s in (7) are all coefficients, valued between 0 and 1. Three end conditions are set in the approach, as listed below. The PSO approach ends if that one condition is satisfied in an iteration, and the best candidate solution found is output as a result of PSO approach:

(1) Fitness value of the best candidate solution found is no worse than δ^E.

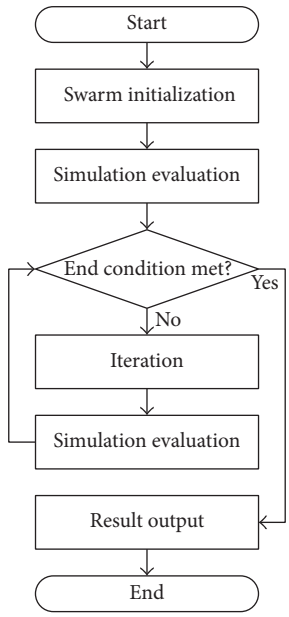

Figure 8: Flowchart of simulation-based optimization using PSO approach.

(2) Current iteration number r reaches the maximum iteration number R.

(3) Best historical solutions has not been changed over the past N^E iterations.

The procedure of simulation-based optimization method using PSO approach is shown in Figure 8.

4.3. GA Approach. In Genetic Algorithm, candidate solutions are treated as chromosomes, of which the basic units are treated as genes, just as the real numbers in the code of a storage plan. In the beginning, a fixed number of chromosomes are created and treated as the initial group. New chromosomes are created from current chromosomes by crossover and variation, after which a number of chromosomes are selected into a new group with the size unchanged in a next generation. Due to the fact that definitions of chromosomes in GA approach and candidate solutions in PSO approach are the same, notations for swarms and particles are used here for groups and chromosomes.

Crossover is executed between two chromosomes as parents. With the values of some genes in same positions changed in pairs, two new chromosomes are created from their parents. For example, chromosomes 1 and 2 in generation 1 crossover at the 5th gene to form two new chromosomes 3 and 4. This crossover could be expressed by formulas as follows:

$$e_{1,3,s} = \begin{cases} e_{1,1,s}, & s \ne 5, \\ r_3, & \text{otherwise}, \end{cases}$$
$$e_{1,4,s} \begin{cases} e_{1,2,s}, & i \ne 5, \\ r_4, & \text{otherwise}, \end{cases} \tag{9}$$

Table 8: Parameters of equipment[1].

Parameter	Value	Parameter	Value
QC efficiency	40 moves/h	AGV moving speed	6 m/s
ASC hoisting speed (unloaded)	1 m/s	ASC hoisting speed (loaded)	0.5 m/s
ASC moving speed	3.5 m/s		
Time of an AGV picking up a container at I/O point			10 s

[1]These parameters have been used in the simulation program for the Next Generation Container Port Challenge (http://www.ise.nus.edu.sg/news/award-mpa-winners.html). Three authors of this paper participated in the challenge, and their team is the final winner.

$$r_3 + r_4 = e_{1,3,5} + e_{1,4,5},$$
$$1 < r_3, r_4 < A + 1, \quad r_3, r_4 \in R. \tag{10}$$

Commas in the formulas above are used to separate digits of different notations in subscripts. Real numbers r_3 and r_4 are randomly generated, under constraints as in formula (10).

Variations are executed with single chromosomes. In a variation, values of some genes in a chromosome will be replaced by a random number; hence, a new chromosome is created.

Selection is executed to the whole group of chromosomes of which the fitness values are already known. In this paper, the best chromosome will always be selected, while the rest are selected following roulette strategy. The better the fitness value of a chromosome, the larger the chance it is selected into a next generation.

End conditions in the GA approach is same as in the PSO approach. The procedure of simulation-based optimization method using GA approach is shown in Figure 9.

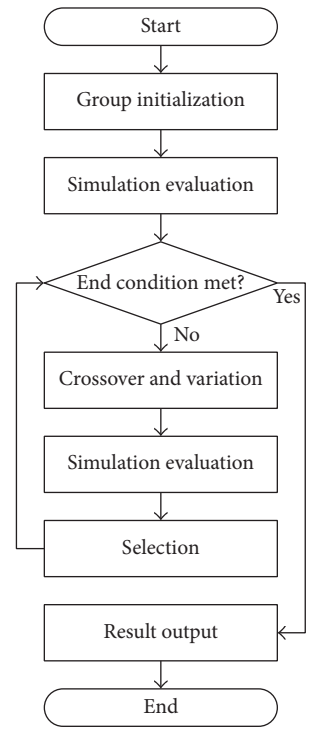

Figure 9: Flowchart of simulation-based optimization using GA approach.

5. Experiment

The simulation-based optimization method is applied for random examples in two scenarios. There are 4 QC, 10 blocks, and 10 ASC in the small scenario and 6 QC, 15 blocks, and 15 ASC in the large scenario. Three AGV are bound to the same ASC in both scenarios. A 15×4 grid of square zones, with side-lengths of 36 meters (with reference to the satellite photo of Maasvlakte Container terminal in Rotterdam), is established to hold the QC, AGV, and I/O points of ASC, as shown in Figure 4. QC 5 and QC 6 in this figure only exist in scenario 2, as ASC 11–15. Containers are stored in blocks in the same size of 20 bays, 6 rows, and 5 tiers each, yet the blocks are not included in Figure 10. ASC are all dedicated to one block. Containers in the same block are always retrieved by the same ASC, while containers in different blocks are not.

Equipment parameters used in the experiment are listed in Table 8.

Instances of outbound containers are generated as follows. In each scenario, there are 40 containers (divided into 8 stacks) to be handled by each QC per hour in the coming 12 hours. In case that the storage plan is properly made, QC waiting could be eliminated entirely, and the storage plan is considered satisfactory. Consequently, there are 384 stacks of containers to be allocated into 10 blocks in the small scenario and 576 stacks to be allocated into 15 blocks in the large scenario, as are the code lengths of them.

The simulation program of the TCPN model is realized in Tecnomatix Plant Simulation 11, a professional software for discrete event simulations. The PSO and GA approaches are realized in the same environment as well. When evaluating a candidate solution, the code of the solution is first converted into a corresponding storage plan. On this basis, simulation runs and QC waiting times are calculated as the fitness value of the candidate solution. For the PSO approach, parameters ω_v, ω_p, ω_s, R, and N^E in the algorithm are assigned to 0.9, 0.8, 0.8, 100, and 20, respectively. For the GA approach, scale of the group is set the same as the scale of swarm in PSO approach. Crossover is executed 5 times every generation, and the probability of a gene to be selected to cross is 0.8. The probability of a chromosome to be selected and varied is 0.15, as the probability of a gene.

Experiments are conducted separately in two scenarios. For each scenario, 20 instances are generated in random. Each instance is solved 5 times with simulation-based optimization method using PSO approach first and then solved another 5 times using GA approach. The best fitness values of the 100

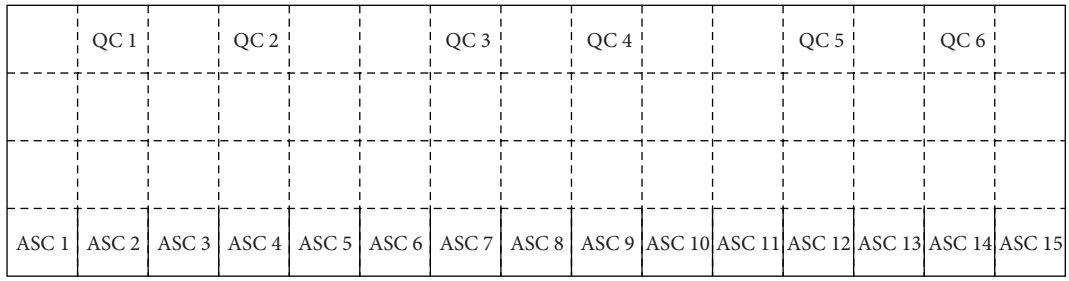

Figure 10: Layout of two scenarios.

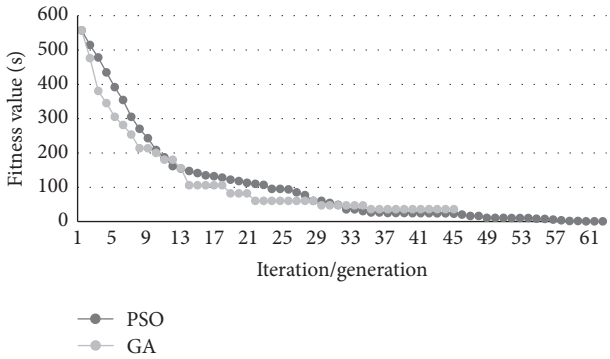

Figure 11: Average best fitness values in the small scenario.

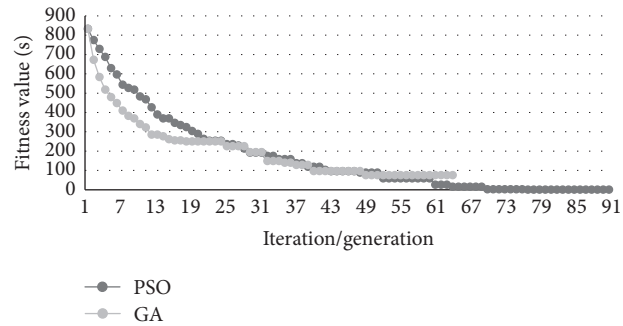

Figure 12: Average best fitness values in the large scenario.

Table 9: Rest results in both scenarios.

Scenario	T^A (s)	Approach	N^0
Small	27.3	PSO	89
		GA	73
Large	38.6	PSO	82
		GA	69

optimization runs in every iteration/generation are recorded, and the mean values of average best fitness values in every iteration/generation (if not ended) of the two approaches are shown in Figures 11 and 12. Moreover, attentions are paid to the number of satisfactory solutions found in the 100 optimization runs using PSO and GA approaches (noted as N^0) and the average time span of simulation runs (T^A).

Figure 11 is the average best fitness values in the small scenario, in which the X-axis is for the iteration/generation numbers and the Y-axis is for the fitness values (in seconds). The figure tells that GA approach outperforms PSO approach in the earlier stage of optimization for fast convergence, while in the later stage PSO approaches are more likely to find a satisfactory storage plan.

Figure 12 is the results in the large scenario. It could be told form the figure that GA approach outperforms PSO approach in the earlier stage of optimization for fast convergence, while in the later stage PSO approaches are more likely to find a satisfactory storage plan, as found in the small scenario.

Number of satisfactory solutions (N^0) and average time span of simulation runs (T^A) are listed in Table 9. In the 100 optimization runs of the small scenario, PSO approach finds 89 satisfactory solutions, while GA approach only finds 73. In the 100 optimization runs of the large scenario, PSO approach finds 82 satisfactory solutions, while GA approach only finds 69. Average time span of simulation runs is 27.3 seconds in the small scenario and 38.6 seconds in the large scenario.

The following could be concluded from Table 9 and Figure 12:

(1) Simulation-based optimization method could be used to solve the SAPOBA, while in this paper a satisfactory solution could not always be guaranteed.

(2) Due to the fact that simulation is quite time-consuming in evaluating candidate solutions, fewer simulation runs are preferred for optimization runs.

(3) PSO approach is more suitable than GA approach in simulation-based optimization for SAPOBA, since it is more likely to output a satisfactory storage plan.

6. Conclusion

This paper proposes a simulation-based optimization method for storage allocation problems of outbound containers in automated container terminals, which has rarely been mentioned in the literature. Considering that there are quite a number of concurrent and asynchronous processes in the container handling system, Timed-Colored-Petri-Net method is used to build up the simulation model, which is used to obtain the average QC waiting time in container handling in case that outbound containers are stacked according to storage plan. Two optimization approaches, using Genetic

Algorithm and Particle Swarm Optimization, respectively, are proposed to form the complete simulation-based optimization method. Experiments are conducted to verify the effectiveness of the simulation-based optimization method proposed and to get the PSO and GA approaches compared. Results of the experiments shows that PSO approach is more likely to output a satisfactory solution, although GA approach converge faster in the early stage. Further work could be verifying the simulation model in real automated container terminals, shortening the time span of simulation runs and improving the performance of optimization approaches.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research is supported by the Specialized Research Fund for the Doctoral Program of Higher Education of Ministry of Education of the People's Republic of China (no. 20133121110005), the Innovation Projects for Graduate Student at Shanghai Maritime University (no. 2013ycx050), "Scientific Research Innovation Project" of Shanghai Municipal Education Commission (no. 14ZZ140), the "Ph.D. Innovation Program" of Shanghai Maritime University (no. 2014ycx040), and the "Yang Fan Plan for Shanghai Youth Science and Technology Talent" of the Science and Technology Commission of Shanghai Municipality (no. 15YF1404900).

References

[1] C. Mi, X. He, H. Liu, Y. Huang, and W. Mi, "Research on a fast human-detection algorithm for unmanned surveillance area in bulk ports," *Mathematical Problems in Engineering*, vol. 2014, Article ID 386764, 17 pages, 2014.

[2] C. Mi, Y. Shen, W. Mi, and Y. Huang, "Ship identification algorithm based on 3D point cloud for automated ship loaders," *Journal of Coastal Research*, vol. 73, pp. 28–34, 2015.

[3] R. Gujjula and H.-O. Günther, "The impact of storage block assignment for import containers on AGV dispatching in highly automated seaport container terminals," in *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management (IEEM '08)*, pp. 1739–1743, Singapore, December 2008.

[4] J. Luo and Y. Wu, "Modelling of dual-cycle strategy for container storage and vehicle scheduling problems at automated container terminals," *Transportation Research Part E: Logistics and Transportation Review*, vol. 79, pp. 49–64, 2015.

[5] R. Dekker, P. Voogd, and E. van Asperen, "Advanced methods for container stacking," in *Container Terminals and Cargo Systems*, pp. 131–154, Springer, Berlin, Germany, 2007.

[6] B. Borgman, E. van Asperen, and R. Dekker, "Online rules for container stacking," *OR Spectrum*, vol. 32, no. 3, pp. 687–716, 2010.

[7] T. Park, R. Choe, Y. Hun Kim, and K. Ryel Ryu, "Dynamic adjustment of container stacking policy in an automated container terminal," *International Journal of Production Economics*, vol. 133, no. 1, pp. 385–392, 2011.

[8] M. Yu and X. Qi, "Storage space allocation models for inbound containers in an automatic container terminal," *European Journal of Operational Research*, vol. 226, no. 1, pp. 32–45, 2013.

[9] H. J. Carlo, I. F. A. Vis, and K. J. Roodbergen, "Storage yard operations in container terminals: literature overview, trends, and research directions," *European Journal of Operational Research*, vol. 235, no. 2, pp. 412–430, 2014.

[10] K. H. Kim and K. T. Park, "A note on a dynamic space-allocation method for outbound containers," *European Journal of Operational Research*, vol. 148, no. 1, pp. 92–101, 2003.

[11] W. Mi, W. Yan, J. He, and D. Chang, "An investigation into yard allocation for outbound containers," *COMPEL*, vol. 28, no. 6, pp. 1442–1457, 2009.

[12] Y. Zhang, Q. Zhou, Z. Zhu, and W. Hu, "Storage planning for outbound container on maritime container terminals," in *Proceedings of the IEEE International Conference on Automation and Logistics (ICAL '09)*, pp. 320–325, IEEE, Shenyang, China, August 2009.

[13] M. Li and S. Li, "An effective heuristic for yard template design in land-scarce container terminals," in *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM '11)*, pp. 908–912, Singapore, December 2011.

[14] L. Chen and Z. Lu, "The storage location assignment problem for outbound containers in a maritime terminal," *International Journal of Production Economics*, vol. 135, no. 1, pp. 73–80, 2012.

[15] Y. H. Jeong, K. H. Kim, Y. J. Woo et al., "A simulation study on a workload-based operation planning method in container terminals," *Industrial Engineeering & Management Systems*, vol. 11, no. 1, pp. 103–113, 2012.

[16] D.-H. Lee, J. G. Jin, and J. H. Chen, "Schedule template design and storage allocation for cyclically visiting feeders in container transshipment hubs," *Transportation Research Record*, vol. 2273, no. 1, pp. 87–95, 2012.

[17] O. Sharif and N. Huynh, "Storage space allocation at marine container terminals using ant-based control," *Expert Systems with Applications*, vol. 40, no. 6, pp. 2323–2330, 2013.

[18] D. T. Eliiyi, G. Mat, and B. Ozmen, "Storage optimization for export containers in the port of izmir," *PROMET—Traffic & Transportation*, vol. 25, no. 4, pp. 359–367, 2013.

[19] W. Hu, H. Wang, and Z. Min, "A storage allocation algorithm for outbound containers based on the outer-inner cellular automaton," *Information Sciences*, vol. 281, pp. 147–171, 2014.

[20] L. H. Lee, E. P. Chew, K. C. Tan, and Y. Han, "An optimization model for storage yard management in transshipment hubs," *OR Spectrum*, vol. 28, no. 4, pp. 539–561, 2006.

[21] Y. Han, L. H. Lee, E. P. Chew, and K. Tan, "A yard storage strategy for minimizing traffic congestion in a marine container transshipment hub," *OR Spectrum*, vol. 30, no. 4, pp. 697–720, 2008.

[22] L. P. Ku, L. H. Lee, E. P. Chew, and K. C. Tan, "An optimisation framework for yard planning in a container terminal: case with automated rail-mounted gantry cranes," *OR Spectrum*, vol. 32, no. 3, pp. 519–541, 2010.

[23] L. P. Ku, E. P. Chew, L. H. Lee, and K. C. Tan, "A novel approach to yard planning under vessel arrival uncertainty," *Flexible Services and Manufacturing Journal*, vol. 24, no. 3, pp. 274–293, 2012.

[24] X. Jiang, L. H. Lee, E. P. Chew, Y. Han, and K. C. Tan, "A container yard storage strategy for improving land utilization and operation efficiency in a transshipment hub port," *European Journal of Operational Research*, vol. 221, no. 1, pp. 64–73, 2012.

[25] S. H. Won, X. Zhang, and K. H. Kim, "Workload-based yard-planning system in container terminals," *Journal of Intelligent Manufacturing*, vol. 23, no. 6, pp. 2193–2206, 2012.

[26] X. Jiang, E. P. Chew, L. H. Lee, and K. Tan, "Flexible space-sharing strategy for storage yard management in a transshipment hub port," *OR Spectrum*, vol. 35, no. 2, pp. 417–439, 2013.

[27] L. Zhen, "Yard template planning in transshipment hubs under uncertain berthing time and position," *Journal of the Operational Research Society*, vol. 64, no. 9, pp. 1418–1428, 2013.

[28] R. McCormack and G. Coates, "A simulation model to enable the optimization of ambulance fleet allocation and base station location for increased patient survival," *European Journal of Operational Research*, vol. 247, no. 1, pp. 294–309, 2015.

[29] B. Dengiz, Y. T. İç, and O. Belgin, "A meta-model based simulation optimization using hybrid simulation-analytical modeling to increase the productivity in automotive industry," *Mathematics and Computers in Simulation*, 2015.

[30] J.-F. Cordeau, P. Legato, R. M. Mazza, and R. Trunfio, "Simulation-based optimization for housekeeping in a container transshipment terminal," *Computers and Operations Research*, vol. 53, pp. 81–95, 2015.

[31] T. Murata, "Petri nets: properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[32] W. M. Zuberek, "Timed Petri nets definitions, properties, and applications," *Microelectronics Reliability*, vol. 31, no. 4, pp. 627–644, 1991.

[33] J. Liu, X. Ye, and J. Zhou, "Test purpose oriented I/O conformance test selection with colored petri nets," *Journal of Applied Mathematics*, vol. 2014, Article ID 645235, 10 pages, 2014.

[34] H. Zhang, H. Zhang, M. Gu, and J. Sun, "Modeling a heterogeneous embedded system in coloured Petri nets," *Journal of Applied Mathematics*, vol. 2014, Article ID 943094, 8 pages, 2014.

[35] J. Zhou, C. Sun, W. Fu, J. Liu, L. Jia, and H. Tan, "Modeling, design, and implementation of a cloud workflow engine based on aneka," *Journal of Applied Mathematics*, vol. 2014, Article ID 512476, 9 pages, 2014.

[36] Y.-S. Weng, Y.-S. Huang, Y.-L. Pan, and M. Jeng, "Design of traffic safety control systems for railroads and roadways using timed Petri nets," *Asian Journal of Control*, vol. 17, no. 2, pp. 626–635, 2015.

[37] C. Li, W. Wu, Y. Feng, and G. Rong, "Scheduling FMS problems with heuristic search function and transition-timed Petri nets," *Journal of Intelligent Manufacturing*, 2014.

[38] D. Briskorn, A. Drexl, and S. Hartmann, "Inventory-based dispatching of automated guided vehicles on container terminals," in *Container Terminals and Cargo Systems*, pp. 195–214, Springer, Berlin, Germany, 2007.