Logist. Res. (2011) 3:19–27
DOI 10.1007/s12159-010-0042-5

ORIGINAL PAPER

Routing automated guided vehicles in container terminals through the Q-learning technique

Su Min Jeon · Kap Hwan Kim · Herbert Kopfer

Received: 23 August 2010 / Accepted: 7 December 2010 / Published online: 28 December 2010

Abstract This paper suggests a routing method for automated guided vehicles in port terminals that uses the Q-learning technique. One of the most important issues for the efficient operation of an automated guided vehicle system is to find shortest routes for the vehicles. In this paper, we determine shortest-time routes inclusive of the expected waiting times instead of simple shortest-distance routes, which are usually used in practice. For the determination of the total travel time, the waiting time must be estimated accurately. This study proposes a method for estimating for each vehicle the waiting time that results from the interferences among vehicles during travelling. The estimation of the waiting times is achieved by using the Q-learning technique and by constructing the shortest-time routing matrix for each given set of positions of quay cranes. An experiment was performed to evaluate the performance of the learning algorithm and to compare the performance of the learning-based routes with that of the shortest-distance routes by a simulation study.

S. M. Jeon (✉)
Seaport Innovations, Schiessstattstrasse 4,
5580 Tamsweg, Austria
e-mail: sj@seaportgroup.eu

K. H. Kim
Department of Industrial Engineering,
Pusan National University, Jangjeon-dong,
Kumjeong-ku, Busan 609-735, Korea
e-mail: kapkim@pusan.ac.kr

H. Kopfer
Lehrstuhl für Logistik, FB 7, University of Bremen,
28334 Bremen, Germany
e-mail: kopfer@uni-bremen.de

Keywords AGV · Reinforcement learning · Shortest pats · Estimation of waiting times · AGV · Container terminal

1 Introduction

Generally, ship operations at container terminals consist in unloading and loading operations that are performed by specific handling equipments. We assume that three types of equipments are used for these ship operations such as Quay Cranes (QCs), Automated Guided Vehicles (AGVs), and Automated Yard Cranes (AYCs).

- Unloading operation: When a ship arrives at a container terminal, the import containers are lifted by QCs and handled over to an AGV. The AGV is used for the transport of the container to the Transfer Point (TP) of the storage yard. An AYC picks up the container from the AGV and stacks it to the storage place. Figure 1 illustrates the process of the ship operation in an Automated Container Terminal (ACT).
- Loading operation: With respect to AGV-routing, this operation starts when an AGV has arrived at the TP of the storage yard and an AYC has picked up an export container from the storage yard and has put it on the AGV. The export container is carried by an AGV to the appropriate QC which will lift the container to the ship.

Figure 1 and the above operations point out that AGVs realize the important link between QCs and AYCs. When a transportation request across this link arises, the AGV control system is started to execute its tasks such as dispatching, routing or scheduling, and traffic control. According to Kim and Tanchoco [4], the primary vehicle-management functions for AGVs can be defined as follows:

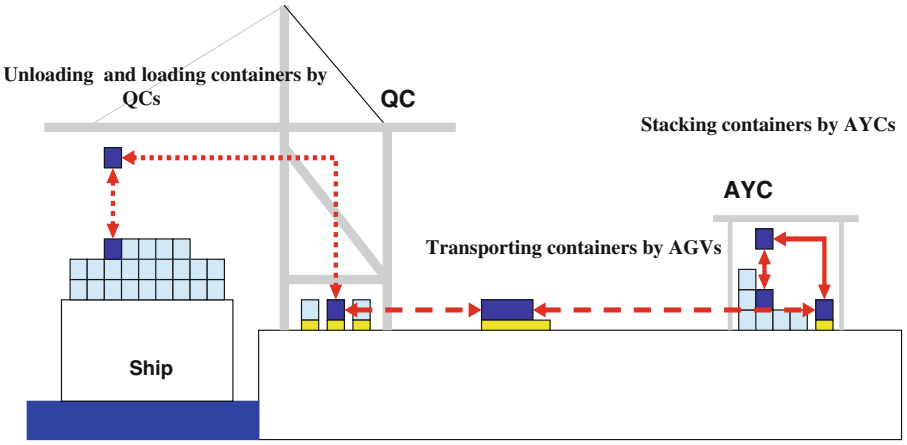

Fig. 1 An illustration of ship operation

- Dispatching is the process of selecting and assigning tasks to vehicles.
- Routing is the selection of the specific paths taken by vehicles to reach their destinations.
- Scheduling is the determination of the arrival and departure times of vehicles at each segment along their prescribed paths to ensure collision-free journeys.
- Traffic control for AGVs is generally representing zone control. Zone control uses a wireless transmitter to transmit a signal in a fixed area. Each AGV contains a sensing device to receive this signal, which is used for collision avoidance with other AGVs.

The routing function is one of the most important components of an AGV control system in automated container terminals (ACTs). The routing function selects a specific path for a vehicle to be followed in order to reach its destination from the present position. In static routing systems, a vehicle is given a predetermined route from its starting position to its destination. Usually, shortest-distance routes are provided to vehicles because efficient terminal operations are aspired. This results in an easy control method, but it does not guarantee for the efficiency of the vehicle operations because mutual influence between the vehicles and waiting times are ignored. For this reason, we present a method for finding shortest paths with respect to travel times including the expected waiting times of the vehicles.

The AGV routing problem has been addressed by several researchers. The conceptual foundations of the AGV routing problem were first laid down by Broadbent et al. [1]. They proposed an AGV scheduler that uses Dijkstra's shortest-path algorithm and generates a timetable that contains the node occupation times for each vehicle. A study by Gaskins and Tanchoco [3] suggested an integer-programming model to determine the directions of path segments on a guide-path so that the total travel distance of vehicles is minimized. Kim and Tanchoco [4] used the concept of time-window graph, which is a directed graph of the free time-windows, for finding the shortest-time route on bidirectional guide-path networks. Rajotia et al. [10] proposed a semi-dynamic time-window routing strategy, the principle of which is quite similar to the path-planning method of Kim and Tanchoco [4]. Time-windows that model the traffic-flow direction are placed on bidirectional arcs, which can be crossed only in one direction at a time. On the basis of these time-windows, the Dijkstra algorithm was applied to find the least congested and fastest routes for vehicles. Oboth et al. [8] addressed operational control problems such as demand assignment and route planning. They proposed a route-generation procedure called the sequential path generation (SPG) heuristic. Lim et al. [5] applied a Q-learning method [6, 7] to estimate the expected travel time of a vehicle on path segments for designing guide-paths in automated guided vehicle systems. An early survey on scheduling and routing algorithms for AGV's can be found in Qiu et al. [9].

This study applies the Q-learning method for route planning for AGVs in ACTs. This paper attempts to find the shortest-time route instead of the shortest-distance route by considering the congestion at intersections and bidirectional path segments for a given set of transportation requests. To find the shortest route in terms of the travel time, the expected delay time of vehicles at each intersection is estimated by utilizing the travel experiences gained during a simulation-based learning process.

The rest of this paper is structured as follows. Section 2 introduces a guide-path network for AGVs, which is assumed in this study, and explains the routing problem. Section 3 describes how to apply the Q-leaning method to find the shortest-time routes, Sect. 4 describes simple traffic control for AGVs. On the basis of a simulation study, and Sect. 5 demonstrates the performance of the presented approach by comparing the routes generated by the proposed learning method with the shortest-distance

routes. Finally, the conclusions and a summary are provided in Sect. 6.

2 AGV path network and the routing problem

In ACTs, there are two types of guide-path networks used in practice which are the closed-loop type and the cross-lane type. The networks of the closed-loop type have several large circular guide-paths for vehicles to follow during travel. This allows a simplified control of vehicles but requires long travel distances for vehicles.

To speed up the deliveries of containers by AGVs, the more complicated guide-path networks of cross-lane type have been developed and applied in ACTs. On the cross-lane networks, which are used in practice, a vehicle moves through shortcuts to travel on the most attractive route from its origin to its destination. That is why cross-lane guide-path networks can significantly reduce the travel distances of AGVs, but the traffic control of vehicles becomes much more complicated.

This study assumes a guide-path network of the cross-lane type, as illustrated in Fig. 2. This network is also used for the simulation experiments presented in this paper.

Figure 2 shows a layout of the guide-path network with yard blocks that are laid out in a perpendicular direction to the berth. In this network, a node corresponds either to a point of intersection of path segments or to a pickup and delivery (P/D) point for QCs or to transfer points (TPs) for AYCs in front of the blocks. An arc in the network represents the direction of travel that is allowed for AGVs. The definition of the travel direction on each arc is an important decision to be made with regard to the design of the guide-path network. The layout in Fig. 2 provides five lanes under QCs, including three lanes that are allocated for transferring containers with QCs, while the other two lanes are only for the running of vehicles. There are 23 intersections for each lane under the QCs, i.e. one intersection for each bitt. All the lanes under QCs point in the same direction. Some lanes in the parking area are directed toward the berth, while some lanes are directed toward the yard. There are seven lanes for the running of vehicles in the area between the parking zone and the TPs. The TPs are in front of the blocks, and each block has three TPs.

Figure 3 illustrates a simplified guide-path network with nine nodes and given travel times on the arcs of the graph. Table 1 shows a route matrix R for this network indicating a path between any two nodes. For a path from a starting node i to the destination node j, the immediate successor of i is given by R(i,j), the successor of the successor is given by R(R(i,j),j), and so on until the final destination j is reached. This paper suggests a method for AGV routing by

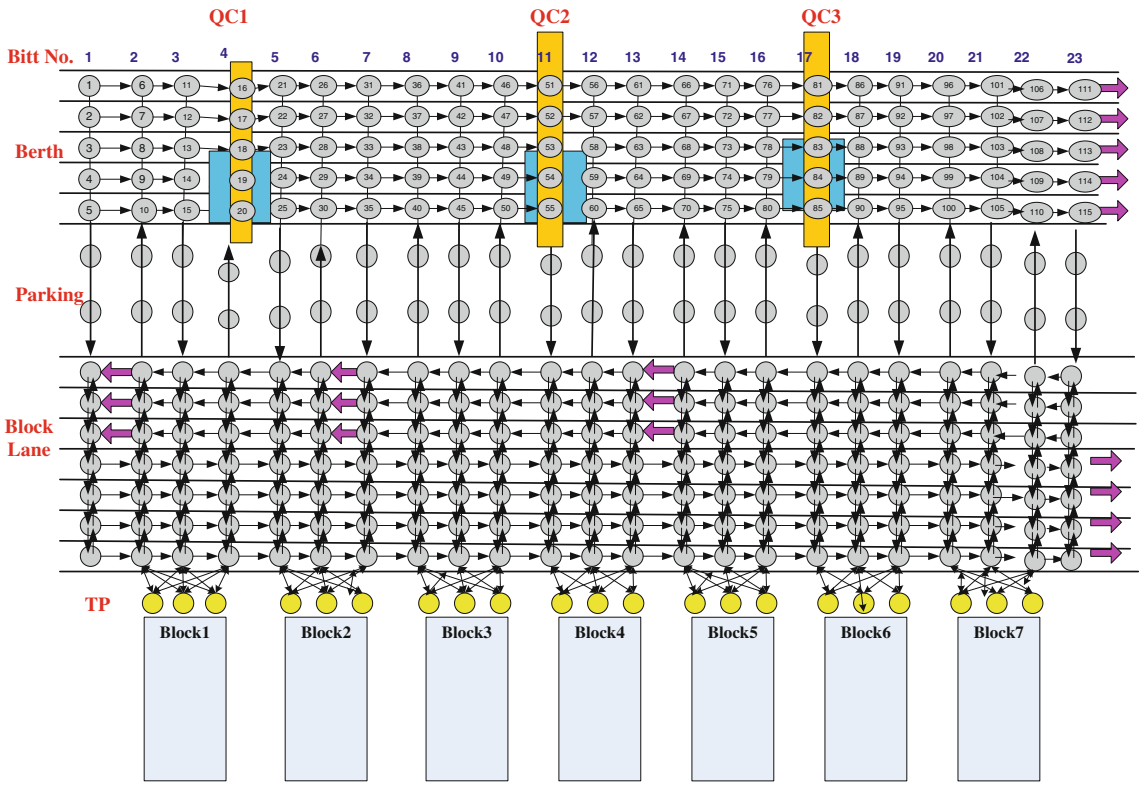

Fig. 2 An example of a guide-path network

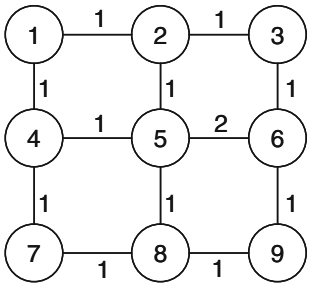

Fig. 3 An example of a simplified guide-path network

generating the route matrix as illustrated in Table 1, which minimizes the total travel time for a given set of transportation requests.

3 Application of the Q-learning technique for routing AGVS

This section discusses how to construct the routes for vehicles by the Q-learning technique. Q-learning is a form of reinforcement learning technique. Reinforcement learning is a process of learning how to match situations with actions in order to maximize a numerical reward signal. The learners are not told what actions to take, as in the case of most machine-learning approaches. Instead, learners must discover by trial and error which actions yield the highest reward. The four sub-elements of reinforcement learning are policy, reward function, value function, and model of the environment. The following describes how the Q-learning technique can be applied to the routing of vehicles [5].

A policy defines the learning agent's way of behaving at a given time. The policy is the core of a reinforcement learning agent in the sense that it alone is sufficient to determine the agent's behavior. In general, policies may be stochastic. The reward function defines what the good or bad events are for the agent which is related to the goal of the problem. However, because the objective of the problem is to minimize the travel time, a penalty function will be used instead of a reward function. A value function specifies what is good or bad in the long run. The value of a state is the total amount of reward or punishment an agent can expect to accumulate over the future, starting from that state. A model of the environment is something that mimics the behavior of the environment (Richard and Andrew 1998). In the shortest path problem of this study, a *state* is defined by the current location of a vehicle and its destination and an *action* is defined as the next immediate node to be selected.

Table 1 Route matrix R for the example in Fig. 3

Current node	Destination node					
	1	2	3	7	8	9
1	1	2	2	4	2	2
2	1	2	3	1	5	5
3	2	2	3	2	6	6
4	1	5	5	7	7	7
5	4	2	2	8	8	6
6	3	3	3	5	5	9
7	4	4	8	7	8	8
8	7	5	9	7	8	9
9	8	8	6	8	8	9

The following notation is introduced to describe the learning process.

t	The destination node of a current vehicle
k	The node where a current vehicle is located
(k,t)	The state of the vehicle, which consists of its current node (k) and its destination node (t)
$A(k,t)$	The set of candidates for the next node (action), from which a vehicle in state (k,t) can choose
a	An action taken by a vehicle, which is an element in $A(k,t)$ The action corresponds to the next node that is selected
γ	The discount factor for future penalties ($0 \leq \gamma \leq 1$)
$r[(k,t),a]$	The penalty, which is the travel time of a vehicle, at state (k,t), from the current node to the next node (a). The travel time may also include the waiting time that is caused by traffic congestion
$Q[(k,t),a]$	The expected discounted cumulative travel time of a vehicle, which is at state (k,t) and selects action a, from the current node to the destination node

The travel time from a node to the next immediate node will be a penalty for the corresponding state-action pair. The value function specifies how to evaluate decisions in the long run, whereas a reward function indicates how good a decision is in the immediate future. For this study, the total travel time from a start node to a destination node will be the value function. For obtaining the value for element (k,t) in the route matrix, it is obvious that the state-action pair with the lowest value of $Q[(k,t),a]$ must be adopted as the value for element (k,t) of the route matrix. Thus, in order to obtain the route matrix, it is necessary only to estimate the value of $Q[(k,t),a]$. The value of $Q[(k,t),a]$ is estimated by a simulation-based learning procedure as follows. Figure 4 describes the simulation-based learning procedure by tracing a single vehicle.

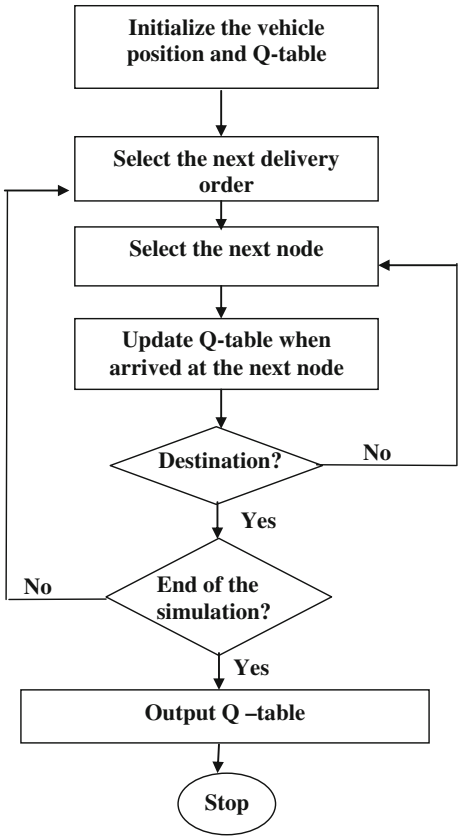

Fig. 4 The procedure of the simulation-based learning for one vehicle

3.1 Initialize the vehicle position and the Q-table

All the vehicles are positioned at parking lanes in the beginning and the initial value for element (k,t) in the Q-table is set to be a very large value for each action a.

3.2 Select the next delivery order

When a vehicle becomes idle, a delivery order is generated through the origin–destination probability matrix and the order is assigned to the vehicle. The (origin, destination) pair of the order is either (a TP at a block, a lane under a QC) or (a lane under a QC, a TP at a block).

3.3 Select the next node

Note that the Q-table is updated during the simulation. When a vehicle, whose final destination is t, selects the next node at a node k, a conditional probability of selecting node (action) a, given state (k, t), is used, which is calculated as follows [7]:

$$p(a|(k,t)) = \frac{\rho^{\hat{Q}[(k,t),a]^{-1}}}{\sum_{a\in A(k,t)} \rho^{\hat{Q}[(k,t),a]^{-1}}}, \quad (1)$$

where $a \in A(k, t)$ and ρ is a positive constant and $\hat{Q}_n[(k,t),a]$ is an estimated value of $Q[(k,t),a]$.

$Q[(k,t),a]$ has different values for varying candidates for the next node, a, for the same state that is represented by the tuple (k,t) for the current and destination nodes. For efficiently using the computational time, more effort must be devoted to estimating lower values of $Q[(k,t),a]$ among various actions, which expression (1) attempts to do. According to (1), when the value of $Q[(k,t),a]$ for a candidate node a is smaller than that for another candidate node a' at a given state, the probability of selecting node a becomes higher than that of selecting node a' as the next node. As a result, more samples will be collected from the experience of vehicles that travel from the current node to the destination through node a than from that through node a'. This implies that the value of $Q[(k,t),a]$ can be estimated in a greater accuracy than that of $Q[(k,t),a']$.

3.4 Update the Q-table when the current vehicle arrives at the next node

The following equation is used for updating the value of $\hat{Q}_n[(k,t),a]$, which is an estimator of $Q[(k,t),a]$, during the simulation.

$$\hat{Q}_n[(k,t),a] = (1-\alpha_n)\hat{Q}_{n-1}[(k,t),a] + \alpha_n\{r[(k,t),a] + \gamma \min_{a'} \hat{Q}_{n-1}[(a,t),a']\} \quad (2)$$

In (2), $\alpha_n = \frac{1}{1+\text{visits}_n[(k,t),a]}$ and $\text{visits}_n[(k,t),a]$ represents the total number of visits to the state-action pair $[(k,t),a]$ during the entire learning process. As the number of visits to the same pair of $[(k,t),a]$ increases, the value of α_n approaches zero, and thus, a larger value of the weight is given to the previous estimate of $Q[(k,t),a]$, which is $\hat{Q}_{n-1}[(k,t),a]$. As a result, the value of $\hat{Q}_{n-1}[(k,t),a]$ is stabilized. The discount factor γ has a value between 0 and 1, which makes that the value of $Q[(k,t),a]$ converges to a finite value.

3.5 Stopping conditions

The value of $\hat{Q}[(k,t),a]$ is updated by using expression (2), whenever a vehicle arrives at a node on the way to the destination. If the change of a $\hat{Q}$ is smaller than a prespecified tolerance, ε, then the count of the stability is increased by one. If the change of a $\hat{Q}$ is greater than or equal to ε, then the count of the stability is reset to zero. When the

count exceeds a pre-specified limit, called "maximum counter of stability", the learning process is stopped.

Through the simulation study, it was found that as the simulation run for the learning is continued, the value of $\hat{Q}[(k,t),a]$ is stabilized, as shown in Fig. 5. By using the final values of $\hat{Q}[(k,t),a]$, the route matrix was constructed by inserting $a^* = \arg\min_a \hat{Q}[(k,t),a]$ into element (k,t) of the route matrix. This route matrix will be used for determining the travel route of a vehicle for each delivery request.

For implementing the learning process, a simulation model for the layout in Fig. 2 was developed by using eM-plant 7.6 version. Deadlock situations were found during the experiments and therefore, some rules for resolving the deadlock situations are provided. The deadlocks were observed when requests of vehicles for the next nodes formed a cycle. For guaranteeing deadlock-free travel of AGVs during the simulation runs, traffic control rules discussed in Sect. 4 are needed.

4 Traffic control rules used to support the learning process

During the learning process, it is necessary that traffic control rules must be provided for guaranteeing conflict and deadlock-free travel of AGVs. In the following, several deadlock situations are described, which were found during the experiments and thus some rules for resolving the deadlock situations must be provided. Figure 6 illustrates a possible deadlock situation, in which a cyclic deadlock may occur; i.e. requests of vehicles for the next nodes form a cycle of nodes. Figure 6 shows four vehicles from AGV1 to AGV4, and the shaded nodes are the current locations of the vehicles. Each arrow represents the claim of a reservation for the next node for the travel by a vehicle on a node. For example, the vehicle on node 1 is claiming node 2 for the travel. However, if AGV1 is allowed to enter node 2, there will be a cyclic claim for the next node, which means a deadlock.

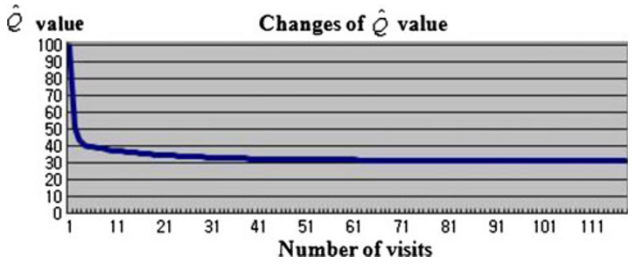

Fig. 5 The stabilization of the $\hat{Q}$-value

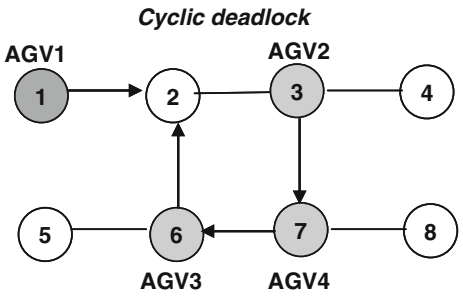

Fig. 6 A cyclic deadlock situation on block lanes

There are two areas with high possibilities of deadlock situations. The first one is the area of block lanes that consist of driving lanes of opposite directions or bidirectional lanes. The other area is between the lanes of the berth and those in front of blocks, as shown in Fig. 2.

We use the 'Semaphore' concept [2] to prevent automated guided vehicle systems from cyclic deadlock situations. A semaphore is a classical solution to prevent resource deadlock. Whenever a vehicle arrives at a semaphore area, the counting semaphore is triggered to check the availability of resources. The control logic of the counting semaphore can be defined by the following procedure of 'wait and proceed.'

4.1 Wait and proceed

When a vehicle predicts a deadlock on its route, the vehicle stops at its entry location and waits until at least one vehicle gets cleared from the predicted deadlock region.

Wait if the capacity of semaphore is the same as the number of resources occupied, then wait until the occupied number of resources becomes smaller than the capacity.

Proceed if the capacity of semaphore is greater than the number of occupied resources, then proceed to this semaphore area and update the number of occupied resources. After proceeding, the number of occupied resources increases by one.

A semaphore area *SPi* is a set of nodes for which vehicles can request a reservation. In some cases, there may be nodes in an overlapped area affected by more than one semaphore area. Semaphore areas with the capacity of 4 are illustrated in Fig. 7. When, for example, a vehicle arrives at node 3 and the next visiting node is node 4, this will trigger the counting of the semaphore areas SP1 and SP2. The set of nodes in SP1 is {1,2,3,4}, and the set of nodes in SP2 is {3,4,5,6}. Only when the numbers of reserved resources in both sets are smaller than the capacities of the two semaphore areas, which are 3, then the vehicle can enter the node; otherwise, the vehicle must wait until the conditions are satisfied.

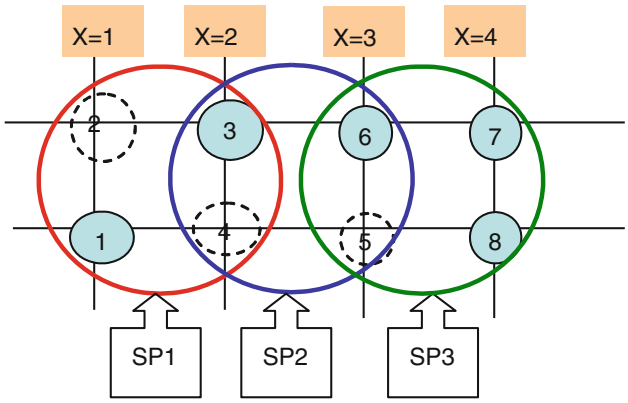

Fig. 7 An example of semaphore areas in block lanes

Figure 8 illustrates a head-to-head conflict on a bidirectional path segment. In the simulation of this study, when a conflict of this type was detected, a detour, instead of the original route, was selected for avoiding this conflict.

5 Simulation study

5.1 Simulation scenario

A simulation program was developed by using the Em-plant® simulation package for the learning process. The simulation program was run on a Pentium IV processor of 3 GHz clock-speed and 1 GB memory. A container terminal with one berth of the length of 360 m, three QCs, and seven storage blocks, each of which had three transfer points, was modeled. The guide-path network corresponded to that in Fig. 2. The size of a node was assumed to be large enough to cover a vehicle. The length of each node was assumed to be 16 m. The total number of nodes in the terminal was 320. Five vehicles were assigned to each QC. During the simulation, every vehicle was dedicated to a single QC.

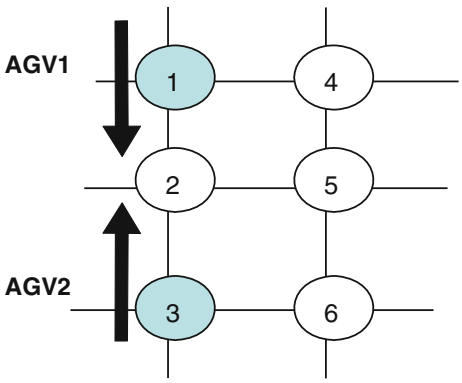

Fig. 8 Head-to-head conflict at bidirectional path

Nine scenarios of the delivery requirements, as shown in Table 2, were used for evaluating the approach of this paper. Each scenario has different positions of QCs or a different assignment of blocks to QCs. In scenarios 1 to 4, the positions of QCs are equal, but the blocks for containers that are to be discharged differ across these scenarios. However, in scenarios 5 to 9, containers are distributed uniformly over all the seven blocks, but the positions of QCs differ across the scenarios. Each QC discharges 150 containers and is assigned five AGVs. The operation cycle time of QCs and YCs are assumed to be 120 and 60 s, respectively. In the experiments, the value of ε was set to be 0.01. The maximum counter of stability was set to 3000. The value for the discount factor γ was set to 0.9 in all experiments.

5.2 Simulation results

The average travel time of vehicles using the routes obtained from the learning method was compared with that from the shortest-distance routes that are the most popular in practice. Table 3 shows the ratio of the travel time from the learning method to that from the shortest-distance routes. The travel time consists of the moving time and the waiting time. It was found that the learning-based routing method outperformed the shortest travel distance route with regard to the average travel time by 17.3%.

Table 4 shows the computational time for the learning and the average travel time of vehicles per trip for different sets of the stopping parameters, which are values of the maximum counter of the stability and ε. Scenario 1 was used for the results in Table 4. Note that the computational time becomes longer and the average travel time becomes shorter as the maximum counter becomes larger and ε becomes smaller. Thus, we assumed that the case with the smallest average travel time in Table 4 defines the reference value of 100%.

6 Conclusions

This study applied the Q-learning algorithm to AGV routing in port container terminals. The method in this paper attempts to find routes with the shortest travel time for each delivery order. It was shown how the Q-learning method can be used to estimate the expected travel time of vehicles between two nodes in a guide-path network. For the implementation of the simulation model, deadlock-avoidance strategies were developed and embedded in the simulation program.

Through a simulation study, the performance of the learning algorithm was compared with that of the approach of shortest-travel-distance routes. It was shown that the

Table 2 Flow requirements used in the simulation study

Scenarios	QC ID—Position	Blocks assigned to each QC	Scenarios	QC ID—Position	Blocks assigned to each QC
1	1–4*	1, 2, 3	6	1–6	1–7
	2–11	3, 4, 5		2–12	1–7
	3–18	5, 6, 7		3–18	1–7
2	1–4	3, 4	7	1–6	1–7
	2–11	3, 4, 5		2–12	1–7
	3–18	3, 4, 5		3–15	1–7
3	1–4	4, 5	8	1–9	1–7
	2–11	4, 5		2–11	1–7
	3–18	3, 4		3–14	1–7
4	1–4	3, 4	9	1–10	1–7
	2–11	3, 4		2–14	1–7
	3–18	3		3–18	1–7
5	1–6	1 ~ 7	*Bitt number		
	2–10	1 ~ 7			
	3–16	1 ~ 7			

Table 3 A comparison between the travel times for the Q-learning-based routes and the shortest-distance routes

Scenario	Travel time by Q-learning/travel time on the shortest travel distance method (%)
1	96.80
2	93.38
3	93.98
4	92.52
5	75.80
6	80.60
7	75.40
8	69.90
9	65.90
Average	82.70

Table 4 Comparison of the average travel time per delivery for different sets of stopping parameters in Q-learning

Stopping parameters		Computational time for learning (min)	Average travel time (%)
Maximum counter	ε		
3,000	0.01	225	101.89
	0.03	210	102.02
	0.05	120	103.04
4,000	0.01	230	100.18
	0.03	212	101.12
	0.05	126	102.54
5,000	0.01	245	100.00
	0.03	220	100.18
	0.05	150	101.20

travel time can be reduced by 17.3% by using the learning-based routes instead of the shortest-distance routes. It was also found that the parameters for the stopping rule affect the quality of the routes and the computational time for the learning. Thus, it is necessary to find appropriate levels of the parameters of the stopping rules.

For future research, the time for learning must be reduced further so that the approach in this paper becomes more applicable in practice. As the structure of the guide-path network becomes more complicated, the possibility of deadlocks becomes higher. Thus, more studies on methods for preventing deadlocks must be conducted in future.

Acknowledgments This study was partially supported by the Korean-German international symposium program of KOSEF in Korea and DFG in Germany and by a Korea Research Foundation Grant that was funded by the Korean Government (MOEHRD) (The Regional Research Universities Program/Institute of Logistics Information Technology).

References

1. Broadbent AJ, Besant CB, Premi SK, Walker SP (1985) Free ranging AGV systems: promises, problems and pathways. In: Proceeding of the 2nd international conference on automated materials handling, pp 221–237
2. Evers JJM, Koppers SAJ (1996) Automated guided vehicle traffic control at a container terminal. Transp Res A 30(1):21–34
3. Gaskins RJ, Tanchoco JMA (1987) Flow path design for automated guided vehicle systems. Int J Prod Res 25(5):667–676
4. Kim CW, Tanchoco JMA (1991) Conflict free shortest time bi-directional AGV routing. Int J Prod Res 29(12):2377–2391
5. Lim JK, Lim JM, Yoshimoto K, Kim KH, Takahashi T (2002) A construction algorithm for designing guide paths of automated guided vehicle system. Int J Prod Res 40(15):3981–3994

6. Mahadevan S (1996) Average reward reinforcement learning; foundation, algorithms, and empirical results. Mach Learn 22(1):159–195
7. Mitchell TM (1997) Machine learning. McGraw-hill, New York
8. Oboth C, Batta R, Karwan M (1999) Dynamic conflict free routing of automated guided vehicles. Int J Prod Res 37(9):2003–2030
9. Qiu L, Hsu WJ, Huang SY, Wang H (2002) Scheduling and routing algorithms for AGV's: a survey. Int J Prod Res 40(3):745–760
10. Rajotia S, Shanker K, Batra JL (1998) A semi-dynamic time window constrained routing strategy in an AGV system. Int J Prod Res 36(1):35–50