Chapter 2
Routing and Scheduling Transporters in a Rail-Guided Container Transport System

Xuefeng Jin, Hans-Otto Guenther and Kap Hwan Kim

Abstract One of the important issues in container terminals for the efficient operation of a rail-based transport system, called "flat car system (FCS)," is how to determine the route and travel schedule of each flat car (FC). It is assumed that a flat car may have to reserve multiple resources simultaneously at a moment during its travel. Example of the resources are transfer points (TPs) and intersections (ISs) on guide-path network. A travel-scheduling algorithm based on Dijkstra's algorithm is suggested using the concept of time-windows.

Keywords Transporters · Container terminal · Routing and scheduling

2.1 Introduction

When applying an FCS to container deliveries, the following problems must be studied: (1) guide-path design; (2) a method for dispatching FCs; (3) an FC routing or routing-scheduling method for FCs; and (4) traffic management. The routing and travel-scheduling methods for FCs are the issue of this research.

Routing implies the selection of a path for an FC to travel from its origin to its destination, while the travel-scheduling of an FC is the determination of not only the path for the FC from the origin to the destination but also the time-points for an FC to enter and exit from every segment on the path.

X. Jin (✉) · K.H. Kim
Department of Industrial Engineering, Pusan National University, Jangjeon-dong 2, Geumjeong-gu, Busan 609-735, Korea, South Korea
e-mail: hbkim@pusan.ac.kr

K.H. Kim
e-mail: kapkim@pusan.ac.kr

H.-O. Guenther
Institute of Logistics Innovation and Networking, Pusan National University, Jangjeon-dong 2, Geumjeong-gu, Busan 609-735, Korea, South Korea
e-mail: hans-otto.guenther@hotmail.de

M. Freitag et al. (eds.), *Dynamics in Logistics*, Lecture Notes in Logistics,
DOI 10.1007/978-3-319-45117-6_2

Usually, FCs travel on the floor, which has a network consisting of TPs and ISs. Designers of an FCS usually predetermine allowable paths on the floor and thus, the allowable paths can be represented by a guide-path network in which nodes represent TPs and ISs and arcs represent path segments between nodes. When a guide-path network and the origin and destination of a delivery order are provided, the travel schedule for an FC with the delivery order must be constructed so that the travel time is minimized. In the process of constructing the schedule, collisions and interferences between moving FCs must be considered.

Contrary to static routing, the dynamic-routing method considers the travel schedules of other FCs when the supervisory computer constructs a travel schedule for the given FC. Since the travel schedules of other FCs are respected, the congestion or the interference of the FC with other FCs can be considered for reducing the travel time as much as possible.

The concepts of conflict-free shortest-time vehicle routing have been introduced by some researchers (Broadbent and Besant 1985; Egbelu and Tanchoco 1983; Huang and Palekar 1989; Walker and Premi 1985). However, Kim and Tanchoco (1990, 1997) were the first to present an optimizing algorithm for finding conflict-free, shortest-time routes for AGVs by utilizing time-window concepts. They introduced the concept of a time-window graph in which the node-set represents the free (uncommitted or unreserved) time-windows and the arc-set represents reachability between the free-time-windows. The algorithm routes the vehicles through the free-time-windows of the time-window graph instead of the physical nodes of the flow-path network.

Rajotia et al. (1998) proposed another method for dynamic routing, where the main algorithm is similar to the path-planning method of Kim et al. (2006). The study maintained time-window data not only at nodes but also at arcs for applying Dijkstra's algorithm for finding the shortest-time routes for a vehicle. With the same concept of time-windows, Lim et al. (2002) suggested a dynamic-routing algorithm under the assumption that the schedules of previous vehicles can be altered when the schedule of a new vehicle is constructed.

Maza and Castagna (2005) improved the dynamic-routing algorithm so that the algorithm could handle changing situations such as vehicle delays and failures. His study proposed a two-stage approach: one stage for finding the shortest-time routes for AGVs and the other stage for avoiding conflicts and deadlocks in real time while maintaining the determined routes of AGVs.

2.2 Problem Definition and a Scheduling Algorithm

This paper assumes that three types of equipment are used for ship operations: QCs (Quay Cranes), FCs (Flat cars), and OSS (Overhead Shuttle System). For example, unloading operations are performed in the following three steps. The first step involves QCs, which transfer containers from a ship to FCs. The second step is to

deliver containers from the quay side to the yard side through FCs. The last step is to transfer containers from FCs to the storage positions through OSs.

Figure 2.1 illustrates a new concept of an automated container terminal and the equipment used for the unloading operation. The loading operation is performed in the reverse order to that of the unloading operation.

Figure 2.2 illustrates a layout of the rail-based driving area. In this example, the driving area consists of three main parts: berth lanes, parking lanes, and block lanes. On berth lanes, an FC travels to the transfer point of a QC and receives (or transfers) a container from (or to) the QC. On parking lanes, FCs may park to await the next delivery order or pass over to travel between berths and blocks. On block lanes, FCs either move to (from) transfer points at blocks (TPs) for transferring containers to (or from) yard cranes or travel parallel to the quay across different blocks. We define physical nodes as TPs or ISs.

The following notation will be used for describing the algorithm.

p_i ID for physical node i

n_i state node i, which is defined by a state of an FC at a physical node p_i, which are TPs or ISs, as illustrated in Fig. 2.3

S_i state of n_i, which is represented by (physical node ID, movement direction, speed)

Q_i set of resources associated with n_i, including TPs or ISs on which an FC cannot stay when an FC is on the physical node corresponding to n_i

F_i^n set of time-intervals corresponding to free (unreserved) time-windows of n_i

R_i^n set of time-intervals corresponding to reserved time-windows of n_i

f_j free-time-window (FTW) node j

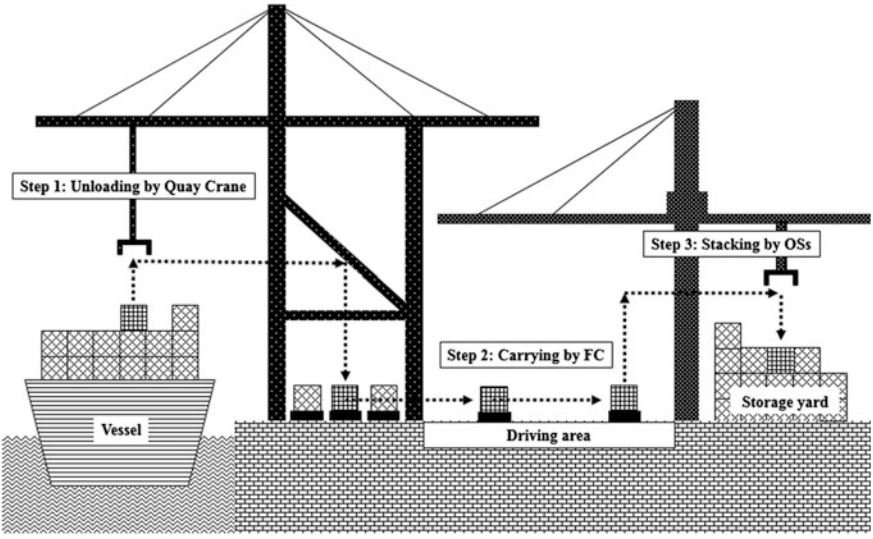

Fig. 2.1 An illustration of unloading operations and the terminal equipment

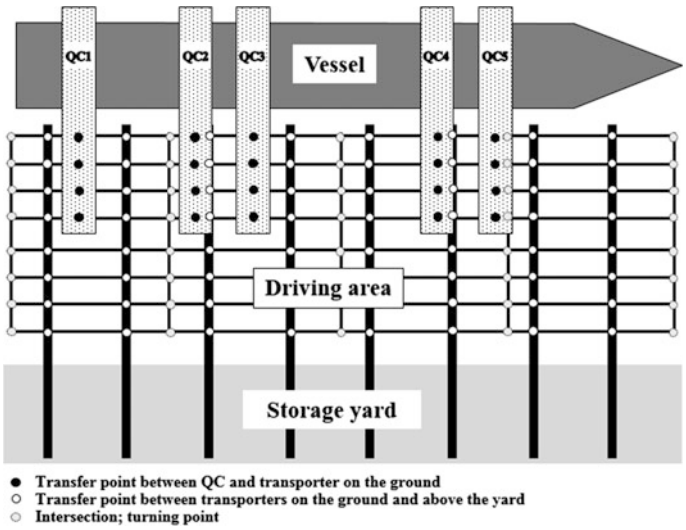

Fig. 2.2 Layout of the terminal

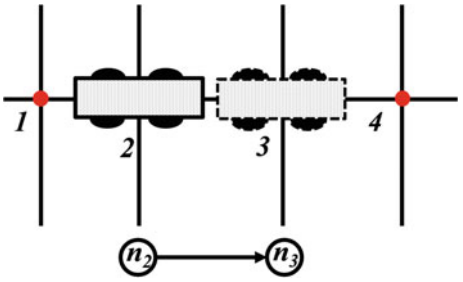

Fig. 2.3 An example of a node

$\hat{f}_j$	node in the state-feasible graph from which f_j originated
t_j^l	lower bound of the free-time-window (FTW) node j
t_j^u	upper bound of FTW node j
L	set of labeled FTW nodes, according to Dijkstra's algorithm
$L_{\max}$	maximum allowed number of labeled FTW nodes in L

2.2.1 Constructing a State-Feasible Graph

A state node n_i is defined by the state S_i of an FC. An example of the state of an FC is the triple consisting of the physical node ID, the moving direction, and the speed.

Suppose state node n_2 is defined by the state S_2, (p_2, *R*, *3*) (which means that the FC is located at physical node *2* and moves from left to right at a speed of *3* m/s) and

node n_3 is defined by (p_3, *R*, *0*). Then, an arc can be connected from n_2 to n_3 only if an FC can reduce its speed from *3 m/s* to *0 m/s* during the travel from p_2 to p_3.

Let $G(V,A)$ be the graph constructed by $V = \{n_i\}$ and $A = \{a_{ij}\}$, where a_{ij} stems from n_i and is directed to n_j and is defined only when the state change is feasible (considering their relative speeds and positions) from n_i to n_j. Graph $G(V,A)$ is called a "state-feasible graph." In Fig. 2.3, node n_3 can be connected to node n_2 if the state change is feasible (considering their relative speeds and positions).

When the number of attributes defining the state increases, the number of nodes also increases. As a result, the size of the problem for Dijkstra's algorithm for finding the shortest-time route may become large. Previous approaches maintain information about free-time-windows only for nodes, while the algorithm in this paper needs to maintain those not only for nodes but also for resources. The time segment for a node is available for a vehicle only if the same time segment is available for all the resources related to the node. This relationship will be maintained by a propagation process to be explained later.

2.2.2 Free-Time-Window Graph

For an FC to travel from an origin to a destination, it must reserve the required time segment of each state node on the travel route. Each state node has both free and reserved time-windows. The reserved time-windows represent the time-segments that other FCs have already reserved for their travel. The free-time-windows are the remaining time segments after the reserved time-windows are removed from the entire planning-horizon.

In a free-time-window (FTW) graph, each FTW $\left(F_i^n\right)$ of a state node in the state-feasible graph becomes an FTW node. For an FTW node (f_i), to have a directed arc to another FTW node (f_j), the following two conditions must be satisfied: (1) in the state-feasible graph, there is an arc from node n_i, from which f_i originates, to node n_j, from which f_j originates; and (2) the travel from f_i to f_j is time-feasible. The time feasibility can be checked as follows: suppose that the earliest departure time from node n_i is t and the travel time from n_i to n_j be t_{ij}. It is time-feasible if there exists (t_i, t_j) where (C1) $t \le t_i$; (C2) $t_j = t_i + t_{ij}$; and (C3) $t_j \le t_j^u - t(n_j)$, and where $t(n_j)$ is the shortest time for the FC to move to the position where there is no interference another FC at the physical position of n_j. Among all the possible values of t_i, we will select the earliest time.

2.2.3 Propagation Process

For an FC to travel from n_i to n_j during (t_i, t_j), applying a conservative strategy, the time-window (t_i, t_j) should be reserved for n_i and (t_i, $t_j + t(n_j)$) should be reserved

for n_j. Then, all the related resources in Q_i and Q_j, respectively, should be reserved for the same corresponding time-window.

2.2.4 Constructing a Shortest-Time Route for an FC

This paper assumes that a travel schedule is constructed whenever an FC starts its travel from an origin to a destination. In this process, the travel schedules of other FCs, which were constructed before, are respected and are therefore unchanged. Using the FTW graph, the algorithm in this study attempts to find the shortest-time path for an FC from an origin to a destination. This algorithm is based on Dijkstra's algorithm combined with a heuristic rule for restricting the number of labeled nodes during the search process. The details of the algorithm for constructing a travel schedule are explained below.

Step 0: $L: = \varnothing$. Include the source FTW node (the FTW node that includes time zero of the starting state) in L.

Step 1: List all the unlabeled FTW nodes by (1) selecting all the unlabeled nodes directly connected to one of the nodes in L; (2) and then selecting one node with the connecting arc of the shortest travel time among all unlabeled nodes connected to each node in L.

Step 2: Select the listed FTW node that has the shortest travel time from the source FTW node. Check if the selected node can be connected to any other FTW node that is not in L. If yes, then go to step 3. Otherwise, remove this FTW node from the set of listed FTW nodes and repeat step 2.

Step 3: If the selected FTW node is the destination node, then go to step 4. Otherwise, add the selected FTW node to L and calculate the departure time from n_i (t_i) and the arrival time at n_j (t_j). Add the selected node to the current FTW graph. If the number of elements in L is larger than L_{max}, then remove an element from L that has the longest expected travel time from the source FTW node to the destination node. Go to step 1.

Step 4: List the shortest-time route. Reserve the corresponding time-windows of all the nodes in the shortest-time route and perform the propagation process. Stop.

2.2.5 A Heuristic Method for Reducing the Computational Time

Since the travel schedule must be constructed in real time, the computational time of the algorithm is very critical. When the number of nodes in the driving area

becomes large, the size of the FTW graph may become too large for finding the shortest-time route in real time. For reducing the computational time, the maximum number of labeled FTW nodes is not allowed to exceed $L_{\max}$. If the number of labeled FTW nodes already exceeds $L_{\max}$, the FTW nodes with the longest estimated travel times to the destination are removed from L until the cardinality of L is at most $L_{\max}$. Note that each labeled FTW node of the form f_j has the shortest travel time from the origin to f_j and thus, the expected time from the origin to the destination can be calculated by adding the shortest travel time up to f_j and the expected travel time from $\hat{f}_j$ to the destination.

The initial value of the expected travel time from one node to another node is calculated by dividing the corresponding shortest travel distance by the maximum speed of an FC. Subsequently, through the following formula, the expected travel time from a node O to another node D is revised whenever a vehicle arrives at the destination of a route on which both nodes O and D are included: $t_{OD}^{rev} = \alpha \cdot t_{OD}^{new} + (1-\alpha) t_{OD}^{old}$, where t_{OD}^{rev} and t_{OD}^{old} are the revised and prior estimates of the expected travel time, respectively. t_{OD}^{new} is the travel time that has just been realized by the FC upon arriving at node D. α is the smoothing parameter of exponential smoothing.

Additional strategies for improving performance will be tested in the future.

(S1) **Delaying the travel start**: FCs should not arrive at the destination earlier than necessary. If it is too early for an FC to start its travel considering its due time, then the FC should intentionally delay its departure. The intention of this logic is to reduce the time of stay of an FC at congested areas. To delay the departure, the FC may be routed to a prespecified parking area before it starts its travel.

(S2) **Backward scheduling**: another scheduling strategy is to back schedule an activity for just-in-time arrival at the destination.

(S3) **Changing schedules of FCs which started their travel earlier**: the algorithm in this study gives higher priorities to the schedules being implemented. This strategy is for the convenience of the scheduling process, which may limit the optimality of the schedules. One minor modification is to track the total slack at each node, which represents the maximum time allowed to be delayed without delaying the scheduled arrival time at the destination. Then, when an FC incurs interference during the schedule at a node, then it is checked whether previous schedules may be delayed to make a way to the new schedule by utilizing the slacks.

The above strategies are subject to intensive testing for their effectiveness.

2.3 Conclusions

FCs routing and travel-scheduling are key issues of operation control of an FCS. In this paper, we focus on both, the problem of FC routing and that of travel-scheduling. A travel-scheduling algorithm based on Dijkstra's shortest-path algorithm is suggested using the concept of time-windows.

For future research, it is necessary to develop a simulation model for testing the performance of the developed heuristic and compare the method in this paper with existing heuristic algorithms. For reducing the computational time, further studies on various strategies for restricting the search space are necessary.

Acknowledgments This work was supported by the Technological Development of Low-carbon Automated Container Terminals funded by the Ministry of Oceans and Fisheries, Korea (Project Number: 201309550003). This work was also supported by the Korean Federation of Science and Technology Societies (KOFST) grant funded by the Korean government (MSIP: Ministry of Science, ICT and Future Planning).

References

Ashayeri J, Gelders LF, Van Looy PM (1985) Micro-computer simulation in design of automated guided vehicle systems. Mater Flow 2(1):37–48

Broadbent AJ, Besant CB (1985) Free-ranging AGV systems: promises, problems and pathways. In: Proceeding of the 2nd international conference on automated material handling, Birmingham UK, pp 221–237

Egbelu PJ, Tanchoco JMA (1983) Designing the operations of automated guided vehicle system using AGVSim. In: Proceeding of the 2nd international conference on automated guided vehicle systems, Stuttgart Germany, pp 21–30

Em-plant software. http://www.emplant.de/english/index.html

Haines CL (1985) An algorithm for carrier routing in a flexible material-handling system. IBM J Res Dev 29(4):356–362

Huang J, Palekar W (1989) A labeling algorithm for the navigating. In: Advanced manufacturing system engineering: proceedings of the ASME Winter meeting, San Francisco, California, pp 181–193

Kim KH, Jeon SM, Ryu KR (2006) Deadlock prevention for automated guided vehicles in automated container terminals. OR Spectrum 28:659–679

Kim CW, Tanchoco JMA (1990) Prototyping the integration requirements of a free-path AGV system. In: Proceedings of the 1990 material handling research colloquium, Hebron KY, pp 355–363

Kim CW, Tanchoco JMA (1997) Conflict-free shortest-time bidirectional AGV routeing. Intern J Prod Res 29(12):2377–2391

Lim JK, Lim JM, Yoshimoto K, Kim KH (2002) A construction algorithm for designing guide paths of automated guided vehicle systems. Intern J Prod Res 40(15):3981–3994

Lim JK, Kim KH (2002) Dynamic routing in automated guided vehicle systems. JSME Intern J 45 (1):323–332

Liu C-I, Jula H (2004) Automated guided vehicle system for two container yard layouts. Transp Res Part C 12:349–368

Maza S, Castagna P (2005) A performance-based structural policy for conflict-free routing bi-directional automated guided vehicles. Comput Ind 56:719–733

Majety SV, Wang MH (1995) Terminal location and guide path design in terminal based AGV systems. Intern J Prod Res 33(7):1925–1938

Rajotia S, Shanker K, Batra JL (1998) A semi-dynamic time window constrained routeing strategy in an AGV system. Intern J Prod Res 36(1):35–50

Rajeeva LM, Wee HG (2003) Cyclic deadlock prediction and avoidance for zone-controlled AGV system. Intern J Prod Econ 83:309–324

Walker SP, Premi SK (1985) The imperial college free-ranging AGV (ICAGV) and scheduling system. In: Proceeding of the 3rd international conference on automated guided vehicle systems, Stockholm Sweden, pp 189–198

Zeng J, Hsu WJ (2008) Conflict-free container routing in mesh yard layouts. Robot Auton Syst 56:451–460

http://www.springer.com/978-3-319-45116-9

Dynamics in Logistics
Proceedings of the 5th International Conference LDIC, 2016 Bremen, Germany
Freitag, M.; Kotzab, H.; Pannek, J. (Eds.)
2017, XI, 502 p. 125 illus., 70 illus. in color., Hardcover
ISBN: 978-3-319-45116-9