ROUTING METHOD FOR AUTOMATED LIFTING VEHICLES CONSIDERING THE RESERVATION SCHEDULE OF TRANSFER POINTS

by

Ri Choe[1], Donggyun Lee[2], and Kwang Ryel Ryu[3]
Department of Computer Engineeering, Pusan National University,
Geumjeong-gu, Busan, Korea
E-mail: choi.lee.choe.ri@gmail.com[1], finht@naver.com[2], krryu @pusan.ac.kr[3]

ABSTRACT

A transfer point (TP) in a seaport container terminal refers to a small buffer area at which the exchange of a container between a crane and a vehicle takes place. An automated lifting vehicle (ALV) has the capability of picking up a container from the ground by itself and can pass over a container grounded at a TP whether or not it is loaded with a container. If an ALV needs to pass through multiple TPs until reaching its destination TP, it must reserve each of those TPs ahead of time to block other equipment, i.e., crane or other ALVs, from attempting to approach the TP in danger of a collision. However, excessively long reservations of TPs can become cause of delays of other equipment, deteriorating the productivity of the entire terminal. In this paper, we propose a routing method that can find the shortest-time routes using A* algorithm with the reservation schedule of the TPs taken into account. We also propose a rerouting method that allows change of the routes of ALVs during traveling whenever any TP reservation schedule is violated due to the uncertainty of equipment operation. The proposed method has been evaluated through simulation experiments.

KEYWORDS
Container Terminal, ALVs, Deadlock, Shortest-Time Routes

INTRODUCTION

At some automated container terminals, automated lifting vehicles (ALVs) are used to transfer containers between quay cranes (QCs), which load and unload berthed cargo vessels, and automated stacking cranes (ASCs), which handle temporary storage of containers. The exchange of a container between a crane and an ALV takes place at a small buffer area called the transfer point (TP). Although a QC can place a container at a TP and move on to the next operation without waiting for an ALV, and although an ALV can lift and hold a container high enough to pass over other containers on a TP, the potential for ALV-QC collisions remains. To prevent such collisions, QCs and ALVs currently reserve all necessary TPs ahead of time, thereby blocking their traversal by other equipment for the duration of the reservation. This course-grained method causes deadlocks among ALVs as they compete for TPs, and can severely retard terminal operation.

Previous studies on equipment deadlocks have focused on terminals employing AGVs. Lehmann et al. (2006) detected deadlocks using a matrix composed of the container terminal resources, and resolved them by modifying the order of AGV operations. Rajeeva et al. (2003) presented a method to avoid AGV deadlocks and collisions using zone control policies. Kim et al.

(2006) showed how to prevent deadlocks using a block schedule grid and a guide path, and Bae et al. (2008) similarly prevented deadlocks by scheduling the movement of AGVs on routes. Unfortunately, none of these approaches addresses the deadlocks that result from simultaneous reservation of TPs by ALVs.

In this paper, we propose an ALV routing method that uses an A* algorithm not only to avoid deadlocks but also to find shortest-time route for a given TP reservation schedule. We also propose a method to reroute ALVs during travel whenever the TP reservation schedule is violated due to the uncertainty of equipment operation.

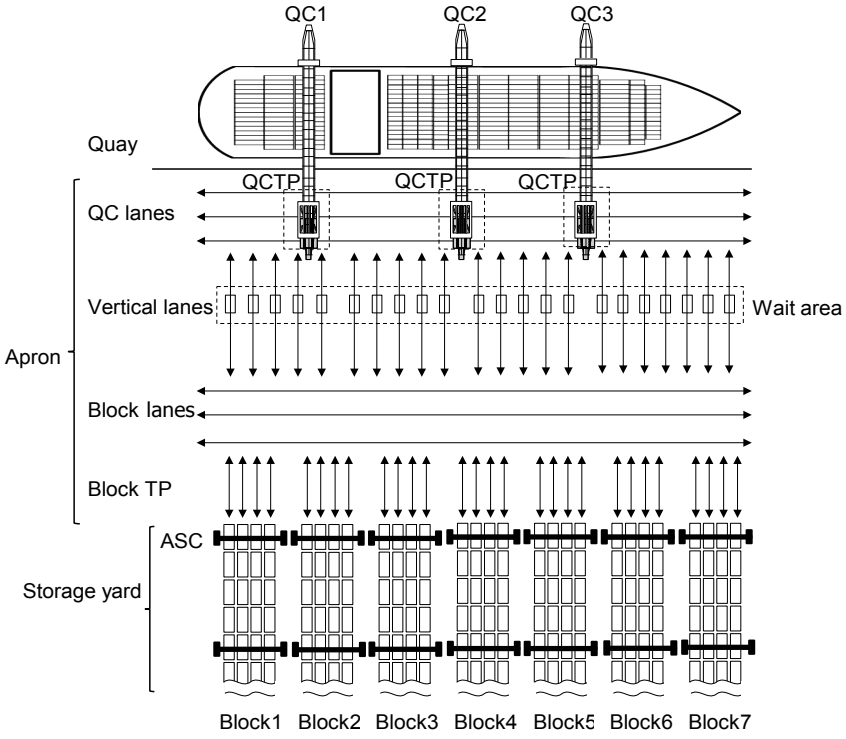
The rest of this paper is organized as follows: section 2 explains the general layout of an automated container terminal; section 3 describes our method for preventing deadlocks among ALVs as they reserve and travel between TPs, and the method for finding the shortest-time route given the current state of the reservation schedule; section 4 presents our experimental results using the proposed methods; and section 5 gives concluding remarks.

AUTOMATED CONTAINER TERMINAL

Layout of Automated Container Terminal

Figure 1 shows the layout of our hypothetical automated container terminal. It is divided into three regions: the quay, the storage yard, and the apron. The quay is where ships dock for loading and unloading of containers. The storage yard is where containers are temporarily stored and is divided into a number of blocks, each of which has two ASCs for handling the containers, and an assigned set of TPs within the apron. The apron is where ALVs operate to deliver containers between the quay and storage yard. There are many horizontal and vertical drive lanes in the apron: the three QC lanes pass through the TPs under the QCs; the three block lanes allow horizontal movement of ALVs in front of the storage blocks; and the vertical lanes connect QC lanes and block lanes. Any of the vertical lanes can become a waiting area (WA), where an ALV with no allocated TP can park and wait. As with TPs, WAs must be reserved ahead of time to avoid collision.

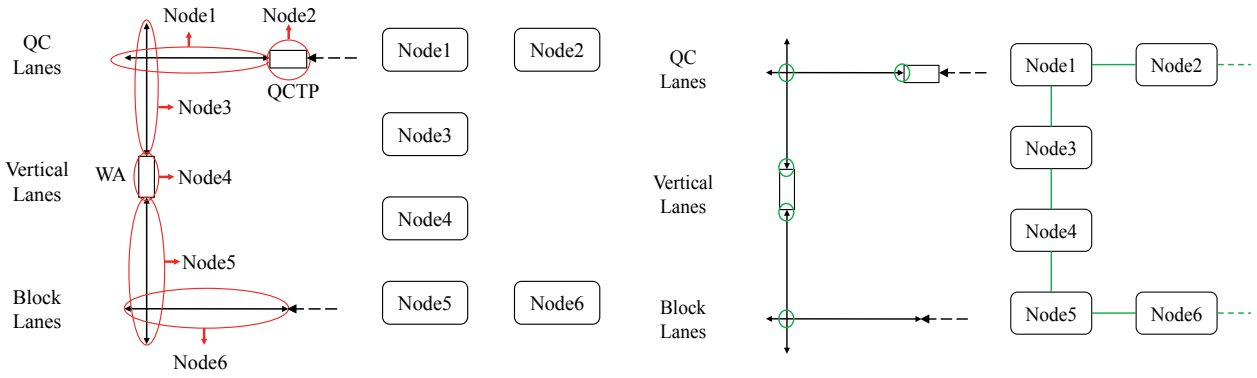
FIGURE 1
LAYOUT OF AN AUTOMATED CONTAINER TERMINAL



The Path Network in Graph Form

In this paper, the apron is modeled as a standard graph, consisting of nodes and edges. All TPs, WAs and lanes of the apron are represented as nodes. QC lanes and vertical lanes are separated by TPs and WAs, respectively, and block lanes are separated into smaller lanes of length equal to a QC lane. All separated lanes are considered distinct nodes and all connections between these nodes are edges. For example, in FIGURE B-2, there is an edge between node1 and node2, as they are directly connected, but no edge between node1 and node4. We refer to this graph as our "path network."

FIGURE 2
GRAPH REPRESENTATION OF APRON



A. graph nodes

B. graph edges

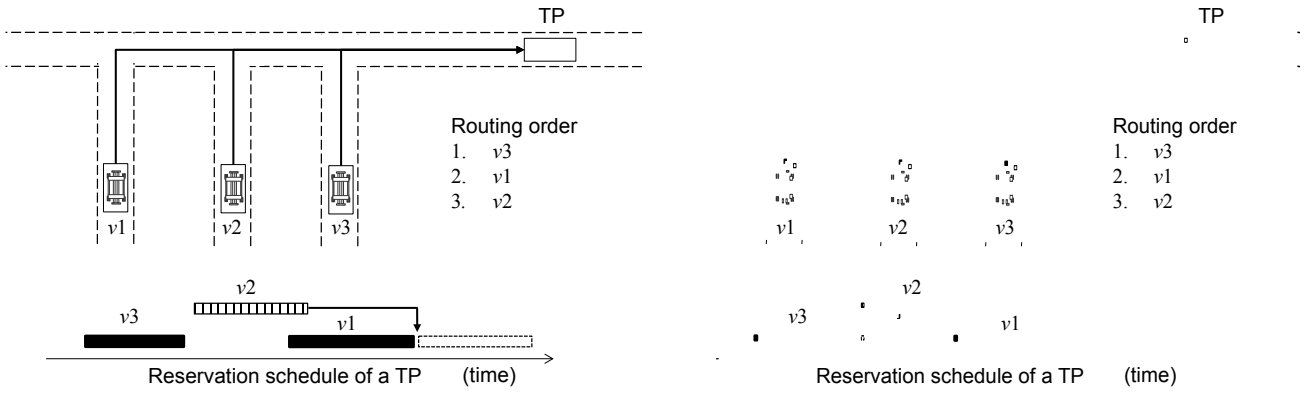
METHOD FOR ROUTING ALVs

In this section, we present our method for routing ALVs between TPs. First, we present the basics of avoiding and resolving deadlocks. We then describe our method for efficient, deadlock-free routing using the A* algorithm in consideration of the reservation schedule. Finally, we explain our rerouting method for handling unexpected violations of the reservation schedule.

Methods for Handing Deadlocks

In essence, our routing method avoids deadlocks by comparing the reservation timespan during which a moving ALV has safely reserved the TPs along its route. If the reservation timespan for a given ALV overlaps with the reservation timespan of ALVs driven earlier, then the reservation timespan of that ALV is changed as shown in FIGURE A-3. If it does not overlap with the reservation timespans of any ALVs driven earlier, then it is allowed to preferentially reserve the necessary TPs. This method prevents deadlock because ALVs driven earlier never fail to reserve the necessary TPs, and so no reservation cycling occurs.

FIGURE 3
BASICS OF AVOIDING DEADLOCK

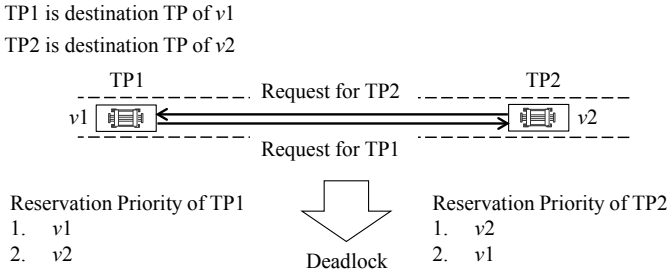


A. Overlap with the reservation timespans of any ALVs driven earlier

B. Not overlap with the reservation timespans of any ALVs driven earlier

Unfortunately, this method cannot prevent deadlocks caused when the destination TP of one ALV is the destination TP of another ALV, as shown in FIGURE 4. In this case, each ALV has a lower priority for its destination TP than the other, creating a deadlock.

FIGURE 4
ANOTHER DEADLOCK



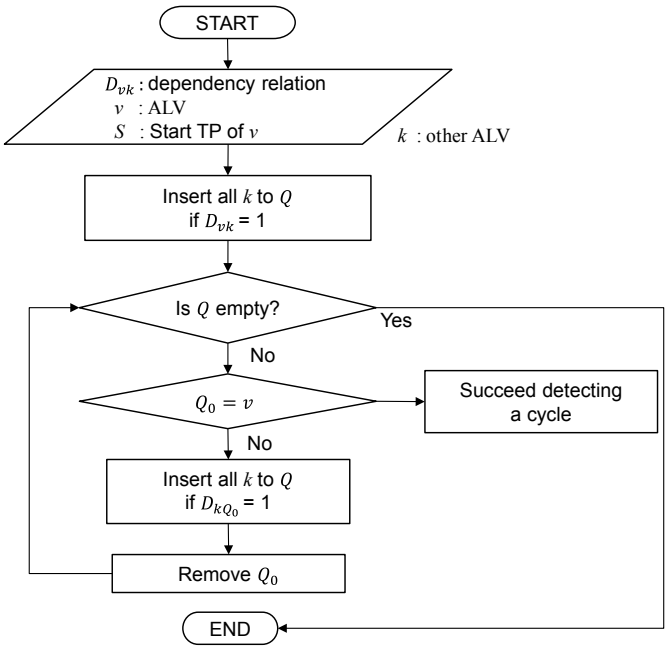
To resolve this deadlock, assume an ALV driving to a destination TP occupied by another ALV fails to find a route, and represent the dependency relations between ALVs using a sparse matrix, as that shown in FIGURE 5. Matrix D contains only 0s and 1s. If D ($v3$, $v2$) = 1, then we say that $v2$ is waiting for $v3$. After setting all such dependencies, we detect a cycle using the algorithm diagrammed in FIGURE 6. If a cycle is detected, the affected ALV must drive to any TP that can resolve the cycle.

FIGURE 5
MATRIX OF DEPENDENCY RELATION BETWEEN ALVs

$D =$

	$v1$	$v2$	$v3$	$v4$	$v5$
$v1$	0	0	0	0	0
$v2$	0	0	1	0	0
$v3$	0	0	0	0	0
$v4$	0	0	0	0	0
$v5$	0	0	0	0	0

FIGURE 6
ALGORITHM FOR DETECING A CYCLE



Methods for finding route

We determine all routes using an A* algorithm on the path network defined above. A* is an optimal, best-first search heuristic that computes a cost function for various locations in the given environment. Equation (1) is the cost function that is minimized at each step of the A* algorithm.

$$f(n) = g(n) + h(n) \tag{1}$$

Where $g(n)$ is the actual cost from the start point to the intermediate point n and $h(n)$ is the estimated cost from the intermediate point n to the destination point. At each step in the A* propagation, the minimized $f(n)$ value is selected and inserted into a sorted list. $h(n)$ is calculated using Equation (2), where v is maximum speed of ALVs.

$$h(n) = \frac{\text{Manhattan distance}(n, \text{destination point})}{v} \tag{2}$$

As the nodes of path network can be either lanes or TPs, $g(n)$ must be calculated differently depending on the type of node. When the node is a lane, $g(n)$ is calculated by estimating the travel time to end of the lane; when the node is a TP, $g(n)$ is calculated by estimating the time of departure from this TP using the reservation schedule, as in FIGURE 3.

Rerouting

After a route is determined for an ALV, the reservation schedule of each TP in its route is updated by using the method illustrated in FIGURE 3 to have the necessary reservation timespan be included before driving. In our simulation model, an ALV drives to its destination TP over a series of reserved TPs, one at a time. An ALV driven earlier has higher priority for reserving TPs, but we allow that an ALV driven late may preferentially reserve a TP when it can leave earlier from a TP than an ALV driven earlier. However, this method can leave an ALV driven earlier without the ability to reserve TPs used by ALVs driven later, due to differences between the real reservation time of a TP and the scheduled reservation time of a TP. To avoid deadlock, we change the routes of ALVs during travel whenever any TP reservation schedule is violated. Rerouting is processed in the order in which ALVs are driven, and may change routes reserving a TP to a destination TP.

EXPERIMENTAL RESULTS

To evaluate the proposed methods, we used the following experimental procedure: We performed a baseline experiment targeting a container terminal with berths the size of those shown in Figure 1. We then increased the number of ALVs from the baseline (12) to 30, with ALVs randomly driving 3,000 times in every experiment. Each experiment was performed 10 times and the results averaged. Using this procedure, we compared the performance of the proposed schedule-aware method with a schedule-agnostic method.

Figure 7 plots the average TP reservation delay for different numbers of ALVs. Both methods achieved the minimum delay time when the number of ALVs was 12, but as the number of ALVs increased, the delay time of the schedule-agnostic routing method increased much more sharply.

FIGURE 7
AVERAGE TP RESERVATION DELAY FOR ALVs

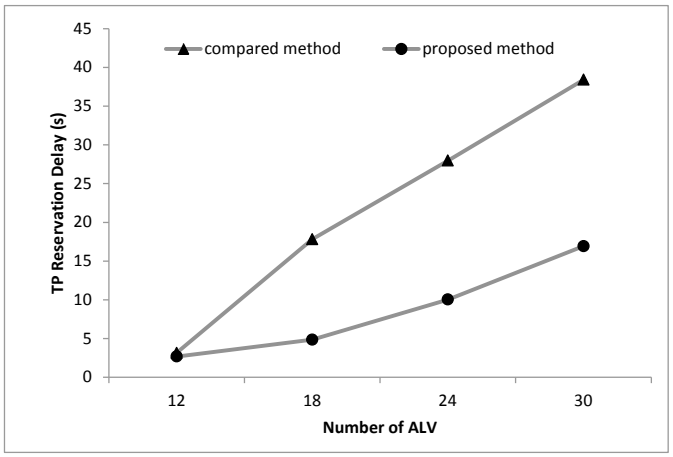


Figure 8 shows the average travel distances for ALVs. Note that the proposed method produces longer travel distances than the schedule-agnostic method for higher numbers of ALVs. The reason for this is that the proposed method finds routes to optimize time but not distance.

FIGURE 8
AVERAGE TRAVEL DISTANCE OF ALVs

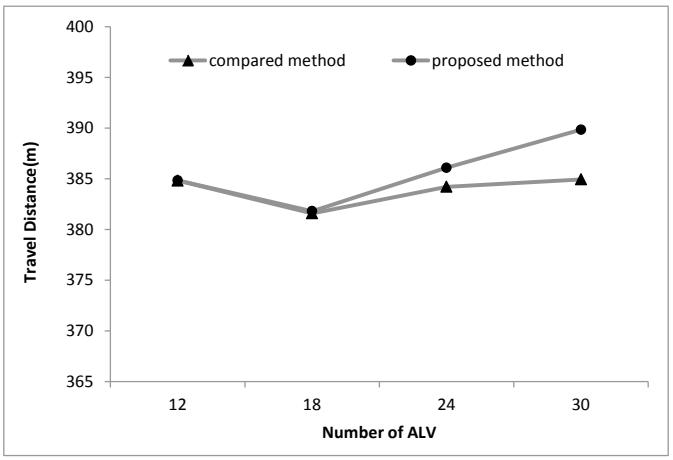
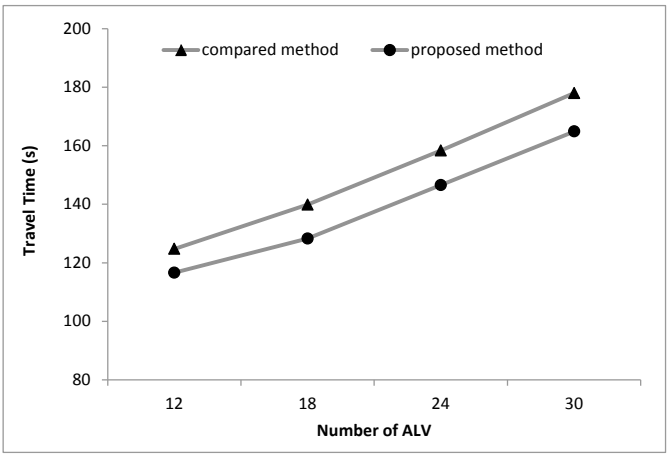


Figure 9 shows the average ALV travel times. These results show that the proposed method produces lower travel times for any number of ALVs. From this, we infer that the proposed method's increases in travel distance resulted in a consistent decrease in travel time. Finally, it should be noted that no deadlocks occurred during all the experiments.

FIGURE 9
AVERAGE TRAVEL TIME OF ALVs



CONCLUSION

In this paper, we proposed a schedule-aware ALV routing method that consistently prevented deadlocks and reduced travel time. We also described enhancements to the method that will allow dynamic rerouting in response to unexpected violations of reservation schedule. We demonstrated the superior performance of our method through a battery of simulation experiments. In the future, we plan to enhance the responsiveness of the proposed method to schedule uncertainties.

ACKNOWLEDGEMENTS

This work was supported by the Brain Korea 21 Project in 2006.

REFERENCES

Bae, H. Y., Choe, R., Park, T. J., & Ryu, K. R. (2008), "Travel Time Estimation and Deadlock-free Routing of an AGV System," *International Conference on Dynamics in Logistics*, pp.77-84

Lehmann, M., Grunow, M., & Gunther, H. O. (2006), "Deadlock handing for real-time control of AGVs at automated container terminals," *OR Spectrum*, Vol.28, No.4, pp.631-657

Kim, K. H., Jeon, S. M., & Ryu, K. R. (2007), "Deadlock prevention for automated guided vehicles in automated container terminals," *Container Terminals and Cargo Systems*, pp.243-263

Moorthy, R. L., Wee, H. G., & Ng, W. C. (2003), "Cyclic deadlock prediction and avoidance for zone-controlled AGV system," *International Journal of Production Economics*, Vol.83, No. 11, pp. 309-324