A dispatching method for automated lifting vehicles in automated port container terminals

Vu Duc Nguyen, Kap Hwan Kim *

Department of Industrial Engineering, Pusan National University, 30 Jangjeon-dong, Geumjeong-gu, Busan 609-735, South Korea

Available online 17 September 2008

Abstract

In automated container terminals, containers are transported from the marshalling yard to a ship and vice versa by automated vehicles. The automated vehicle type studied in this paper is an automated lifting vehicle (ALV) that is capable of lifting a container from the ground by itself. This study discusses how to dispatch ALVs by utilizing information about pickup and delivery locations and time in future delivery tasks. A mixed-integer programming model is provided for assigning optimal delivery tasks to ALVs. A procedure for converting buffer constraints into time window constraints and a heuristic algorithm for overcoming the excessive computational time required for solving the mathematical model are suggested. Numerical experiments are reported to compare the objective values and computational times by a heuristic algorithm with those by an optimizing method and to analyze the effects of dual cycle operation, number of ALVs, and buffer capacity on the performance of ALVs.

© 2008 Elsevier Ltd. All rights reserved.

Keywords: Automated lifting vehicle; Container port terminal; Dispatching; Time window

1. Introduction

At container terminals, a ship operation process consists of an unloading operation, during which containers in a containership are unloaded from a containership and stacked in a marshalling yard, and loading operation, during which containers are handled in a reverse direction of an unloading operation. In this study, we consider an automated port container terminal in which three types of equipments, such as quay cranes (QCs), ALVs, and automated yard cranes (AYCs), are used for ship operations.

During an unloading operation, a container picked up by a QC is put down on buffer space under the QC in an apron. Then, an ALV picks up and delivers it to the marshalling yard. In the marshalling yard, the ALV releases a container on the buffer space at a transfer point (TP) of the yard. An AYC picks up and stacks it onto an empty slot in a bay. A loading operation is performed as the reverse order of an unloading process. Fig. 1 illustrates a layout of an automated port container terminal that has buffer space at QCs and TPs in an apron and yard.

* Corresponding author. Tel.: +82 51 510 2419; fax: +82 51 512 7603.
E-mail addresses: duc.isepusan.ac.kr (V.D. Nguyen), kapkim@pusan.ac.kr (K.H. Kim).

0360-8352/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
The handling activities performed by QCs are called “seaside operations,” while those performed by ALVs and AYCs are called “landside operations.” We define the task of an ALV in ship operations as delivering a container from an apron to a yard in an unloading operation or from a yard to an apron in a loading operation.

Before ship operations actually begin, a work schedule for QCs is constructed first based on a stowage plan that is sent as a guideline for unloading and loading operations by a shipping agent. Then, based on the work schedule, a sequence list is made that specifies the sequence of unloading and loading operations for individual containers. Actual ship operations are usually carried out in the same order as specified in a sequence list. Thus, it can be said that the delivery tasks of ALVs are known in advance and the sequence of them is also predetermined.

Dispatching can be defined as the assignment of ALVs to delivery tasks. In this paper, we consider certain buffer space in aprons and yards on which QCs, AYCs, and ALVs can release containers. QCs or AYCs can pick up (put down) containers directly from (onto) an ALV. However, because ALVs are also capable of lifting a container from the ground by itself (Fig. 2), a QC can put a container on the buffer space if the buffer space is available. Therefore, for releasing a container on buffer space, a QC or an ALV must wait when the buffer space is full. To improve the productivity and reduce delays in a ship operation, it is important to simultaneously minimize the delay time of QCs and travel time of ALVs.

### Notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>the capacity of buffer</td>
</tr>
<tr>
<td>$m$</td>
<td>the number of tasks for the QC</td>
</tr>
<tr>
<td>$q_j$</td>
<td>the number of container inventories at period $j$ when the inventory plans up to the event $(j - 1)$ are considered</td>
</tr>
<tr>
<td>$W_i$</td>
<td>the time window for task $i$</td>
</tr>
<tr>
<td>$L(W_i)$</td>
<td>the lower bound of time window $W_i$</td>
</tr>
<tr>
<td>$U(W_i)$</td>
<td>the upper bound of time window $W_i$</td>
</tr>
</tbody>
</table>

Fig. 1. Layout of an automated container terminal.
In summary, the dispatching problem regarded in this study has the following three special properties:

1. Tasks must be carried out in the exact same order as that in an unloading and loading sequence list that are made by planners in a terminal.
2. The objective of minimizing delays in the operation of QC cons has a higher priority than the objective of minimizing the total travel time of ALVs. This means that a higher priority is given to the operation of QC cons. Because a QC is much more expensive than an ALV or an AYC, QC cons are usually a bottleneck resource in port container terminals.
3. If a QC had a delay in a transfer operation, then all succeeding transfer operations in the same QC will be delayed by the same amount of time, because the sequence list of unloading and loading operations is predetermined.

Most previous studies have focused on AGV dispatching methods, and assumed that pickup calls are issued randomly and that the sequence of calls cannot be known in advance. Thus, the dispatching decision is made after a pickup call is issued, or when a vehicle becomes free from a previous delivery task. Egbelu (1987) suggested a demand-driven rule in which AGVs are first dispatched to delivery tasks whose destinations are input buffers whose lengths are below a threshold value. Bilge and Ulusoy (1995) provided a simultaneous scheduling method for the operation of machines and transfer of materials by AGVs. Kim, Tanchoco, and Koo (1999) suggested an AGV dispatching method in which the balancing work loads among different workstations is the first criterion for selecting the next delivery task. Lim, Kim, Yoshimoto, Lee, and Takahashi (2003) introduced an AGV dispatching method using a bidding concept in that the dispatching decisions are made through communication among related vehicles and machines. Briskorn, Drexl, and Hartmann (2006) presented an alternative formulation of the AGV assignment problem that does not include due time and is based on a rough analogy to inventory management, and proposed an exact algorithm for solving the formulation.

A few previous researches have concerned in ALVs. Van der Meer (2000) evaluated various dispatching rules, including rules using pre-arrival information, for ALVs in container terminals by using a simulation. Vis and Harika (2004) and Yang, Choi, and Ha (2004) compared the performance of two types of automated vehicles, namely AGV and ALV, by a simulation study.

In the perspective of scheduling problems, this study is similar to an AGV dispatching problem in the paper of Kim and Bae (2004) in that there have been no buffer space; however, this study have considered buffer space for the container at QC cons and TPs in an apron and yard. Another similar problem is a multiple traveling salesmen problem with precedence constraints and time windows in the paper of Dumas, Desrosiers, and Solomon (1995).

The organization of this paper is as follows. The relationship between the ALV dispatching and ship operations in automated container terminals is introduced, and the mixed-integer programming model for the ALV dispatching is suggested in Section 2. In Section 3, a procedure for converting buffer constraints into time window constraints is presented and a heuristic algorithm is suggested for solving the dispatching problem with time window constraints. Section 4 shows the numerical experiments to compare a heuristic algorithm with an optimizing method and to analyze the effects on the heuristic algorithm. Finally, Section 5 gives some concluding remarks.
2. Problem definition and model formulation

The following assumptions are introduced in this study:

(1) A pooling strategy for ALVs is applied to a dispatching process. That is, an ALV can deliver a container for any QCs, which means that no ALVs are dedicated to one QC.
(2) All ALVs are the same and transfer one container at a time.
(3) Time spent by ALVs at the transfer points of blocks is known. To consider the waiting time of ALVs for the transfer of containers by an AYC, the detail operation schedule of the AYC must be known that is another big problem and makes it a much more complicated problem. AYCs are not usually recognized as a bottleneck resource in container terminals. Thus, we can assume that the waiting time of ALVs at the transfer points in the marshalling yard is known or constant.
(4) In a dual cycle, the time between the end of the release of a container by a QC (or ALV) in buffer space and the beginning of the pickup of another container by an ALV (or QC) from buffer space is negligible.
(5) Pick up and release time of a container by a QC can be neglected.
(6) Congestions in ALVs on guide paths are not considered.

2.1. Problem definition

A loading operation cycle by a QC begins with the pickup of a container from the buffer space at the QC in an apron, while an unloading operation cycle ends with the release of a container onto the buffer space at the QC in an apron. Table 1 shows an example of a sequence list applied in ship operations for the QC 1, which performs the ship operations in a dual cycle manner. The progress of operations by the QC 1, which performs the operations in Table 1, can also be represented in Table 2.

Table 2 illustrates events and their notations in the ship operation of QC 1. The earliest event times in the last column of Table 1 are times when the QC transfers a container to/from an ALV under the assumption that the QC operations are performed without any interruptions and delays. For a QC to load a container without delay, an assigned ALV must put down a loading container into a specific buffer at the QC in an apron before the QC begins to pick up this container. For a QC to release an inbound container onto the floor, a buffer space for an inbound container at the QC must be available or an assigned ALV must be ready for receiving an inbound container.

2.2. Model formulation

Let $q^k_i$ be an event representing the moment that QC $k$ transfers the $i$th container (the $i$th operation of QC $k$). When the $i$th operation of QC $k$ is a loading operation, an event $q^k_i$ corresponds to the pickup of the $i$th container by QC $k$ from the buffer at QC $k$. When the $i$th operation of QC $k$ is an unloading operation, it

<table>
<thead>
<tr>
<th>Task sequence</th>
<th>Type$^a$</th>
<th>Ship location$^b$</th>
<th>Yard location$^c$</th>
<th>Operation cycle time</th>
<th>Earliest event time</th>
</tr>
</thead>
<tbody>
<tr>
<td>QC 1 (buffer capacity = 2)</td>
<td>L</td>
<td>11/03/04</td>
<td>A/21/3/2</td>
<td>120</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>U</td>
<td>11/04/10</td>
<td>A/17/4/2</td>
<td>110</td>
<td>230</td>
</tr>
<tr>
<td>2</td>
<td>L</td>
<td>11/03/06</td>
<td>A/07/4/1</td>
<td>120</td>
<td>460</td>
</tr>
<tr>
<td>3</td>
<td>U</td>
<td>11/04/08</td>
<td>A/21/3/1</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

$^a$ L, loading; U, unloading.
$^b$ Ship bay no./row no./tier no.
$^c$ Yard block/yard-bay no./row no./tier no.
Table 2
Events of the ship operation of QC 1

<table>
<thead>
<tr>
<th>Event sequence</th>
<th>Time</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>$q_1^i$</td>
<td>QC 1 picks up the 1st container from the buffer at QC 1 (assume that this container was ready in the buffer at QC 1)</td>
</tr>
<tr>
<td>2</td>
<td>120</td>
<td>–</td>
<td>QC 1 releases the 1st container in the ship’s bay, and picks up the 2nd container from the ship’s bay</td>
</tr>
<tr>
<td>3</td>
<td>230</td>
<td>$q_1^i$, $q_3^i$</td>
<td>QC 1 releases the 2nd container into the buffer (at this time, the buffer space must be available), and picks up the 3rd container from the buffer at QC 1 (assume that this container was ready in the buffer at QC 1)</td>
</tr>
<tr>
<td>4</td>
<td>350</td>
<td>–</td>
<td>QC 1 releases the 3rd container in the ship’s bay, and picks up the 4th container from the ship’s bay</td>
</tr>
<tr>
<td>5</td>
<td>460</td>
<td>–</td>
<td>$q_1^i$ QC 1 releases the 4th container into the buffer at QC 1 (at this time, the buffer space must be available)</td>
</tr>
</tbody>
</table>

The problem of this paper is addressed as a static scheduling problem where ALVs must be assigned to complete all tasks in a known set under a limited buffer capacity. A feasible dispatching decision is made as a one-to-one assignment between all events in $S \cup T$ and those in $D \cup T$.

Let $K' = \{O\} \cup K$, $K'' = \{F\} \cup K$, and $x_{ij}^l$ be a decision variable that becomes 1 if $a_{ij}$ is assigned to $a_j$ for $k \in K'$ and $l \in K''$. For $k$ and $l \in K$, the assignment of $a_{ij}$ to $a_j$ implies that the ALV, which have just delivered the $i$th container of QC $k$, is scheduled to deliver the $j$th container of QC $l$.
Let $a$ be the travel cost per unit time of an ALV, and $\beta$ be the penalty cost per unit time for the delay in the completion time. It is assumed that $a \ll \beta$. Also, let $m_O$ and $m_F$ equal to |$V$|. Then, the dispatching problem can be formulated as follows.

Minimize $a \sum_{k \in K'} \sum_{i=1}^{m_k} \sum_{j=1}^{m_k} x_{ij}^{kl} + \beta \sum_{k \in K} (y^k m_k - s^k m_k)$ \hfill (1)

Subject to

\[ \sum_{i=1}^{m_k} x_{ij}^{kl} = 1, \quad \forall k \in K', \quad i = 1, \ldots, m_k \] \hfill (2)

\[ \sum_{k = l}^{m_k} x_{ij}^{kl} = 1, \quad \forall i \in K'', \quad j = 1, \ldots, m_l \] \hfill (3)

\[ z^k_j - (z^k_i + c^j_{kl}) \geq M(x_{ij}^{kl} - 1), \quad \forall k \in K', \quad i \in K, \quad i = 1, \ldots, m_k, \quad j = 1, \ldots, m_l \] \hfill (4)

\[ y^k_{i+1} - y^k_i \geq s^k_{i+1} - s^k_i, \quad \forall k \in K, \quad i = 1, \ldots, m_k - 1 \] \hfill (5)

\[ y^k_i \geq s^k_i, \quad \forall k \in K', \quad i = 1, \ldots, m_k \] \hfill (6)

\[ y^k_i \leq z^k_i, \quad \forall k \in K, \quad i \in U^k \] \hfill (7)

\[ y^k_i \geq z^k_i, \quad \forall k \in K', \quad i \in L^k \] \hfill (8)

\[ u^i_{tk} - \sum_{i=1}^{m_k} u^i_{tk} + \sum_{i=1}^{m_l} v^i_{tk} - \sum_{i=1}^{m_l} v^i_{tk} \leq b_k \quad \forall k \in K, \quad t \in \{y^k, \forall i \in U^k; z^k, \forall i \in L^k \} \] \hfill (9)

\[ t - y^k_i < M u^i_{tk}, \quad \forall k \in K, \quad i \in U^k, \quad t \in \{y^k\} \] \hfill (10)

\[ t - y^k_i > M(u^i_{tk} - 1), \quad \forall k \in K, \quad i \in L^k, \quad t \in \{y^k\} \] \hfill (11)

\[ t - z^k_i < M u^i_{tk}, \quad \forall k \in K, \quad i \in L^k, \quad t \in \{z^k\} \] \hfill (12)

\[ t - z^k_i > M(u^i_{tk} - 1), \quad \forall k \in K, \quad i \in U^k, \quad t \in \{z^k\} \] \hfill (13)

\[ u^i_{tk}, v^i_{tk} = 0 \text{ or } 1, \quad \forall k \in K, \quad i = 1, \ldots, m_k, t \in \{y^k, z^k\} \] \hfill (14)

\[ x_{ij}^{kl} = 0, \quad 1, \quad \forall k \in K', \quad l \in K'', \quad i = 1, \ldots, m_k, \quad j = 1, \ldots, m_l \] \hfill (15)

Note that $s^k_0 = 0$ for $\forall i \in V$.

Objective function (1) is to minimize the total travel time of ALVs and the total delays in QC operations. Because $a \ll \beta$, the sum of delays in QC operations will be minimized first. For the same value of the total delays, the total travel distance of the ALVs will be minimized. Constraints (2) and (3) force the one-to-one assignment between all events in $S \cup T$ and those in $D \cup T$. Constraint (4) implies the fact that two events that are served consecutively by the same ALV must be set apart by at least the time required by an ALV for traveling and transferring between the two events. Constraint (5) implies that two events that are served by the same QC must be set apart by at least the time required for the QC to perform all movements between the two events. Constraint (6) represents that the actual event time is always greater than or equal to the earliest possible event time. Constraints (7) and (8) show that the $i$th container in the buffer at QC $k$ must be available for picking up by an ALV in an unloading operation, and by the QC in a loading operation, respectively. Constraint (9) represents that the number of containers of the buffer at QC $k$ released or picked up by the QC or ALVs at time $t$ cannot exceed the capacity of the buffer. Constraint (10) ensures that $u^i_{tk}$ is equal to 1 when $t > y^k_i$, which means that the $i$th container has already been released at time $t$ by QC $k$ into the buffer at QC $k$. Constraint (11) ensures that $u^i_{tk}$ is equal to 1 when $t > y^k_i$, which means that the $i$th container has already been picked up at time $t$ by QC $k$ from the buffer at QC $k$. Constraint (12) ensures that $u^i_{tk}$ is equal to 1 when $t > y^k_i$, which means that the $i$th container is released by an ALV into the buffer at QC $k$. Constraint (13) ensures that $u^i_{tk}$ is equal to 1 when $t > z^k_i$, which means that the $i$th container has already been picked up by an ALV at time $t$ from the buffer at QC $k$.

In a model formulation process, the decision variable, $x_{ij}^{kl}$, do not have an index for identifying a specific ALV. However, for a given set of values of $x_{ij}^{kl}$, it can be interpreted as an assignment of delivery tasks to specific individual ALVs as follows. Consider an example of 2 QCs, 2 ALVs, and 2 tasks for each QC. Suppose that a solution ($x_{11}^{12} = x_{12}^{12} = x_{21}^{22} = x_{22}^{22} = x_{12}^{22} = x_{22}^{22} = 1$ and all the others $x_{ij}^{kl} = 0$) was obtained.
A feasible solution can be represented graphically as shown in Fig. 3 in which each arc corresponds to $x_{ij}^k$ with the value of 1. The set $S \cup T$ includes the starting events (initial location) of ALVs ($S$) and events related to the transfer operations by ALVs ($T$). While the set $D \cup T$ includes the stopping events (final destination) of ALVs ($D$) and events related to the transfer operations by ALVs ($T$). An ALV that started travel from its initial location ($S$) or completed a delivery for a transfer operation ($T$) can be assigned to another transfer operation ($T$) or its final destination ($D$). Therefore, nodes in $S \cup T$ can be considered as supply sources of ALVs, and nodes in $D \cup T$ as demand sources of ALVs. A feasible solution of $x_{ij}^k$ is a one-to-one assignment from a node in $S \cup T$ to a node in $D \cup T$.

In Fig. 3, the solid arc shows the assignment for ALV 1, while the dotted arc is for ALV 2. By the solid arc, node $a_{1O}$ in the supply source is connected to the node $a_{11}$ in the demand source, which implies that ALV 1 starts the empty travel to the pick up position of the task related to the 1st operation of QC 1 and performs the loaded travel for the task – and then the node $a_{11}$ in the supply source is connected to the node $a_{21}$ in the demand source, which implies that after ALV 1 completes the task related to the 1st operation of QC 1, it performs the task related to the 2nd operation of QC 2 and finally the node $a_{21}$ in the supply source is connected to the node $a_{F1}$ in the demand source, which can be interpreted as that ALV 1 stops traveling after it completes all assigned tasks. In a similar way, the solution for ALV 2 can be interpreted as follows. ALV 2 delivers a container for the 1st operation of QC 2, and then another container for the 2nd operation of QC 1, and finally stops the travel.

The problem in this paper is NP-hard. Formulations (1)–(15) are a scheduling problem with precedence constraints and buffer constraints. A problem similar to problems (1)–(15) is the multiple traveling salesmen problem with precedence constraints and time windows. Both the problems must be NP-hard. Thus, a heuristic algorithm is suggested in the following section for solving this problem.

3. A heuristic algorithm

In this section, we introduce a heuristic algorithm for solving the formulation in the previous section. Formulations (1)–(15) are addressed as a scheduling problem with precedence constraints and buffer constraints. First, a procedure for converting buffer constraints in the formulation into time window constraints is presented in Section 3.1. Also, a heuristic algorithm for solving this scheduling problem with precedence constraints and time window constraints is suggested in Section 3.2.

3.1. Procedure for converting buffer constraints into time window constraints

A general procedure for converting buffer constraints into time window constraints in a formulation is suggested in this section. At the initial stage, we suppose that $y_i^k$’s are equal to $s_i^k$’s, for $i = 1, \ldots, m_k$, and $k = 1, \ldots, |K|$, and events are sequenced in the increasing order of $y_i^k$, for $k = 1, \ldots, |K|$. We will denote the $j$th event in the sequence as “event $(j)$”, the event time of event $(i)$ as $y_i$ for operations of a QC and $z_i$ for operations of ALVs visiting the QC.

Determining the time window for an unloading and loading task is to determine the latest allowable pickup time of the inbound container and the earliest allowable release time of the outbound container on the buffer space by a transporter, respectively. A time axis is divided into multiple periods each of which corresponds to the time interval between event times.
The calculation of the upper and the lower bounds of time windows is carried out from the first event to the last event. For an unloading task, the time window begins from the moment of the release of an inbound container by a QC onto the buffer space in an apron. The first succeeding period is examined to check whether the period can be included into the time window (the feasibility of the extension). Next, the next period will be examined. This process will be repeated until we reach a period that cannot be included into the time window due to the limitation in the buffer space. In the process of checking the feasibility of the extension, changes in the number of container inventories on the buffer by the following four events must be monitored: discharging inbound containers by the QC; picking up outbound containers by the QC; picking up inbound containers by transporters; and delivering outbound containers by transporters. Time of the first two events is already known from the unloading and loading schedule of the QC. Because these time windows are constructed in a chronological order, time of the third events that is related to the unloading events for which time windows were already scheduled are also known. Time for the last events that is related to the loading events for which time windows were not scheduled, are not known and so they must be estimated for calculating the time window. Time for a delivery of an outbound container by a transporter is estimated in a way that the length of time windows of the current unloading task becomes equal to that of the corresponding loading task in the unit of the number of periods.

For an unloading task \( i \), the upper bound of time window is extended from \( y_i \) to the positive direction on the time axis. For loading task \( i \), the lower bound of time window is extended from \( y_i \) to the negative direction on the time axis. In this process, all time values that are necessary for determining time windows are known.

Period \( i \) is defined to be the time interval between event \((i - 1)\) and \( i \). In the following, the procedure to evaluate time windows is described.

Notations

\( b \) the capacity of buffer
\( m \) the number of tasks for the QC
\( q_j \) the number of container inventories at period \( j \) when the inventory plans up to the event \((j - 1)\) are considered
\( W_i \) the time window for task \( i \)
\( L(W_i) \) the lower bound of time window \( W_i \)
\( U(W_i) \) the upper bound of time window \( W_i \)

A procedure for converting buffer constraints into time window constraints for a QC

**Step 0. (Initializing)** Set \( y_i = s_i \) and \( W_i = \emptyset \), for \( i = 1, \ldots, m \). And set \( q_i = 0 \), for \( i = 1, \ldots, m \). Start \( \xi = 0 \).

**Step 1. (Next task)** \( \xi = \xi + 1 \). If \( \xi > m \), then stop. Otherwise, if event \( \xi \) is loading task, then go to Step 2, and if event \( \xi \) is unloading task, then go to Step 3.

**Step 2. (Extending the lower bound for the loading task)**

Start \( i = \xi \).

Repeat

\[ q_i = q_i + 1 \] and \( W_\xi = W_\xi \cup \left[ y_{i-1}, y_i \right] \) (add period \( i \) to the time window for event \( \xi \)).

\( i = i - 1 \).

Until \( q_i = b \) or \( i = 0 \).

Go to Step 1.

**Step 3. (Extending the upper bound for the unloading task)**

(Set tentative arrival times for loading containers)

Start \( k = \xi + 1 \)

Repeat

If event \( k \) is a loading task, then set the tentative lower bound of the time window for event \( k \) to be

\[ s = \left( \frac{\xi}{k} \right) / 2 \] (For example, if \( s = 3 \), then the tentative lower bound is set to be the end of period 3)
In Fig. 4, the above procedure is illustrated by using an example of ten loading and unloading operations. For the first task, \( L_1 \) (loading task), we extend time window by using Step 2 in the procedure. Then, it is easy to get the time window of \( L_1 \), \( W_1 = [0, y_1] \) (as shown in Fig. 4a). For the second task, \( U_2 \) (unloading task), we extend time window by using Step 3 as follows: firstly, we estimate the period numbers corresponding the lower bounds of the time window for loading tasks (expected arrival times of loading containers) succeeding \( U_2 \), and denote it with an apostrophe ('). For example, we consider \( L_3 (\xi = 2, k = 3, s = 2.5) \), the tentative lower bound of the time window for \( L_3 \), which is represented by \( L_3' \), is 2.5, indicating the middle of period 3. And, in a similar way, the tentative lower bound for \( L_8' \) and \( L_{10}' \) are 5 and 6, respectively (as shown in Fig. 4b). Then, we start to extend the upper bound of the time window for \( U_2 \) one by one period. For period 3, the inventory becomes two: one arrival of an inbound container for \( U_2 \) and one arrival of an outbound container for \( L_3 \). For period 4, we have one departure for \( L_3 \) and the inventory is reduced to one. For period 5, the inventory is increased to two because of one arrival of an inbound container for \( U_4 \). Thus, the upper bound of the time window for \( U_2 \) can be extended to the end of period 5. For period 6, because \( L_8' \) and \( U_5 \) are located at the beginning of period 6, one outbound container for \( L_8 \) and one inbound container for \( U_5 \) may arrive at the buffer and the inventory level may reach to 4 if the upper bound for \( U_2 \) is extended to period 6. Thus, the time window of \( U_2 \) becomes \( W_2 = [y_2, y_5] \).

For the third task, \( L_3 \) (loading task), the calculation is similar to that of \( L_1 \). A time window is extended to the left direction one by one period as far as the buffer capacity allows. Thus, the time window of \( L_3 \) becomes \( W_3 = [0, y_3] \) (as shown in Fig. 4c). For the fourth task, \( U_4 \) (unloading task), the calculation is similar to that of \( U_2 \). Firstly, we estimate the tentative lower bounds for loading tasks succeeding \( U_4 \). For \( L_8(\xi = 4, k = 8, s = 6) \), the tentative lower bound of the time window for \( L_8' \) is 6. And, in the same way, values of \( s \) for \( L_{10}' \) is 7 (as shown in Fig. 4d). Then, we start extending time window one by one period. For period 5, the inventory becomes two: one from the arrival for \( U_2 \) and one for \( U_4 \). For period 6, we have one departure of an inbound container for \( U_2 \) and one arrival of an outbound container for \( U_5 \), so the inventory remains two. For period 7, then, if we extend the upper bound of the time window for \( U_4 \) by one more period, then the inventory becomes to exceed the limitation of 2 because of additional arrivals for \( U_6 \) and \( L_8 \). Thus, the final time window of \( U_4 \) becomes \( W_4 = [y_4, y_6] \).

This process is repeated until the time windows for all tasks are determined. Time windows of remaining tasks are, as shown in Fig. 4e, \( W_5 = [y_5, y_6] \), \( W_6 = [y_6, y_7] \), \( W_7 = [y_7, y_9] \), \( W_8 = [y_6, y_8] \), \( W_9 = [y_9, \infty] \), and \( W_{10} = [y_8, y_{10}] \).

In this way, the constraint on the buffer space, (9)-(14), for \( z_i^k \) was converted into the constraint of the feasible time window \( W_i^k \) as follows:

\[
\begin{align*}
    z_i^k & \in W_i^k = [y_i^u, y_i^v], \quad u \leq i, \ u \in L^k \cup U^k, \ i \in L^i, \ k \in K & \quad (16) \\
    z_i^k & \in W_i^k = [y_i^u, y_i^v], \quad i \leq v, \ v \in L^k \cup U^k, \ i \in U^i, \ k \in K & \quad (17)
\end{align*}
\]

3.2. Heuristic algorithm

After applying constraints on the buffer, (9)-(14), formulations are converted into the constraint of time windows; the formulation becomes a scheduling problem with precedence and time window constraints. A heuristic algorithm for solving this scheduling problem is suggested as follows: in the initial stage, we assume that \( y_i^k \) which are equal to \( x_i^k \), \( k = 1, \ldots, |K|, \ i = 1, \ldots, m_i \), are given. Also, let \( z_i^k \) be set to \( (LW_i^k) \). Then, events are sequenced in an increasing order of \( y_i^k \). We attempt to match a vehicle with the first QC event.
**ETQC: Event time of the QC ($y_i^E$)**

**TLBL: Tentative lower bound of time window for loading container**

(a) Time window of the first task

<table>
<thead>
<tr>
<th>Period</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETQC</td>
<td>L1</td>
<td>U2</td>
<td>L3</td>
<td>U4</td>
<td>U5</td>
<td>U6</td>
<td>U7</td>
<td>L8</td>
<td>U9</td>
<td>L10</td>
<td></td>
</tr>
<tr>
<td>TLBL</td>
<td></td>
<td></td>
<td>L3'</td>
<td></td>
<td></td>
<td>L8'</td>
<td>L10'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Containers on buffer</td>
<td>1</td>
<td></td>
<td>L3</td>
<td>U4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>L1</td>
<td>U2</td>
<td>U2</td>
<td>U2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Time window of the first two tasks

<table>
<thead>
<tr>
<th>Period</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETQC</td>
<td>L1</td>
<td>U2</td>
<td>L3</td>
<td>U4</td>
<td>U5</td>
<td>U6</td>
<td>U7</td>
<td>L8</td>
<td>U9</td>
<td>L10</td>
<td></td>
</tr>
<tr>
<td>TLBL</td>
<td></td>
<td></td>
<td>L3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L8'</td>
<td>L10'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Containers on buffer</td>
<td>1</td>
<td></td>
<td>L3</td>
<td>L3</td>
<td>L3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>L1</td>
<td>U2</td>
<td>U2</td>
<td>U2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(c) Time window of the first three tasks

<table>
<thead>
<tr>
<th>Period</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETQC</td>
<td>L1</td>
<td>U2</td>
<td>L3</td>
<td>U4</td>
<td>U5</td>
<td>U6</td>
<td>U7</td>
<td>L8</td>
<td>U9</td>
<td>L10</td>
<td></td>
</tr>
<tr>
<td>TLBL</td>
<td></td>
<td></td>
<td>L3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L8'</td>
<td>L10'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Containers on buffer</td>
<td>1</td>
<td></td>
<td>L3</td>
<td>L3</td>
<td>L3</td>
<td>U4</td>
<td>U4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>L1</td>
<td>U2</td>
<td>U2</td>
<td>U2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(d) Time window of the first four tasks

<table>
<thead>
<tr>
<th>Period</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETQC</td>
<td>L1</td>
<td>U2</td>
<td>L3</td>
<td>U4</td>
<td>U5</td>
<td>U6</td>
<td>U7</td>
<td>L8</td>
<td>U9</td>
<td>L10</td>
<td></td>
</tr>
<tr>
<td>TLBL</td>
<td></td>
<td></td>
<td>L3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L8</td>
<td>L8</td>
<td>L10</td>
<td>L10</td>
</tr>
<tr>
<td>Containers on buffer</td>
<td>1</td>
<td></td>
<td>L3</td>
<td>L3</td>
<td>L3</td>
<td>U4</td>
<td>U4</td>
<td>L8</td>
<td>L8</td>
<td>L10</td>
<td>L10</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>L1</td>
<td>U2</td>
<td>U2</td>
<td>U2</td>
<td></td>
<td></td>
<td>U5</td>
<td>U6</td>
<td>U7</td>
<td>U9</td>
</tr>
</tbody>
</table>

(e) Time window of all tasks

Fig. 4. Converting buffer constraints into time window constraints.
And then, we match vehicles with the first two events, and then with the first three events, and so on. In the procedure, if there exists a QC event to which a vehicle cannot be assigned, then we delay an ALV event time until every candidate QC event is assigned to a vehicle. If every QC event is not assigned to a vehicle even after delays of AVL events up to the upper bound of the time windows, then QC event times will be delayed. This process is continued until all the QC events are assigned to vehicles. Finally, the assignment problem is solved to minimize the total travel time.

Denote the jth event in the sequence as event (j), the event time of event (i) as \( y_j \), for operation of QC and \( z_i \) for operation of ALVs, the time required for an ALV to be ready for event (j) after it experiences event (i) as \( c_{ij} \) (which corresponds to the notation of \( c_{ij}^U \)), the pure travel time from the location of event (i) to the location of event (j) as \( t_{ij} \), the decision variable for the assignment of event (i) to event (j) as \( x_{ij} \), and the time window of \( z_i \) as \( W_i \).

Let \( T_L \) be a subset of T which includes only the first \( \xi \) events in the sequence, and \( L_\xi \) and \( U_\xi \) be subsets of \( T_\xi \) which includes only loading and unloading tasks, respectively. Then, the constraint subset \( \xi \) of (2)–(4) and (16) and (17) can be written as follows:

**Constraint subset \( \xi \)**

\[
\sum_{j \in S \cup T_\xi} x_{ij} = 1, \quad \forall i \in D \cup T_\xi \quad (18)
\]

\[
\sum_{j \in D \cup T_\xi} x_{ij} = 1, \quad \forall i \in S \cup T_\xi \quad (19)
\]

\[
z_i - (z_i + c_{ij}) \geq M(x_{ij} - 1), \quad \forall i \in S \cup T_\xi, \quad j \in T_\xi \quad (20)
\]

\[
z_i \in W_i, \quad \forall i \in L_\xi \cup U_\xi \quad (21)
\]

\[
x_{ij} = 0 \text{ or } 1, \quad \forall i \in S \cup T_\xi, \quad j \in D \cup T_\xi \quad (22)
\]

Then, the following algorithm assigns ALVs to tasks.

**A heuristic solution procedure**

**Step 0. Initializing.** Set \( y_i = s_i, \quad i = 1, \ldots, m_k, \quad k \in K \). Also, evaluate the time window for each \( s_i \).

\[
z_i^k = L(W_i^k), \quad \forall i \in L^k \cup U^k, \quad \text{and} \quad z_i = U(W_i^k), \quad \forall i \in V. \]

Sequence the events in the increasing order of \( y_i \). Denote the jth event time in the sequence as \( y_j \) for QC operations and \( z_i \) for ALV operations. Set \( \xi = 0 \).

**Step 1. Next task.** \( \xi = \xi + 1 \). If \( \xi > m \) (the total number of tasks in sequence \( T \)), then go to Step 5. Otherwise, sequence the events in the increasing order of \( y_i \). And let \( y_{\xi} \) correspond to \( y_{\xi}^j \) (the \( \lambda \)th event time of QC \( \gamma \)). Set \( z_j = L(W_j) \) for all \( j \in L_{\xi} \cup U_{\xi} \). Go to Step 2.

**Step 2. Feasibility checking.**

**Step 2.1.** Check the existence of a feasible solution to constraints subset \( \xi \), (18)–(20). If there is a feasible solution, then go to Step 2.2. Otherwise, go to Step 3.1.

**Step 2.2.** Check whether the event time of event (\( \xi \)) for ALVs, \( z_{\xi} \), satisfies constraint subset \( \xi \), (21).

If it was satisfied, then go to Step 4. Otherwise, go to Step 3.2.

**Step 3. Delaying event time.**

**Step 3.1.** Let \( \pi_{\xi} = \min_{i \in S \cup T_{\xi} + 1} \{z_i - (z_i - z_{\xi})\} \). Then, \( z_{\xi} = z_{\xi} + \pi_{\xi} \). Go to Step 2.1.

**Step 3.2.** Let \( z_{\xi U} \) correspond to \( y_{\xi U} \). Let \( \pi_{\xi} = z_{\xi} - z_{\xi U} \). Then, set \( y_n^\sigma = y_n^\sigma + \pi_{\xi} \), for \( n \geq \sigma \) (Note \( \sigma = \lambda \) for \( \xi \in L_{\xi} \)). Also, update \( z_n^\sigma = z_n^\sigma + \pi_{\xi} \), for \( n > \lambda \) (Note \( z_n^\sigma \neq 0 \)), and set \( z_{nU} = z_{nU}^\sigma \), for \( n \geq \lambda \). Go to Step 2.2.

**Step 4. Time window adjustment.**

Let \( z_{nU}^\sigma \) be the event time of \( z_{nU}^\sigma \) in the previous iteration.

If \( \xi \in L_{\xi} \) and task \( n \) is the related preceding unloading task such that \( z_n^U \in (z_{\xi U}^\sigma, z_{\xi U}^\sigma) \), then set \( z_n^U = z_{nU}^\sigma \). Go to Step 1.

If \( \xi \in U_{\xi} \) and task \( n \) is the related succeeding loading task such that \( z_n^U \in (z_{\xi U}^\sigma, z_{\xi U}^\sigma) \), then set \( z_n^U = z_{nU}^\sigma \). Go to Step 1.

**Step 5. Task assignment.** Evaluate \( t_{ij} \). Solve the assignment problem with the objective of minimizing the total travel time subject to constraint subset \( |T| \). Stop.
3.2.1. Feasibility checking

This step consists of two steps. In the first step, for given values of \( z_j, j \in L_2 \cup U_2 \), the feasibility can be checked by solving a maximum cardinality matching problem in the bipartite graph (Evans & Minieka, 1992). When the maximum cardinality is the same as \( |S \cup T| \), it can be concluded that the constraint subset \( \xi \) of (18)–(20) has a feasible solution. For a given feasible solution in Step 2.1, Step 2.2 checks whether \( z_i \) does not exceed the upper bound of the time window.

3.2.2. Delaying event time

To make constraint subset of (18)–(20) to be satisfied, one or more additional \( x_{ij} \) must be allowed to become 1 by relaxing constraint (20). That is, \( z_i \) is delayed by such an amount that at least one \( x_{ij} \), for \( i < \xi \), becomes 1, which is denoted as \( \pi_{\rho \xi} \). Also, in some cases, for satisfying constraint subsets of (18)–(20), it may be necessary to increase \( z_i \) beyond the upper bound of time window (constraint subset (21)), in which event time for QCs, \( y_n \), must be increased, and time windows must be updated simultaneously. The process is repeated until the current constraint subset becomes feasible.

3.2.3. Time window adjustment

After obtaining a feasible solution of the constraint subset \( \xi \) of 18, 19, 22, the time window must be adjusted if possible. When a feasible solution is found, the event time \( z_i \) of event (\( \xi \)) will be determined. Then, the time window of the event related to that of event (\( \xi \)) must be adjusted by extending its bound. For loading event (\( \xi \)), the upper bound of the related preceding unloading tasks must be extended. For unloading event (\( \xi \)), the lower bound of the related succeeding loading tasks must be extended. Fig. 5 shows an example for extending time windows of task \( L_4 \) and \( U_2 \) which correspond to loading and unloading task, respectively. In Fig. 5a, after the event time \( z_1 \) of \( U_2 \) is determined (an ALV is assigned to task \( U_2 \)), the lower bound of time window of the succeeding loading task \( L_4 \) can be extended to the position of \( z_1 \) (as shown in Fig. 5a). In Fig. 5b, after the event time \( z_4 \) of \( L_4 \) is determined, the upper bound of time window of the preceding unloading task \( U_2 \) can be extended to the position of \( z_4 \) (as shown in Fig. 5b).

An application of the heuristic algorithm for the example in Table 1 which includes QC 1, \( b_1 = 2, m_4 = 4 \), and two ALVs is illustrated as follows. The event numbers, \( i \) and \( j \), in Table 3 correspond to task numbers for QC 1. The time windows of this example can be obtained by considering the first four tasks of the example in Section 3.1, which is shown in Fig. 4d. (Note that \( y_4 = \infty, i \geq 5 \).

---

**Step 0. Initializing.** Set \( y_1 = 0, y_2 = 230, y_3 = 230, \) and \( y_4 = 460, W_1 = [0, 0], W_2 = [230, \infty], W_3 = [0, 230], \) and \( W_4 = [460, \infty] \). Thus, \( z_1 = 0, z_2 = 230, z_3 = 0, \) and \( z_4 = 460. \) And \( z_1 \mid U = 0, z_2 \mid U = \infty, z_3 \mid U = 230, \) and \( z_4 \mid U = \infty \). Let \( z_1^0 = z_2^0 = 0 \). Set \( \xi = 0 \).

**Step 1. Next task.** \( \xi = \xi + 1 = 1 \). Sequence the events in the increasing order of \( y_1^0, y_1^1 = (y_1^1, y_2^1, y_3^1, y_4^1) \).

**Step 2. Feasibility checking.**

**Step 2.1.** There is no feasible solution to constraints subset \( \xi \), (18)–(20).

**Step 3. Delaying event time.**

**Step 3.1.** Calculate \( \pi_{\rho \xi} = \min \{180 – (0–0); 180 – (0–0)\} = 180. \) Thus, \( z_1 = 180 \).

**Step 2. Feasibility Checking.**

**Step 2.1.** At least, a feasible solution to constraints subset \( \xi \), (18)–(20) can be found.

**Step 2.2.** Compare \( z_1 \) with its upper bound. It does not satisfy constraint subset \( \xi \), (21).

**Step 3. Delaying event time.**

**Step 3.2.** Let \( \pi_1 = 180 – 0 = 180. \) Then, \( y_1^1 = 180, y_2^1 = 410, y_3^1 = 410, \) and \( y_4^1 = 640. \) And update \( z_1^1 = 180, z_2^1 = 410, z_3^1 = 0, \) and \( z_4^1 = 640; \) and \( z_1 \mid U = 180, z_2 \mid U = \infty, z_3 \mid U = 410, \) and \( z_4 \mid U = \infty \).

**Step 2. Feasibility checking.**

**Step 2.2.** Compare \( z_1 \) with its upper bound. It satisfies constraint subset \( \xi \), (21).

**Step 4. Time window adjustment.** There is no preceding unloading task of task (\( \xi \)). Thus, there no adjustment of time windows.

**Step 1. Next task.** \( \xi = 2 \). Sequence the events in the increasing order of \( y_1^2, y_1^3 = (y_1^2, y_2^2, y_3^2, y_4^2) \).
Step 2. Feasibility checking.

Step 2.1. At least a feasible solution to constraints subset $\xi$, (18)-(20) can be found.

Step 2.2. Compare $z_2$ with its upper bound. It satisfies constraint subset $\xi$, (21).

Step 4. Time window adjustment. There is a succeeding loading task of task ($\zeta$). However, its lower bound $z_3$ is less than $z_2$, it cannot be extended.

Step 1. Next task. $\zeta = 3$. Sequence the events in the increasing order of $y_1^3, y_1^4 = (y_1^3, y_2^3, y_3^3, y_4^3)$.

Step 2. Feasibility checking.

Step 2.1. There is no feasible solution to constraints subset $\xi$, (18)-(20).

Step 3. Delaying event time.

Step 3.1. Calculate $p_{i3} = \min\{160 - (0 - 410); 200 - (0 - 0)\} = 200$. Thus, $z_3 = 200$.

Step 2. Feasibility checking.

Step 2.1. At least a feasible solution to constraints subset $\xi$, (18)-(20) can be found.

Step 2.2. Compare $z_3$ with its upper bound. It satisfies constraint subset $\xi$, (21).

Step 4. Time window adjustment. There is no unassigned preceding unloading task of task ($\zeta$). Thus, there is no adjustment of time windows.

Step 1. Next task. $\zeta = 4$. Sequence the events in the increasing order of $y_1^4, y_1^4 = (y_1^4, y_2^4, y_3^4, y_4^4)$.

Step 2. Feasibility checking.

Step 2.1. At least a feasible solution to constraints subset $\xi$, (18)-(20) can be found.

Step 2.2. Compare $z_4$ with its upper bound. It satisfies constraint subset $\xi$, (21).

Step 4. Time window adjustment. There is no succeeding loading task of task ($\zeta$). Thus, there is no adjustment of time windows.

Step 5. Task assignment. The solution from the assignment problem is as follows:

$$x_{11} = x_{12} = x_{13} = x_{14} = x_{22} = x_{32} = x_{42} = 1, \text{ and all the others, } X_{ij}^k = 0; \ y_1^1 = 180, y_2^1 = 410, y_3^1 = 410, \text{ and } y_4^1 = 640; \text{ and } z_1^2 = 180, z_2^2 = 410, z_3^2 = 200, \text{ and } z_4^2 = 640.$$

Fig. 6 illustrates the solution procedure. The final solution can be explained as follows: from the initial location, ALV 1 starts the empty travel to the pickup position of the 1st task of QC 1 and performs the loaded travel for the task. During the time required for ALV 1 to reach QC 1, QC 1 is waiting for the arrival of the ALV. After ALV 1 completes the 1st task of QC 1, it waits 230 s for performing the 2nd task of QC 1, and finally goes to the yard location of the task. In a similar way, the solution for ALV 2 can be interpreted as follows. ALV 2 delivers a container for the 3rd task of QC 1, and then waits 440 s for performing the 4th task of QC 1, and finally goes to the yard location of the 4th task.
4. Numerical experiments

A numerical experiment was conducted for evaluating the performance of the heuristic algorithm applied in this study and analyzing properties of the optimal solution.

4.1. Comparing the optimizing method and the heuristic algorithm

In this section, a number of problems are generated to compare the optimal solutions obtained from formulations (1)–(15) with those from a heuristic algorithm. In the generated problem, the number of ALVs per QC, the number of QCs, operations for each QC, and buffer capacities ranged from 1 to 4, from 1 to 2, from 4 to 12, and from 1 to 2, respectively. Formulations (1)–(15) were solved by ILOG CPLEX® 10.0.0. The generated problems are shown in Table 4 with the result of the numerical experiment.

Table 4 describes each problem by using the number of ALVs per QC, the number of QCs, the number of operations per each QC, and buffer capacity, which are given in the second column. The third and fourth columns show the objective values obtained by the heuristic algorithm and the optimization model. Also, the computational times were compared in the last three columns.

In Table 4, the ratio of the objective value by the heuristic algorithm to that by ILOG CPLEX® ranged from 1.000 to 1.016, and had the average value of 1.006. The ratio of the computational time for the heuristic algorithm to that of ILOG CPLEX® ranged from 0.00001 to 0.118, and had the average value of 0.026. We found that the computational time is sensitive to the total number of operations. Therefore, we can conclude that the heuristic algorithm showed very good performance.

4.2. Sensitivity analysis

To analyze effects of the dual cycle operation, the number of ALVs, and the buffer capacity on objective terms, various numerical experiments were conducted as follows: for evaluating the effect of the dual cycle operation on the performance of the vessel operation, an experiment was conducted by solving problems with the number of QCs of two, the number of ALVs of four, the buffer capacity of two, and the number of operations per each QC of twenty. The dual cycle operation is expected to reduce the total travel time of ALVs, delay time of QCs, and completion time in ship operations. For the experiment, ten problems are generated randomly for different percentages in dual cycle operations among twenty loading or discharging operations per QC.

Figs. 7–9 show that the total travel time of ALVs, delay time of QCs, and completion time in ship operations decreases rapidly as percentages of dual cycle increases. As the percentage of dual cycle operations increases from 10% to 90%, they were reduced by 53.7%, 51.5%, and 67.7%, respectively. These mean that the percentage of dual cycles significantly influences the total travel time of ALVs, delay time of QCs, and completion time in ship operations.

To analyze effects of the number of ALVs and buffer capacity, a numerical experiment was conducted as follows: the number of QCs and tasks per each QC are 2 and 20, respectively. The number of ALVs and buffer capacity were varied from 0 to 3 and from 2 to 7, respectively. For each combination, ten problems are generated randomly. Therefore, the total number of problems solved is 240.
The result from Fig. 10 shows that the delay time of QCs decreases rapidly as the number of ALVs increases. When the number of ALVs exceeds a certain level (five ALVs in Fig. 10), the reduction was negligible. For the change of the buffer capacity, the delay time of QCs decreases as the buffer capacity increases. However, when the number of ALVs reaches four (as shown in Fig. 10), the change in the buffer capacity does not affect the delay time of QCs.

Fig. 6. An illustration of the heuristic algorithm.
Fig. 11 shows the differences in the total delay time between the cases with buffers and the case without buffer. When the number of ALVs is two, the difference ranges from 5.4% to 17.7%. However, the difference decreases as the number of ALVs increases. The model in this paper is a generalized version of the study by Kim and Bae (2004) in which no buffer was provided.

Fig. 12 compares the average travel of ALVs. The cases with smaller buffer sizes showed the shorter average travel distance than cases with larger buffer sizes. It can be said that the reduction in the delay time was obtained at a cost of the other objective term of minimizing the travel distance.

5. Conclusions

This study discussed a method for dispatching ALVs for supporting efficient operations of QCs. The problem was introduced as a scheduling problem with precedence constraints and buffer constraints. A mixed-inte-

<table>
<thead>
<tr>
<th>No.</th>
<th>Problem</th>
<th>Objective value</th>
<th>Computational time (s)</th>
<th>Ratio</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-1-4-1</td>
<td>16660</td>
<td>1.000</td>
<td>0.05</td>
<td>0.84</td>
</tr>
<tr>
<td>2</td>
<td>1-1-6-1</td>
<td>16860</td>
<td>1.000</td>
<td>0.08</td>
<td>1.62</td>
</tr>
<tr>
<td>3</td>
<td>1-1-8-2</td>
<td>17060</td>
<td>1.000</td>
<td>0.09</td>
<td>10.76</td>
</tr>
<tr>
<td>4</td>
<td>1-1-10-2</td>
<td>17260</td>
<td>1.000</td>
<td>0.09</td>
<td>19.90</td>
</tr>
<tr>
<td>5</td>
<td>1-1-12-2</td>
<td>17460</td>
<td>1.000</td>
<td>0.08</td>
<td>35.01</td>
</tr>
<tr>
<td>6</td>
<td>2-1-4-1</td>
<td>16520</td>
<td>1.000</td>
<td>0.05</td>
<td>0.40</td>
</tr>
<tr>
<td>7</td>
<td>2-1-6-1</td>
<td>16720</td>
<td>1.000</td>
<td>0.05</td>
<td>0.62</td>
</tr>
<tr>
<td>8</td>
<td>2-1-8-2</td>
<td>17120</td>
<td>1.012</td>
<td>0.06</td>
<td>0.70</td>
</tr>
<tr>
<td>9</td>
<td>2-1-10-2</td>
<td>17320</td>
<td>1.012</td>
<td>0.08</td>
<td>5.28</td>
</tr>
<tr>
<td>10</td>
<td>2-1-12-2</td>
<td>17520</td>
<td>1.012</td>
<td>0.06</td>
<td>70.06</td>
</tr>
<tr>
<td>11</td>
<td>3-1-4-1</td>
<td>16520</td>
<td>1.000</td>
<td>0.05</td>
<td>0.81</td>
</tr>
<tr>
<td>12</td>
<td>3-1-6-1</td>
<td>16780</td>
<td>1.016</td>
<td>0.06</td>
<td>1.01</td>
</tr>
<tr>
<td>13</td>
<td>3-1-8-2</td>
<td>16980</td>
<td>1.016</td>
<td>0.06</td>
<td>1.42</td>
</tr>
<tr>
<td>14</td>
<td>3-1-10-2</td>
<td>17180</td>
<td>1.004</td>
<td>0.08</td>
<td>2.73</td>
</tr>
<tr>
<td>15</td>
<td>3-1-12-2</td>
<td>17380</td>
<td>1.003</td>
<td>0.06</td>
<td>10.57</td>
</tr>
<tr>
<td>16</td>
<td>4-1-4-1</td>
<td>16520</td>
<td>1.000</td>
<td>0.05</td>
<td>0.82</td>
</tr>
<tr>
<td>17</td>
<td>4-1-6-1</td>
<td>16780</td>
<td>1.004</td>
<td>0.06</td>
<td>1.01</td>
</tr>
<tr>
<td>18</td>
<td>4-1-8-2</td>
<td>17040</td>
<td>1.007</td>
<td>0.08</td>
<td>1.21</td>
</tr>
<tr>
<td>19</td>
<td>4-1-10-2</td>
<td>17240</td>
<td>1.007</td>
<td>0.08</td>
<td>2.25</td>
</tr>
<tr>
<td>20</td>
<td>4-1-12-2</td>
<td>17440</td>
<td>1.007</td>
<td>0.06</td>
<td>3.90</td>
</tr>
<tr>
<td>21</td>
<td>2-2-4-1</td>
<td>43420</td>
<td>1.000</td>
<td>0.08</td>
<td>4.67</td>
</tr>
<tr>
<td>22</td>
<td>2-2-6-1</td>
<td>43820</td>
<td>1.000</td>
<td>0.09</td>
<td>74.95</td>
</tr>
<tr>
<td>23</td>
<td>2-2-8-2</td>
<td>44520</td>
<td>1.007</td>
<td>0.08</td>
<td>126.75</td>
</tr>
<tr>
<td>24</td>
<td>2-2-10-2</td>
<td>44920</td>
<td>1.007</td>
<td>0.09</td>
<td>7809.70</td>
</tr>
<tr>
<td>25</td>
<td>2-2-12-2</td>
<td>45320</td>
<td>1.007</td>
<td>0.09</td>
<td>6976.00</td>
</tr>
<tr>
<td>26</td>
<td>3-2-4-1</td>
<td>43420</td>
<td>1.003</td>
<td>0.08</td>
<td>3.46</td>
</tr>
<tr>
<td>27</td>
<td>3-2-6-1</td>
<td>43820</td>
<td>1.003</td>
<td>0.08</td>
<td>30.75</td>
</tr>
<tr>
<td>28</td>
<td>3-2-8-2</td>
<td>44520</td>
<td>1.010</td>
<td>0.06</td>
<td>31.51</td>
</tr>
<tr>
<td>29</td>
<td>3-2-10-2</td>
<td>44920</td>
<td>1.010</td>
<td>0.06</td>
<td>209.90</td>
</tr>
<tr>
<td>30</td>
<td>3-2-12-2</td>
<td>45320</td>
<td>1.010</td>
<td>0.09</td>
<td>14729.00</td>
</tr>
<tr>
<td>31</td>
<td>4-2-4-1</td>
<td>43420</td>
<td>1.004</td>
<td>0.05</td>
<td>3.45</td>
</tr>
<tr>
<td>32</td>
<td>4-2-6-1</td>
<td>43820</td>
<td>1.004</td>
<td>0.08</td>
<td>85.60</td>
</tr>
<tr>
<td>33</td>
<td>4-2-8-2</td>
<td>44520</td>
<td>1.011</td>
<td>0.08</td>
<td>50.00</td>
</tr>
<tr>
<td>34</td>
<td>4-2-10-2</td>
<td>44920</td>
<td>1.011</td>
<td>0.08</td>
<td>242.60</td>
</tr>
<tr>
<td>35</td>
<td>4-2-12-2</td>
<td>45320</td>
<td>1.011</td>
<td>0.08</td>
<td>2100.00</td>
</tr>
</tbody>
</table>

Average 1.006 0.02619
Min 1.000 0.00001
Max 1.016 0.11750

* Number of ALVs per QC – number of QCs – number of operations per each QC – buffer capacity.
A new programming model was proposed for the ALV scheduling problem. In addition, a procedure was suggested for converting buffer constraints into time window constraints. Then, a heuristic algorithm was suggested for solving the scheduling problem with precedence constraints and time window constraints.

Fig. 7. Effects of the percentage of dual cycles on the total travel time of ALVs.

Fig. 8. Effects of the percentage of dual cycles on the total delay time of QCs.

Fig. 9. Effects of the percentage of dual cycles on the completion time of ship operations.
Fig. 10. Effects of the number of ALVs on the total delay time of QCs.

Fig. 11. Improvements in the delay time of the cases with buffers compared with that without buffer.

Fig. 12. Effects of the number of ALVs on the total travel time of ALVs.
A number of problems were generated to compare the optimal solution from the mathematical formulations with those from the heuristic algorithm. According to the results of this experiment, it was found that the ratio of the objective values in the heuristic algorithm to the optimal objective values in the ILOG CPLEX \(^\text{\textregistered}\) was between 1.000 and 1.016, and had the average value of 1.006. The ratio of the computational time in the heuristic algorithm to that of ILOG CPLEX \(^\text{\textregistered}\) ranged from 0.00001 to 0.118, and had the average value of 0.026. Thus, it could be concluded that the heuristic algorithm showed such a good performance that an application to practice may be positively considered.

The sensitive analysis showed that as the proportion of dual cycle operations increases, there have been significant improvements in the total travel time of ALVs, delay time of QCs, and completion time in ship operations. As the number of ALVs increases, the total delay in QC operations decreases as we expected. The effect of the buffer size was large when the number of ALVs is small, while the effect decreases as the number of ALVs increases. About the total travel time of ALVs, the cases with smaller buffer sizes showed the shorter average travel distance than cases with larger buffer sizes.

As a future study, a simulation experiment will be necessary to evaluate the performance of the algorithm proposed in this study in a dynamic environment. This study addressed only the dispatching of ALVs, however, the integrated scheduling problem of the travels of ALVs, operations of yard cranes, and operations of QCs may be addressed as a further study.

Acknowledgments

This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Advancement) (IITA-2008-C1090-0901-0004).

References


