International Journal of Management Science
Vol 5, No 1, May 1999

A Dispatching Method for Automated Guided Vehicles to Minimize Delays of Containership Operations*

Kap-Hwan Kim
Department of Industrial Engineering,
Pusan National University, Pusan, 609-735, Korea

Jong-Wook Bae
Research Institute of Mechanical Technology,
Pusan National University, Pusan, 609-735, Korea

(Received November 1998 ; revision received January 1999)

ABSTRACT

There is a worldwide trend to automate the handling operations in port container terminals in an effort to improve productivity and reduce labor cost. This study discusses how to apply an AGV (automated guided vehicle) system to the handling of containers in the yard of a port container terminal. The main issue of this paper is how to assign tasks of container delivery to AGVs during ship operations in an automated port container terminal. A dual-cycle operation is assumed in which the loading and the discharging operation can be performed alternately. Mixed integer linear programming formulations are suggested for the dispatching problem. The completion time of all the discharging and loading operations by a quayside crane is minimized, and the minimization of the total travel time of AGVs is also considered as a secondary objective. A heuristic method using useful properties of the dispatching problem is suggested to reduce the computational time. The performance of the heuristic algorithm is evaluated in light of solution quality and computation time.

1. INTRODUCTION

It is, in port container terminals, important to reduce the time of ship operations for a shorter turnaround time of a vessel. Ship operations include the discharging

* This work was supported by Grant No.971-1013-076-1 from the Korea Science and Engineering foundation.

operation, during which containers in a containership are unloaded from a ship and stacked in a marshalling yard, and the loading operation, during which containers are handled in the reverse direction of the discharging operation. In container terminals that we assume in this study, three types of equipment are used for ship operations: quay cranes (QCs), prime movers, and yard cranes. QCs transfer containers from a ship to a prime mover (we call this the "discharging operation"). Then, the prime mover delivers the inbound (import or discharging) container to a yard crane that picks it up and stacks into a position in a marshalling yard. For the loading operation, the process is carried out in the opposite direction. The handling activities performed by QCs are "waterside operations", while those performed by prime movers and yard cranes are "landside operations". We assume that the container terminal in this paper is automated, in which the yard crane may be an automated rail-mounted gantry crane, an automated stacking crane (ASC), or an overhead bridge crane, and the prime mover is an automated guided vehicle (AGV).

Figure 1 illustrates an automated port container terminal and equipment used in loading/discharging operations. It shows ASCs in marshalling yard that stack and pick up containers during the landside operation. And, it also shows how a QC releases an inbound container onto an empty AGV or picks up an outbound (export or loading) container from a loaded AGV.

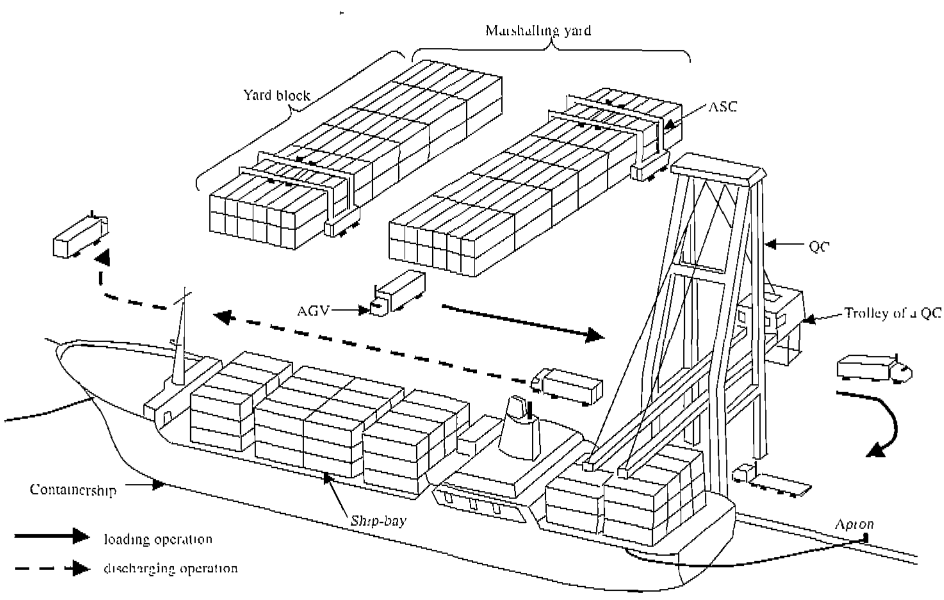


Figure 1. A bird's—eye view of ship operations in an automated port container terminal

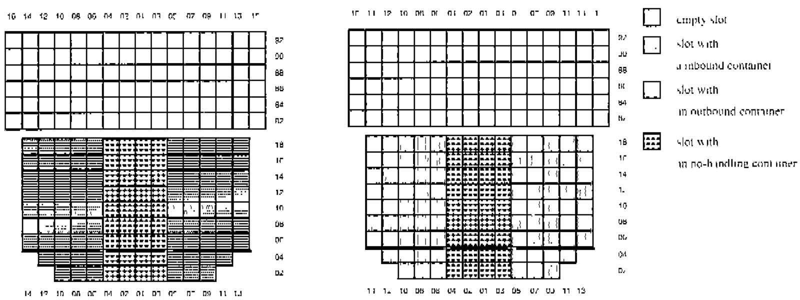
There are two alternative strategies for a waterside operation: a single-cycle operation and a dual-cycle operation. When the waterside operation is performed in the single-cycle manner by a QC, only either the loading operation or the discharging operation is carried out during one cycle. But, during the dual-cycle waterside operation, the loading operation and the discharging operation are performed alternately, so that the empty traversal time of a QC trolley is minimized.

The landside operation can also be performed in either a single-cycle manner or a dual-cycle manner. During the single-cycle landside operation, an AGV delivers a container from apron (yard) to yard (apron) and returns empty for the next inbound (outbound) container. When the dual-cycle landside operation is performed, an empty AGV that delivered an outbound container to a QC can receive another inbound container from the QC at the same position, and an AGV that delivered an inbound container to a yard crane can receive another outbound container instead of traveling empty to the apron. In this paper, we assume that both the waterside and the landside operations are performed in the dual-cycle manner.

Although discharging and loading tasks are pre-assigned to a QC according to a load/discharge sequence list, task assignments to yard equipment (yard cranes and AGVs) are determined dynamically, based on the their status when the operation is actually being performed. We call the decision process "dispatching". The main issue in this paper is how to assign the delivery tasks to AGVs in real time.

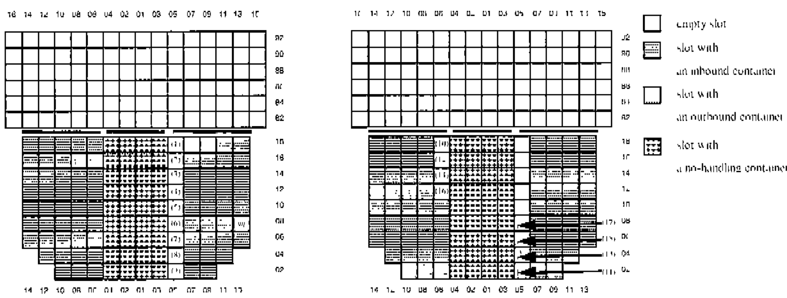
Planners in a Terminal Operation Company (TOC) construct a schedule for the waterside operation of a containership before the operation actually begins. For the planning, a ship agent usually transfers a guideline of discharging and loading operations, which is called a "load profile" to the TOC. A part of a load profile is illustrated in Figure 2. A slot in Figure 2 means a unit space in a ship-bay into which a container can be positioned. Figure 2(a) shows the slots, from which containers should be discharged, while Figure 2(b) shows the slots for a loading operation. It is common practice for a QC to begin outbound containers after all the discharging operations are completed for the same ship-bay. But, the empty traversal time of a QC trolley can be reduced if a loading operation is combined with a discharging operation, which is illustrated in Figure 3. In order to perform the tasks in Figure 2 in a dual-cycle manner, the QC has to first discharge a stack in a single-cycle manner, as in Figure 3(a). Then, discharging and loading operations can be performed alternately in order of the number indicated in each slot in Figure 3(b). Although the dual-cycle waterside process is known to be more efficient, compared with the single-cycle operation, it is also more complicated in terms of operation control.

Based on the bay profile as shown in Figure 2, a work schedule is constructed for each QC, as illustrated in Table 1. The work schedule shows tasks at each ship-bay, type of tasks, the number of containers of each task, a sequence of tasks, and a schedule for the tasks. Based on the work schedule, planners make a detailed schedule specifying the sequence of discharging and loading operations of individual containers. The sequence list shows the exact location of each container in the ship-bay and in the marshalling yard, as illustrated in Table 2.



(a) (b)

Figure 2. An example of a bay profile



(a) (b)

Figure 3. An example of a dual−cycle waterside operation

Table 1. An example of a QC working schedule for the dual−cycle operation

Ship-bay no.	Operation time		Discharging (number of containers)		Loading (number of containers)	
	From	To	20 feet	40 feet	20 feet	40 feet
#28 D	07:05	09:45	0	26	0	18
#33 H	09:45	11:20	30	0	0	0
#33 D	11:20	16:50	84	0	84	0
#36 H	16:50	19:30	0	47	0	0
#38 H	19:30	21:50	0	23	0	31

Table 2. An example of a QC working sequence list for the dual−cycle operation

Seq.	Operation of a QC				Requested interarrival time (RIT) (sec.) (d_i)	Earliest possible event time without delay (sec.) (s_i)
	Type(1)	Ship location(2)	Yard location(3)	Operation cycle time (sec.)		
1	D	33-10-38-D	B-11-02-03	185	20	165
2	L	33-10-36-D	D-11-01-01	190	275	185
3	D	33-12-32-D	B-07-04-03	105	20	460
4	L	33-10-36-D	D-07-04-02	205	205	480
5	L	33-10-38-D	E-12-03-03	205	295	685
6	D	33-06-36-D	A-11-01-02	110	:	980
:	:	:	:	:	:	:

(1) D: Discharging, L: Loading (2) Bay no.-Row no.-Tier no.-Hold/Deck (3) Block no.-Bay no.-Row no.-Tier no.

When a landside operation is performed in the single-cycle, assigning a delivery task to yard equipment is simple. A transfer task may be assigned to a yard crane nearest to the position of the target container. A delivery task can be assigned to the nearest AGV that is idle at that time. However, if the dual-cycle is applied to a landside operation, the decisions can be more complicated because the total travel time of AGVs may become different, according to the assignment of delivery tasks. When a pickup task is given after a delivery task to the same AGV, and both tasks can be done in positions close to each other, empty travel time can be significantly reduced. Reduced empty travel time of vehicles implies a higher throughput rate of AGVs, which will reduce the possibility of delays of loading and discharging operations by a QC resulting from late arrivals of vehicles. Considering that the most important objective of the ship operation is to complete the whole loading/unloading operation as rapidly as possible, the delay of the operation by a QC should be avoided by wisely assigning tasks to AGVs.

The dispatching problem of AGV systems are mostly studied in the manufacturing field. Maxwell and Muckstadt [9] proposed a dispatching model in which vehicle trips are assigned to routes so that the time between visits (deliveries or pickups) at each station are spread uniformly. In a study by Egbelu and Tanchoco [3], various dispatching rules were suggested for dynamic assignment of vehicles to tasks. The dispatching rules were classified into work-center initiated task assignment rules and vehicle initiated task assignment rules. Russell and Tanchoco [10] presented a simulation model to evaluate vehicle dispatching rules for a computer-dispatched lift truck in a job shop. They observed that dispatching rules do not affect very much the traditional measures of shop performance such as mean flow time and machine utilization. Egbelu [4] and Yim and Linn [11] proposed two different rules, pull or push strategies, to control material flows in a manufacturing environment. Ihsan and Hommertzheim [7] suggested a scheduling algorithm that considers the operation time of machines and the handling time of an AGV simultaneously. Klein and Kim [8] proposed that the multi-attribute decision-making method can be used for dispatching vehicles. More recently, Anwar and Nagi [1] focused on the simultaneous scheduling of processing and material handling operations for just-in-time (JIT) production of complex assemblies under multiple capacity constraints.

Durrant-Whyte [2] described the design of an autonomous guided vehicle system to transport ISO standard cargo containers in a port environment. Evers and Koppers [6] proposed a distributed control architecture for an AGV system that utilizes a signaling concept for traffic control. It is very useful when a large number of vehicles move in the same infra-structural facilities.

The problem discussed in this paper is different from the ones of previous studies in that the delivery tasks discussed here are already known and the sequence of the tasks are also predetermined, while previous studies assumed that the pickup calls are issued randomly and that the sequence of calls is unknown. When the state of the system changes dynamically and so is extremely uncertain, there may be no other way than making a decision on dispatching passively responding to the current state of the system whenever the decision has to be made. But, in case of loading and discharging operations, because pre-assigned vehicles serve a QC that performs tasks pre-specified and sequenced in advance, it is possible for the controller to utilize the information to preplan delivery tasks of vehicles.

2. PROBLEM DESCRIPTION

In this section, the dispatching problem will be defined and a mixed integer program is suggested. Two assumptions are introduced for the formulation of the problem:

(1) All the operation times for handling equipment to perform tasks are assumed to be deterministic, which include the operation time of a QC, travel time of an AGV, and the transfer time between equipment.

(2) Several AGVs are dedicated to a QC, and their specification, such as load capacity, travel velocity, and tasks, are the same.

2.1 Operations of a Quay Crane

We will illustrate an AGV dispatching problem and a method for solving it by using the data in Table 2. It also shows the initial and the final locations of each corresponding container. And, the operation cycle time in Table 2 includes the release or the pickup time of 20 seconds. An operation cycle of a loading operation by a QC begins with a pick-up of a container from an AGV in apron. It ends with a release of the container into a slot in a ship-bay when the succeeding operation is a discharging, while a positioning at a pickup location in the apron-side is the end of the operation cycle when the succeeding operation is a loading. An operation cycle of a discharging operation begins at a position of a ship-bay when the preceding operation is a loading, while it begins at a position of the apron-side when the preceding operation is a discharging. It ends with a release of a container onto an AGV. Thus, when a discharging operation follows just after a loading operation, the cycle times of both the loading and discharging operations become lower than others, which is the effect of a dual-cycle waterside operation.

The progress of the operation by a QC can be drawn as in Figure 4. Note that a discharging operation (for example, operations 1, 3, and 6) ends with a release of a container, while a loading operation begins with a pickup of a container. In order for the operation of a QC to proceed without delay, an AGV has to be ready before the transfer of a container begins at a specified location under the corresponding QC. We define an event, e_i , to be the beginning of pickup or release of a container by a QC. Then, the interval between adjacent events can be interpreted as a requested interarrival time (RIT) of an AGV, so that the operation of a QC will not be delayed. The length of a RIT between e_i and e_{i+1} is denoted by d_i , as in Figure 4, and the time of event i is denoted by y_i that can be replaced with s_i when any delay of the operation by the QC is not allowed. The event time, y_i , becomes a decision variable when the delay of an operation of a QC is allowed in section 4. Table 2 also illustrates the earliest possible event times for which no delay of an operation of a QC occurs.

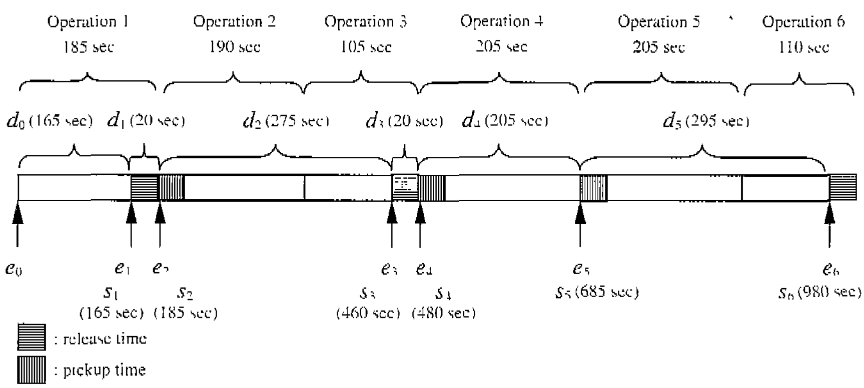


Figure 4. The operation cycle, the event time, and the requested interarrival time (RIT)

Some notations related to operations of a QC can be summarized as follows:

m : the number of tasks assigned to a QC. Tasks of a QC mean all the discharging and loading operations described in its working sequence list.

e_i : the event that corresponds to the beginning of pickup (or release) of a container by a QC from (onto) an AGV for the ith operation. e_0 represents the starting event of an operation of a QC. An AGV must be dispatched for every event in Figure 4, except e_0.

d_i : the length of RIT i that is defined as the interval between event i and event (i+1) under the assumption that there is no delay of an operation of a QC, where $0 \le i \le m-1$ and $d_0 = s_1$

s_i : the earliest possible time of event i under the condition that there is no delay. Note that $s_i = \sum_{k=0}^{i-1} d_k$, where $i > 1$.

y_i : the actual time of event i. When there is no delay in the operation of a QC, $y_i = s_i$. But, y_i is a decision variable, when a delay is allowed.

q_{ij} : the difference between event times, that is, $y_j - y_i\,(i < j)$. This can be evaluated by $s_j - s_i$ when no delay is allowed, which is shown in Table 3.

Table 3. The difference for the example between event times (q_{ij}) of a QC without delay

e_i \ e_j	e_1	e_2	e_3	e_4	e_5	e_6
e_0	165	185	460	480	685	980
e_1	-	20	295	315	520	815
e_2	-	-	275	295	500	795
e_3	-	-	-	20	225	520
e_4	-	-	-	-	205	500
e_5	-	-	-	-	-	295

2.2 Operations of AGVs

Figure 5 illustrates blocks in a marshalling yard, paths for AGVs, and locations of pickup and drop-off points (P/D points). In this example, ASCs are used as yard cranes and there are two AGVs to perform the tasks of ship operations. Thus, P/D points are located in front of blocks. Because the location of a QC does not change until all tasks for a ship-bay are completed, it is represented as a point. Considering the layout of the guide path for AGVs, the travel time between every combination of P/D points is provided in Table 4.

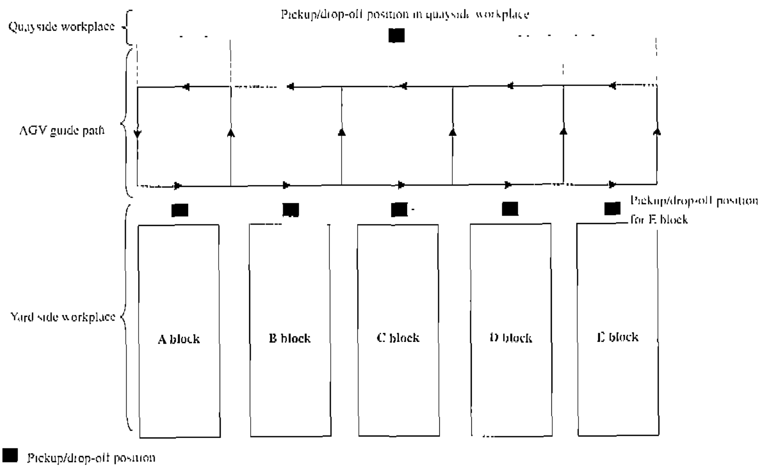


Figure 5. An example of a layout of a container terminal

Table 4. Travel time among P/D points in a container terminal (sec.)

From \ To	A block	B block	C block	D block	E block	QC
A block	-	30	60	90	120	255
B block	140	-	30	60	90	225
C block	170	140	-	30	60	195
D block	200	170	140	-	30	225
E block	230	200	170	140	-	255
QC	255	225	195	225	255	-

The problem studied in this paper is to assign tasks, for example, tasks given in Table 2, to minimize first the delay of the operation of a QC and then the total travel time of AGVs next. A higher priority is given to the operation of a QC because the ultimate objective of discharging and loading operations is to minimize the completion time of the ship operation. It can be accomplished by minimizing the idle time of the QC. This dispatching problem has the following three special properties that should be considered in developing a solution algorithm:

(1) Tasks should be carried out in the exact same order as specified in the sequence list.

(2) The objective of minimizing delays of operations by a QC has a higher priority than the objective of minimizing the total travel time of AGVs.

(3) A delay of a waterside operation by a QC results in delays, by the same amount of time, of all succeeding waterside operations by the same QC.

Types of events of AGVs that are closely related to the operation of a QC are the starting event, the event when an AGV begins to receive a container from a QC, the event when an AGV begins to transfer a container to a QC, and the stopping event after completing all assigned tasks.

For the formulation of dispatching problem, the following notations are introduced:

n : number of AGVs assigned to a QC.

$e_{(i,0)}$: the starting event of AGV i, $i = 1, \cdots, n$.

$e_{(i,m+1)}$: the stopping event of AGV i, $i = 1, \cdots, n$.

e_j : the event when an AGV begins to receive (transfer) a container from (to) a QC for the jth operation of a QC. Note that the same notation is used here as the jth event for the QC because both events are, in actuality, the same.

a_j : the location at the apron for the delivery of the container related to event e_j.

b_j : the location at the yard for the delivery of the container related to event e_j.

p_i : the initial location of AGV i, $i = 1, \cdots, n$.

T_{uv} : the travel time for an AGV to travel from location u to location v.

R_{QC} : the time for a QC to load a container onto an AGV (release time of QC).

P_{QC} : the time for a QC to unload a container from an AGV (pickup time of QC).

R_{ASC} : the time for an ASC to load a container onto an AGV (release time of ASC).

P_{ASC} : the time for an ASC to unload a container from an AGV (pickup time of ASC).

c_{ij} : the minimum elapsed time required for an AGV to be ready for e_j after it experiences e_i.

r_{ij} : the reachability of an AGV from e_i to e_j. By comparing c_{ij} with q_{ij}, it is possible to check if an AGV can be ready for e_j after experiencing e_i, so that the operation of the QC needed is not delayed. If yes, $r_{ij} = 1$. Otherwise, $r_{ij} = 0$. That is, an entry, r_{ij}, becomes 1 if $c_{ij} \le q_{ij}$.

Table 5 shows how to evaluate c_{ij} for various combinations of consecutive operations. An illustration of the c_{ij} matrix for the tasks in Table 2 is shown in Table 6.

Comparing Table 6 with Table 3, a reachability matrix, (r_{ij}), can be constructed, as in Table 7. Note that q_{0j} and ∞ are used in place of $q_{(i,0)j}$ and $q_{i(j,m+1)}$, respectively.

In the following section, we discuss how to assign a delivery task to an AGV so that the total travel time of AGVs is minimized in case that an assignment is possible without delaying the operation of a QC. In section 4, we allow a situation in which the operation of a QC is delayed, but with a much higher penalty cost than the travel cost of AGVs.

Table 5. Formulas for evaluating c_{ij}

Succeeding Event / Preceding Event	Event j related to A loading operation	Event j related to a discharging operation	Event $e_{(j,m+1)}$
Event $e_{(i,0)}$	$T_{p,b_i} + R_{ASC} + T_{b,a_j}$	T_{p,a_j}	0
Event i related to a loading operation	$P_{QC} + T_{a,b_i} + R_{ASC} + T_{b,a_j}$	P_{QC}	0
Event i related to a discharging operation	$R_{QC} + T_{a,b_i} + P_{ASC} + T_{b,b_i} + R_{ASC} + T_{b,a_j}$	$R_{QC} + T_{a,b_j} + P_{ASC} + T_{b,a_j}$	0

Table 6. c_{ij} for the example

e_i \ e_j	e_1	e_2	e_3	e_4	e_5	e_6	$e_{(1,7)}$	$e_{(2,7)}$
$e_{(1,0)}$	0	470	0	470	530	0	0	0
$e_{(2,0)}$	0	470	0	470	530	0	0	0
e_1	-	570	490	570	630	490	0	0
e_2	-	-	20	490	550	20	0	0
e_3	-	-	-	570	630	490	0	0
e_4	-	-	-	-	550	20	0	0
e_5	-	-	-	-	-	20	0	0
e_6	-	-	-	-	-	-	0	0

Table 7. The reachability matrix (r_{ij})

i \ j	e_1	e_2	e_3	e_4	e_5	e_6	$e_{(1,7)}$	$e_{(2,7)}$
$e_{(1,0)}$	1	0	1	1	1	1	1	1
$e_{(2,0)}$	1	0	1	1	1	1	1	1
e_1	0	0	0	0	0	1	1	1
e_2	0	0	1	0	0	1	1	1
e_3	0	0	0	0	0	1	1	1
e_4	0	0	0	0	0	1	1	1
e_5	0	0	0	0	0	1	1	1
e_6	0	0	0	0	0	0	1	1

3. WHEN DELAY OF A SHIP OPERATION IS NOT ALLOWED

Since any delay of an operation of a QC is not allowed, the objective of the problem is to minimize the total travel time of AGVs. The dispatching problem in this case can be formulated as an assignment problem that is defined by a bipartite graph, as in Figure 6(a). The bipartite graph is constructed by nodes and arcs, like a network. The set of nodes consists of the following subsets of nodes:

S : set of source nodes that correspond to the starting events. For instance, nodes $e_{(1,0)}$ and $e_{(2,0)}$ on the supply side in Figure 6(a) are included in this set.

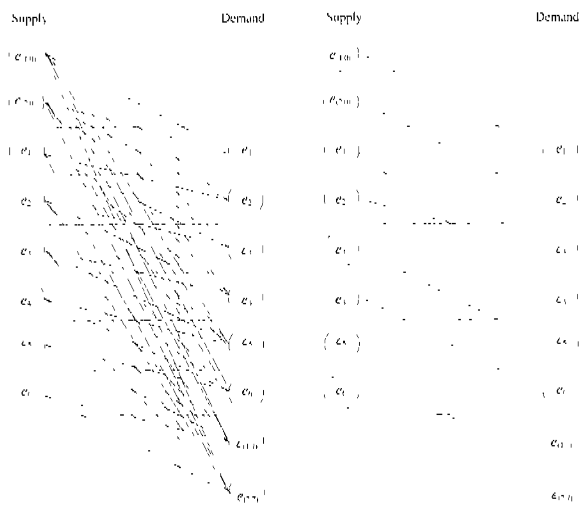
T : set of transshipment nodes that correspond to e_j. There are six transshipment nodes in Figure 6(a).

D : set of sink nodes that represent the stopping events of AGVs. Nodes $e_{(1,7)}$ and $e_{(2,7)}$ on the demand side in Figure 6(a) are included in this set.

Let

$$x_{ij} = \begin{cases} 1 & \text{if node } j \text{ on the demand side is assigned to node } i \text{ on the supply side} \\ 0, & \text{otherwise} \end{cases}$$

and c'_{ij} be the pure travel time for an AGV to go from event i to event j.



(a) (b)

Figure 6. Bipartite graph for a dispatching problem

Using data in Figures 4 and 5 and Table 3, c'_{ij} can be obtained as in Table 8. For example, the travel time of an AGV to go from event i to event j can be evaluated as follows:

An AGV needs 225 seconds to travel from the apron to block B, where the delivery of the inbound container is completed, and needs 60 seconds to travel from block B to block D, where a pickup of a container for loading is performed. Then, it has to move to the apron, which requires 225 seconds for the travel. Thus, it takes 510 seconds for an AGV to move from the last location of e_1 to the starting location of e_4.

Table 8. The travel time (c'_{ij}) matrix for the example

e_i \ e_j	e_1	e_2	e_3	e_4	e_5	e_6	$e_{(1,7)}$	$e_{(2,7)}$
$e_{(1,0)}$	0	450	0	450	510	0	0	0
$e_{(2,0)}$	0	450	0	450	510	0	0	0
e_1	-	510	450	510	570	450	0	0
e_2	-	-	0	450	510	0	0	0
e_3	-	-	-	510	570	450	0	0
e_4	-	-	-	-	510	0	0	0
e_5	-	-	-	-	-	0	0	0
e_6	-	-	-	-	-	-	0	0

r_{ij} is the entry of the reachability matrix. Since the reachability matrix in Table 7 results in no feasible assignment, we will temporarily use a new reachability matrix, as shown in Table 9, for the illustration of the solution of this section.

Table 9. The temporary reachability (r_{ij}) matrix for the example

e_i \ e_j	e_1	e_2	e_3	e_4	e_5	e_6	$e_{(1,7)}$	$e_{(2,7)}$
$e_{(1,0)}$	1	1	1	1	1	1	1	1
$e_{(2,0)}$	1	1	1	1	1	1	1	1
e_1	-	1	1	1	1	1	1	1
e_2	-	-	1	1	1	1	1	1
e_3	-	-	-	1	1	1	1	1
e_4	-	-	-	-	1	1	1	1
e_5	-	-	-	-	-	1	1	1
e_6	-	-	-	-	-	-	1	1

From the problem definition expressed in Figure 6, the dispatching problem can be formulated as follows:

$$\text{Minimize} \quad \sum_{i \in S \cup T} \sum_{j \in T \cup D} c'_{ij} x_{ij} \tag{1}$$

subject to

$$\sum_{j \in T \cup D} r_{ij} x_{ij} = 1, \qquad \forall i \in S \cup T \tag{2}$$

$$\sum_{i \in S \cup T} r_{ij} x_{ij} = 1, \qquad \forall j \in T \cup D \tag{3}$$

$$x_{ij} = 0 \text{ or } 1. \tag{4}$$

The objective function (1) is to minimize the total travel time of AGVs. Equations (2) and (3) imply that every node in $S \cup T$ must be assigned to one and only one node in $T \cup D$, and vice versa.

The formulations (1) – (4) are known as the assignment problem. Thus, the Hungarian Method can be used to solve the problem. Figure 6(b) illustrates the optimal feasible solution to the assignment problem. According to the solution, AGV 1 is supposed to move through the starting event $e_{(1,0)}$, event e_1, event e_4, event e_6, and the stopping event, e_{17}, which means that it delivers the inbound container of the first operation, the outbound container of the fourth operation, and the inbound container of the sixth operation. AGV 2 is supposed to move through the starting event $e_{(2,0)}$, event e_2, event e_3, event e_5, and the stopping event, e_{17}, which means that it delivers the outbound container of the second operation, the inbound container of the third operation, and the outbound container of the fifth operation.

4. WHEN DELAY OF A SHIP OPERATION IS ALLOWED WITH A HIGH PENALTY COST

If there is no feasible solution for problems (1) – (4), this means that a QC has to wait for the arrival of an AGV at a given moment of the operation. It is assumed that the prevention of a delay of an operation of a QC is a more important objective than minimizing the total travel time of AGVs. Thus, we will first try to minimize the completion time then try to minimize the total travel time of AGVs, under the condition that the minimum possible completion time is maintained.

Let υ be the cost of an AGV for the travel of unit distance, and ρ be the penalty cost per unit time for the delay of the completion time. In this paper, we assume $\rho >> \upsilon$.

The dispatching problem in this section can be formulated as follows:

$$\text{Minimize} \quad \sum_{i \in S \cup T} \sum_{j \in T \cup D, i<j} \upsilon c'_{ij} x_{ij} + \rho (y_m - s_m) \tag{5}$$

subject to

$$\sum_{j \in T \cup D} x_{ij} = 1, \qquad \forall i \in S \cup T \text{, and } i < j \text{ for } i,j \in T \tag{6}$$

$$\sum_{i \in S \cup T} x_{ij} = 1, \qquad \forall j \in T \cup D \text{ and } i < j \text{ for } i,j \in T \tag{7}$$

$$y_j - (y_i + c_{ij}) \geq M(x_{ij} - 1), \qquad \forall i \in S \cup T, j \in T \text{, and } i < j \tag{8}$$

$$y_i \geq y_{i-1} + d_i, \qquad \forall i \in T \tag{9}$$

$$x_{ij} = 0 \text{ or } 1, \tag{10}$$

where M = a very large number.

The objective (equation (5)) is to first minimize the penalty cost and then minimize the total travel cost of AGVs because $\rho >> \upsilon$. Equations (6), (7), and (10) are the same as formulations (2) - (4). Equations (8) and (9) come from the definitions of c_{ij}, d_i, and y_i. The above formulations can be solved by using LINDO®. For the problem of Tables 3, 4, 6, and 7, (y_1^*, y_2^*, y_3^*, y_4^*, y_5^*, y_6^*) = (160, 470, 745, 765, 1315, 1610). $x^*_{(1,0)2} = x^*_{(2,0)} = x^*_{14} = x^*_{23} = x^*_{36} = x^*_{45} = x^*_{5(2,7)} = x^*_{6(1,7)}$ = 1, and all the other x^*_{ij} = 0. The solution can be interpreted as the assignment of tasks to AGV 1 and AGV 2, as follows:

AGV1: idle at the apron → perform task 2 (delivering an outbound container from block D to the apron) → perform task 3 (delivering an inbound container from the apron to block B) → perform task 6 (delivering an inbound container from the apron to block A)

AGV2: idle at the apron → perform task 1 (delivering an inbound container from the apron to block B) → perform task 4 (delivering an outbound container from block D to the apron) → perform task 5 (delivering an outbound con-

tainer from block E to the apron)
According to this solution, the events will be delayed, as in Table 10.

Table 10. Delays of event times in the example

Event	Event time without delay (s_i)	Event time considering the delay (y_i)	Delayed time
e_2	185	470	285
e_3	460	745	285
e_4	480	765	285
e_5	685	1315	630
e_6	980	1610	630

5. A HEURISTIC METHOD

In this section, a heuristic algorithm is suggested to solve the problem. Before the algorithm is described, we define constraint subset k, which has fewer variables than the original constraint sets (2) - (4).
(Constraint subset k)

$$\sum_{j \in T_k \cup D} r_{ij} x_{ij} = 1, \qquad \forall i \in S \cup T_k \tag{11}$$

$$\sum_{i \in S \cup T_k} r_{ij} x_{ij} = 1, \qquad \forall j \in T_k \cup D \tag{12}$$

$$x_{ij} = 0 \text{ or } 1 \tag{13}$$

where T_k includes indices of $e_1, e_2, \cdots, e_k$.

Property 1: If constraint subset k is infeasible, then constraint subset $(k+1)$ is also infeasible.

Proof : Suppose that nodes in constraint subset k are arranged as in Figure 6. Notice that all the arcs from the nodes on the supply side to nodes on the demand side are directed downward. The fact that constraint subset k is infeasible implies that we cannot find a set of arcs that connect nodes on the supply side with nodes on the demand side in a one-to-one manner. Considering the fact that each node in D has an arc from every node in $S \cup T_k$, it means that we cannot find a solution in which every node in T_k on the demand side is connected to a node on the supply side. Suppose that two nodes for event $(k+1)$ are added to the network of constraint subset k. But, no arc is added that is connected to a node in T_k on the demand side. Thus, the infeasibility is not resolved by adding two nodes for event $(k+1)$ and corresponding arcs. **Q.E.D.**

In the algorithm, the feasibility is checked from constraint subset 1 to constraint subset m, one by one. In the process, when an infeasible constraint subset, for example, subset k, is found, we resolve the infeasibility by adding one or more arcs to the bipartite graph.

The heuristic solution procedure is summarized in the following:

step 1: (Initialization) Set $y_i = s_i$ for $i = 1,\ldots,m$, and k = 1. Evaluate q_{ij} from (y_i). Evaluate r_{ij} from q_{ij} and c_{ij}.

step 2: (Feasibility checking) If $k > m$, go to step 4. Check for the existence of a feasible solution to constraint subsets (11) – (13). If there exists a feasible solution, set $k = k + 1$ and go to the beginning of this step. Otherwise, go to step 3.

step 3: (Delaying event times) Find the minimum value among $\pi_{ik} = c_{ik} - q_{ik}$ for all i such that $r_{ik} = 0$, where $i \in S$ or $i < k\ (i \in T)$ and denote it as Δ. For all $y_j\ (j \geq k)$, $y_j = y_j + \Delta$. Revise $Q = (q_{ij})$ and $R = (r_{ij})$. Go to step 2.

step 4: (Task assignment) Solve the assignment problems (1) - (4). Stop.

Feasibility checking

For a given matrix (r_{ij}), the feasibility can be checked by solving a maximum-cardinality-matching problem in the bipartite graph [5]. When the maximum cardinality is the same as the number of nodes on either the demand or the supply side, we can conclude that constraint subset k, (11) - (13), has a feasible solution. Otherwise, it is infeasible.

Delaying event times

In order to make an infeasible constraint subset feasible, one or more arcs have to be added to the bipartite graph. Note that we make constraint subset k – 1 feasible before we proceed to check the feasibility of constraint subset k, according to the solution procedure. The questions are, which arcs should be added, which y_i's should be increased, and how much they should be increased in order to add the intended arcs. For that purpose, we introduce the minimum delay matrix for adding arcs (π_{ij}) that is $\max\{(c_{ij} - q_{ij}), 0\}$. In the evaluation, q_{0j} and ∞ are used as $q_{(i,0)j}$ and $q_{i(j,m+1)}$, respectively. Suppose it is found that constraint subset 2 is infeasible. Thus, the minimum delay matrix is calculated as in Table 11. According to Table 9, if event 2 is delayed by 550 seconds, three arcs from $e_{(1,0)}$ to

e_2, from $e_{(2,0)}$ to e_2, and from e_1 to e_2 are added to the bipartite graph.

Table 11. An example of the minimum delay matrix (π_{ij})

e_i \ e_j	e_1	e_2	e_3	e_4	e_5	e_6	$e_{(1,7)}$	$e_{(2,7)}$
$e_{(1,0)}$	0	285	0	0	0	0	0	0
$e_{(2,0)}$	0	285	0	0	0	0	0	0
e_1	-	550	195	255	110	0	0	0
e_2	-	-	0	195	50	0	0	0
e_3	-	-	-	550	405	0	0	0
e_4	-	-	-	-	345	0	0	0
e_5	-	-	-	-	-	0	0	0
e_6	-	-	-	-	-	-	0	0

Considering that a higher priority is given to minimizing the delay of an operation of a QC, we begin by adding one arc with the minimum additional delay. In this example, the first arc can be added by increasing y_2 by 285 seconds. If the resulting assignment problem is still infeasible, we increase y_2 up to 570 seconds (Note that c_{12} = 570 in Table 6) next, and so on. In the process, when the constraint subset becomes feasible, we stop the iterative process, and go to the next stage.

Two main steps of feasibility checking and delaying events are the same as solving the following problem:

Minimize $\quad (y_k - s_k)^+$ (14)

Subject to

$\sum_{j\in T_k\cup D} x_{ij} = 1,\qquad \forall i \in S \cup T_k$, and $i<j$ for $i,j \in T_k$ (15)

$\sum_{i\in S\cup T_k} x_{ij} = 1,\qquad \forall j \in T_k \cup D$ and $i<j$ for $i,j \in T_k$ (16)

$y_j - (y_i + c_{ij}) \geq M(x_{ij} - 1),\quad \forall i \in S \cup T_k,\ j \in T_k$, and $i<j$ (17)

$y_i \geq y_{i-1} + d_i,\qquad \forall i \in T_k$ (18)

$x_{ij} = 0$ or 1, (19)

where M = a very large number, and $(z)^+ = \max\{z, 0\}$.

We next introduce a useful property to support the rationale of the heuristic procedure.

Property 2: The steps of feasibility checking and delaying events in stage k provide the optimal solution to problems (14) - (19).

Proof : When the constraint subset is infeasible at k = 1, problems (14) - (19) become feasible by delaying y_1 by $\min_i \{\pi_{(i,0)1}\}$. Suppose that the above property holds at stage (k – 1). Subproblem k is equivalent to finding $y_1,\cdots,y_k$ that make constraint subset k feasible and minimize y_k at the same time. If constraint subset k is infeasible, one or more arcs to transshipment node k have to be added to the bipartite graph. Note that an arc is added to transshipment node k whenever y_k is increased beyond $y_k + \pi_{ik}$ for $i < k$. By Property 2, the feasibility checking and the delaying event times procedures result in $(y_1,\cdots,y_m)$ that gives the minimum delay of the completion time of an operation of a QC. Thus, it minimizes y_k to check the feasibility of constraint subset k, while increasing y_k by π_{ik} in an increasing order. **Q.E.D.**

However, steps 1 - 3 do not necessarily lead to $(y_1,\cdots,y_m)$ that give the minimum total travel time of AGVs, which is the secondary objective of the original problem. This is the reason why we consider the algorithm as a heuristic procedure.

6. A NUMERICAL EXAMPLE AND EXPERIMENTATION

The numerical illustration given below is a step-by-step illustration of the algorithm for the example.

(1) (Initialization) From Table 2, $(y_1,y_2,y_3,y_4,y_5,y_6)$ = (165, 185, 460, 480, 685, 980). The matrices q_{ij}, c_{ij}, and r_{ij} are in Tables 3, 6, and 7, respectively. Set k = 1.

(2) (Feasibility checking) k = 1. There is a feasible solution to constraint subset 1. $k = k + 1$.

(3) (Feasibility checking) k = 2. There is no feasible solution to constraint subset 2.

(4) (Delaying event times) $\pi_{i2} = (\pi_{(1,0)2}, \pi_{(2,0)2}, \pi_{12})$ = (285, 285, 550). Thus, Δ = 285, which results in $(y_1,y_2,y_3,y_4,y_5,y_6)$ = (165, 470, 745, 765, 970, 1265). Calculate q_{ij} and r_{ij} based on the revised y_i. Revised q_{ij} and r_{ij} are as in Tables 12 and 13, respectively.

(5) (Feasibility checking) There is a feasible solution to constraint subset 2. $k = k + 1$.

(6) (Feasibility checking) k = 3. There is a feasible solution to constraint subset 3. $k = k + 1$.

(7) (Feasibility checking) $k = 4$. There is a feasible solution to constraint subset 4. $k = k + 1$.
(8) (Feasibility checking) $k = 5$. There is no feasible solution to constraint subset 5.
(9) (Delaying event times) $\pi_{i5} = (\pi_{(1,0)5}, \pi_{(2,0)5}, \pi_{15}, \pi_{25}, \pi_{35}, \pi_{45}) = (0, 0, 0, 50, 405, 345)$. Thus, $\Delta = 50$, which results in $(y_1, y_2, y_3, y_4, y_5, y_6) = (165, 470, 745, 765, 1020, 1315)$. Calculate q_{ij} and r_{ij} based on the revised y_i.
(10) (Feasibility checking) There is no feasible solution to constraint subset 5.
(11) (Delaying event times) $\pi_{i5} = (\pi_{(1,0)5}, \pi_{(2,0)5}, \pi_{15}, \pi_{25}, \pi_{35}, \pi_{45}) = (0, 0, 0, 0, 355, 295)$. Thus, $\Delta = 295$, which results in $(y_1, y_2, y_3, y_4, y_5, y_6) = (165, 470, 745, 765, 1315, 1610)$. Calculate q_{ij} and r_{ij} based on the revised y_i.
(12) (Feasibility checking) There is a feasible solution to constraint subset 5. $k = k + 1$.
(13) (Feasibility checking) k = 6. There is a feasible solution to constraint subset 6. $k = k + 1$.
(14) (Task assignment) By solving the assignment problem, we can get the results equal to the optimal solution, as in Figure 7(c).

Table 12. The difference between event times (q_{ij}) of a QC, considering the delay for the example

e_i \ e_j	e_1	e_2	e_3	e_4	e_5	e_6
e_0	165	470	745	765	970	1265
e_1	-	305	580	600	805	1100
e_2	-	-	275	295	500	795
e_3	-	-	-	20	225	520
e_4	-	-	-	-	205	500
e_5	-	-	-	-	-	295

Table 13. The reachability matrix (r_{ij}) to constraint subset 2

i \ j	e_1	e_2	e_3	e_4	e_5	e_6	$e_{(1,7)}$	$e_{(2,7)}$
$e_{(1,0)}$	1	1	1	1	1	1	1	1
$e_{(2,0)}$	1	1	1	1	1	1	1	1
e_1	0	0	1	1	1	1	1	1
e_2	0	0	1	0	0	1	1	1
e_3	0	0	0	0	0	1	1	1
e_4	0	0	0	0	0	1	1	1
e_5	0	0	0	0	0	1	1	1
e_6	0	0	0	0	0	0	1	1

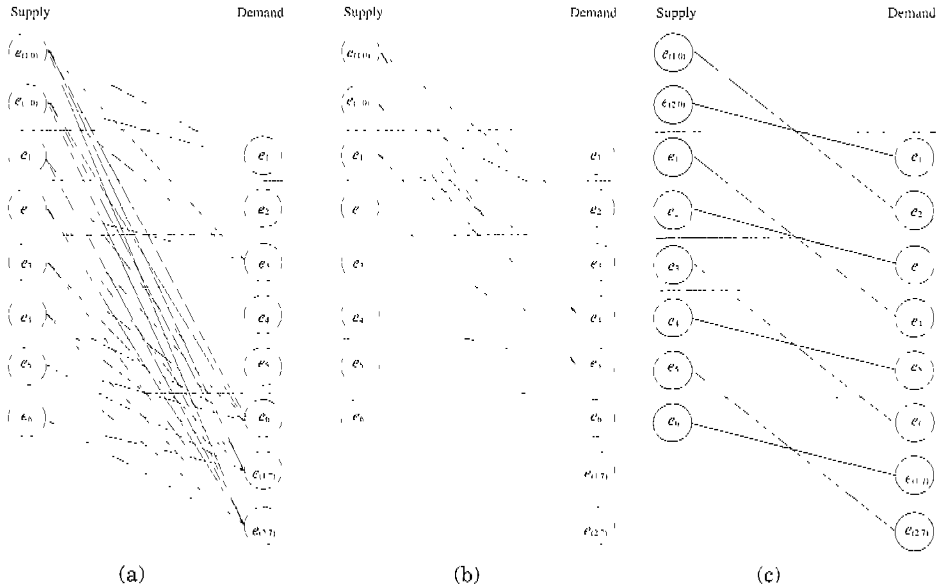


Figure 7. An example of a heuristic algorithm application

An arc from node i on the supply side to node j on the demand side is connected when $y_j - y_i \geq c_{ij}$. Figure 7(a) shows the graph when $y_i = s_i$ for all i, which means that there is no delay of an operation of a QC. Note that some nodes on the demand side have no incoming arcs, which means the assignment is impossible without delays of operations of a QC.

During the whole process of the algorithm, y_i are increased, and new arcs are added to the initial bipartite graph in order to make constraint subset k feasible. Figure 7(a) represents the initial bipartite graph made from Table 7, and Figure 7(b) shows arcs added to the initial graph during the whole process of the heuristic algorithm.

In this section, we also provide the results of various experiments in order to compare the optimal solutions with those provided by the heuristic algorithm. P/D stations in the yard side and the operating time of waterside operations are determined randomly. The same data in the previous example were used as the travel time of an AGV and the release/pickup time of a QC/ASC. The performances of solution methodologies are evaluated by the completion time of the last operation of a QC and the computation time. The results of the experiment are summarized in Tables 14 – 18. The results in different tables are obtained using different combinations of the number of tasks and the number of AGVs.

We solved ten randomly generated problems for each combination of n and m using LINDO® software. The computational time increased rapidly as the number of AGVs or the number of tasks increased. Therefore, because the dispatching decisions must be made in real time, we could conclude that directly solving the mathematical programming model is impractical in real practice.

The same problems were also solved by the heuristic method shown in this paper. The objective value of the heuristic method exceeded the optimal objective value by no more than 3% on average for all the combination of n and m. But, the computational times of the heuristic method were significantly reduced, compared with those for the optimal solution.

Table 14. Comparison of mixed integer programming with the heuristic method ($n = 2$ and $m = 10$)

No.	Mixed integer programming		Heuristic Algorithm		Performance	
	Computational time (A)	Objective value (B)	Computational Time (C)	Objective value (D)	D/ B	C/ A
1	26	2620	0.25	2710	1.03	0.04
2	15	2669	1.65	2729	1.02	0.07
3	9	2239	0.27	2349	1.05	0.11
4	10	2427	0.27	2427	1.00	0.10
5	12	2431	0.36	2531	1.04	0.08
6	10	2239	0.27	2239	1.00	0.10
7	19	2754	0.23	2754	1.00	0.05
8	11	2582	0.23	2582	1.00	0.09
9	20	2750	0.33	2860	1.04	0.05
10	7	2237	0.32	2405	1.08	0.13
Average	13.9	2494.8	0.42	2558.6	1.03	0.08

Table 15. Comparison of mixed integer programming with the heuristic method ($n = 3$ and $m = 10$)

No.	Mixed integer programming		Heuristic method		Performance	
	Computational time (A)	Objective Value (B)	Computational Time (C)	Objective value (D)	D/ B	C/ A
1	58	2259	0.91	2259	1.00	0.02
2	9	2274	0.90	2274	1.00	0.11
3	20	1924	0.90	1924	1.00	0.05
4	149	2040	0.91	2040	1.00	0.01
5	62	1950	0.90	1950	1.00	0.02
6	35	1984	0.90	1984	1.00	0.03
7	3	2414	0.90	2414	1.00	0.33
8	177	2329	0.90	2329	1.00	0.01
9	14	2409	0.90	2409	1.00	0.07
10	13	1978	1.65	1978	1.00	0.08
Average	54	2156.1	0.98	2156.1	1.00	0.07

Table 16. Comparison of mixed integer programming with the heuristic method ($n = 4$ and $m = 10$)

No.	Mixed integer programming		Heuristic method		Performance	
	Computational time (A)	Objective value (B)	Computational time (C)	Objective value (D)	D/ B	C/ A
1	21	2259	1.65	2259	1.00	0.05
2	4	2274	0.90	2274	1.00	0.25
3	5	1924	0.10	1924	1.00	0.20
4	9	2040	0.90	2040	1.00	0.11
5	173	1950	1.65	1950	1.00	0.01
6	40	1984	1.65	1984	1.00	0.03
7	7	2414	0.90	2414	1.00	0.14
8	636	2329	0.10	2329	1.00	0.00
9	151	2409	1.65	2409	1.00	0.01
10	120	1978	0.10	1978	1.00	0.01
Average	116.6	2156.1	0.96	2156.1	1.00	0.08

Table 17. Comparison of mixed integer programming with the heuristic method ($n = 2$ and $m = 15$)

No.	Mixed integer programming		Heuristic method		Performance	
	Computational Time (A)	Objective value (B)	Computational time (C)	Objective value (D)	D/ B	C/ A
1	199	3680	1.65	3796	1.03	0.01
2	223	3647	1.65	3647	1.00	0.01
3	112	3606	1.65	3716	1.03	0.01
4	197	3642	1.65	3642	1.00	0.01
5	176	3814	1.31	3960	1.04	0.01
6	179	3486	1.65	3486	1.00	0.01
7	164	3872	1.65	3941	1.02	0.01
8	87	3573	1.65	3573	1.00	0.02
9	127	3694	0.75	3804	1.03	0.01
10	134	3530	1.65	3698	1.05	0.01
Average	159.8	3654.4	1.526	3726.3	1.02	0.01

Table 18. Comparison of mixed integer programming with the heuristic method ($n = 2$ and $m = 20$)

No.	Mixed integer programming		Heuristic method		Performance	
	Computational time (A)	Objective value (B)	Computational Time (C)	Objective value (D)	D/B	C/A
1	900	4943	3.65	5057	1.023	0.004
2	1526	4889	3.65	4914	1.005	0.002
3	1774	4546	3.65	4751	1.045	0.002
4	1421	4643	2.72	4643	1.000	0.002
5	527	4685	2.52	4830	1.031	0.005
6	439	4536	3.65	4644	1.024	0.008
7	1034	5202	2.55	5384	1.035	0.002
8	1702	4918	2.12	4918	1.000	0.001
9	1528	4964	2.12	5074	1.022	0.001
10	1447	4612	2.53	4734	1.026	0.002
Average	1229.8	4793.8	2.916	4894.9	1.021	0.003

7. CONCLUSIONS

This paper discusses how to assign container-delivery tasks to AGVs during ship operations in automated container terminals. The dual-cycle operation is analyzed for the loading and discharging operations. The dispatching problem is formulated as an assignment problem when any delay of the operation of a QC is not allowed. And, a mixed integer linear programming model is suggested for the case when the delay of an operation by a QC is allowed, but with a high penalty cost. The primary objective is assumed to be minimizing the completion time of a QC, and the secondary objective is minimizing the total travel time of AGVs. A heuristic method is suggested for the latter case in order to overcome the long computational time required by the MIP model. It was found that the objective value of the heuristic method does not exceed the optimal objective value by more than 3% on average, but the computational time of the heuristic method was less than 4 seconds, which may be considered short enough to be used in real practice.

The results of this study are expected to contribute to the automated operation of container terminals and the development of operation technologies for other large-sized automated material handling systems.

REFERENCES

[1] Anwar, M. F. and R. Nagi, "Integrated scheduling of material handling and manufacturing activities for just-in-time production of complex assemblies," *International Journal of Production Research* 36 (1998), 653-681.

[2] Durrant-Whyte, Hugh F., "An Autonomous Guided Vehicle for Cargo Handling Applications," *International Journal of Robotics Research* 15 (1996), 407-440.

[3] Egbelu, P. J. and J. M. A. Tanchoco, "Characterization of automatic vehicle dispatching rules," *International Journal of Production Research* 22 (1984), 359-374.

[4] Egbelu, P. J., "Pull versus push strategy for automated guided vehicle load movement in a batch manufacturing system," *Journal of Manufacturing Systems* 6 (1987), 209-221.

[5] Evans, J. R. and E. Minieka, *Optimization Algorithms for Networks and Graphs*, Marcel Dekker, New York 1992.

[6] Evers, Joseph J. M. and Stijn A. J. Koppers, "Automated guided vehicle traffic control at a container terminal," *Transportation Research - A* 30 (1996), 21-34.

[7] Ihsan, S. and Don L. Hommertzheim, "Dynamic dispatching algorithm for scheduling machines and automated guided vehicles in a flexible manufacturing system," *International Journal of Production Research* 30 (1992), 1059-1079.

[8] Klein, C. M. and J. Kim, "AGV dispatching," *International Journal of Production Research* 34 (1996), 95-110.

[9] Maxwell, W. L. and J. A. Mukstadt, "Design of automated guided vehicle system," *IIE Transactions* 14 (1982), 114-124.

[10] Russell, R. and J. M. A. Tanchoco, "An evaluation of vehicle dispatching rules and their effect on shop performance," *Material Flow* 1 (1984), 271-280.

[11] Yim, D. and R. J. Linn, "Push and pull rules for dispatching automated guided vehicles in a flexible manufacturing system," *International Journal of Production Research* 31 (1993), 43-57.