Journal of the Society of Korea Industrial and Systems Engineering
Vol, 31, No. 3, pp.24－34, September 2008.

존 조정의 양방향 자동반송차량 네트워크에서 고착 예방 및 방지를 위한 효과적인 알고리듬

임 동 순†

한남대학교 공과대학 산업경영공학과

Efficient Algorithms for Preventing and Avoiding Deadlocks in Zone-controlled Bi-directional AGV Networks

Dong-Soon Yim†

Department of Industrial and Management Engineering, Hannam University

본 논문은 존 조정하에서의 자동반송차량 네트워크에서 발생하는 고착을 해결하기 위한 두가지 효과적인 알고리듬을 소개한다. 이 알고리듬들은 특별히 양방향 네트워크에 알맞도록 고안되었다. 사이클 제거 알고리듬은 고착을 예방하기 위한 차량의 안전한 라우트를 결정하나, 그래프 축소 알고리듬은 고착을 회피하기 위하여 미래 잠재적인 고착 발생 조건을 결정한다. 시뮬레이션을 통하여 알고리듬들의 성능을 비교 분석한 결과 작업물의 이동 횟수와 알고리듬의 시간 복잡성 측면에서 그래프 축소 알고리듬이 사이클 제거 알고리듬 보다 우수함을 나타내었다.

Keywords : AGV(Automated Guided Vehicle), Deadlocks, Graph

1. Introduction

Automated guided vehicles (AGVs) have been widely used for material handling in modern manufacturing systems, distribution systems, and container terminals. A significant phenomenon in designing and operating AGV systems that must be handled carefully is vehicle conflict.

To avoid conflicts, zone control has been commonly implemented. Zone control in an AGV system is a control method that divides an AGV path network into non-overlapping zones and allows at most, one vehicle at a time in each zone to prevent conflicts among several vehicles. When zone control is employed, a safe route for vehicles that guarantees freedom from conflict will be obtained whenever a new route for a vehicle is determined.

The determination of safe routes can be classified into two methods according to whether or not the travel time is considered. Kim and Tanchoco [3] considered the deterministic traveling time in a bi-directional path network. Their method finds the shortest path that guarantees conflict-free routes. Oboth et al. [6] proposed a method of determining conflict-free routes from a graph model in which the intersection of the path network is represented as a vertex, and the lane between intersections is represented as an edge. Reveliotis [8] proposed a route planning method without considering travel time. This method determines a safe route using a graph model representing a bi-directional path network under zone control. In particular, they used the modi-

논문접수일 : 2008년 05월 06일 논문수정일 : 2008년 06월 17일 게재확정일 2008년 06월 17일
† 교신저자 dsyim@mail.hannam.ac.kr
※ 이 논문은 2006년도 정부(교육인적자원부)의 재원으로 한국대학교육협의회 대학교수 국내교류 연구비 지원에 의한 것임.

fied Banker's algorithm to test the safeness of vehicle routes. Although their method is free of system uncertainty because it does not consider travel time, it is conservative in nature.

In AGV systems using zone control, a deadlock can occur because of inherent constraints incurred in the zone control. A deadlock in zone control refers to a circular waiting state caused by traveling vehicles. In addition to determining safe routes, another method of avoiding deadlocks under zone control is to predict the possibility of future deadlocks, i.e., impending deadlocks (Wysk et al. [9]) and to make the vehicle wait at the current zone until there is no longer any possibility of a deadlock. Lee and Lin [4] proposed a deadlock avoidance method using a Petri net model representing uni-directional AGV path networks. When a vehicle must be moved to the next zone, the possibility of a deadlock is examined using a reachability analysis of a Petri net model. Yeh and Yeh [10] also examined the possibility of a deadlock using a graph model representing a uni-directional path network. These methods under zone control examine only the possibility of a circular waiting state in a uni-directional AGV path network. An additional deadlock can be caused, however, by restricting the vehicle's movement to the next zone. Banazak and Krogh [1] referred to this type of deadlock as a restricted deadlock.

It is observed that most research on AGV deadlocks caused by a circular waiting state has not considered the restricted deadlock. This is mainly due to the simple uni-directional AGV path network that has been considered. Recent papers by Fanti [2] and Rajeeva et al. [7] mentioned this aspect by referring to second-level-cycle and multi-cycle deadlock situations, respectively. Simple-cycle deadlocks and second-level-cycle or multi-cycle deadlocks caused by restricting the vehicles' movement in each simple cycle are considered. It is not difficult to discover, however, that restricting the vehicles'movement in a second-level cycle or a multi-cycle will cause another deadlock. Deadlock detection using the CDG (capacitated directed graph) model proposed by Yim et al. [11] demonstrates this kind of incremental deadlock. They used a hierarchical procedure of cycle reduction to detect all restricted deadlocks. However, their method has a conservative characteristic in that it does not explicitly specify vehicle routes in the graph model. Even if future system states will not result in deadlocks, the deadlock avoidance policy based on CDG may presume that there will be a possibility of a deadlock. This conservativeness may often cause the deterioration of the performance of the considered system.

This work introduces efficient algorithms for handling deadlocks in zone-controlled bi-directional AGV path networks. Even though this work is inspired by previous works on safe routes and deadlock prediction, more robust and efficient algorithms were developed. Throughout the extensive simulation study conducted, the proposed algorithms were evaluated.

2. Zone-controlled bi-directional path network

<Figure 1> shows the generic AGV path network considered in this paper. The network includes 30 P/D (pickup/delivery) stations connected by bi-directional guided paths. All its 91 zones are specified in each intersection and in front of each P/D station. A vehicle is allowed to move along the guided path between adjacent zones only if the next zone is clear.

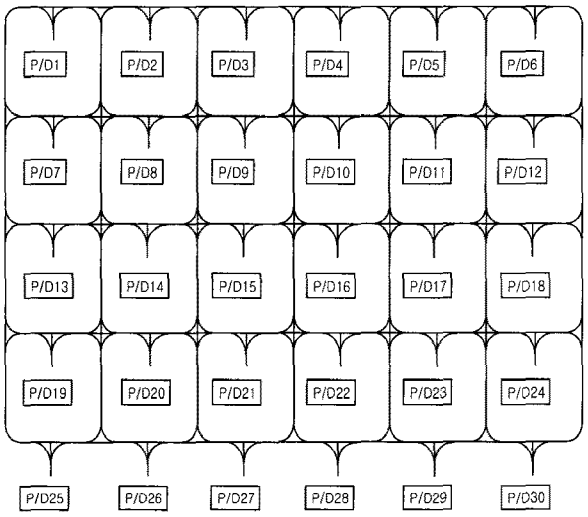
Since all the guided paths are bi-directional, the occurrence of conflicts and deadlocks is inevitable. Three strategies are commonly used to resolve these occurrences : deadlock prevention, deadlock avoidance, and deadlock recovery. Among these, avoidance based on real-time prediction of deadlocks is known to result in better resource utilization (Leung and Shen [5]). In this paper, prevention and avoidance of deadlocks are considered.

To prevent deadlocks in the given bi-directional path network, the main concern is the determination of a safe route for vehicles. The algorithm developed by Kim and Tanchoco [3] requires an $O(v^4n^2)$ time complexity, in which v is the number of vehicles and n is the number of zones. The method of Oboth et al. [6] ensures that only one vehicle occupies a path segment. Even though there will be no conflicts in this method, a cyclic deadlock can occur. Fanti [2] proposed a different approach to the determination of a safe route in bi-directional path networks. Instead of determining a safe route directly from the system states, a proposed route is checked through path validation to see if an impending deadlock and a restricted deadlock are foreseen. This procedure is repeated until a safe route is obtained or no more paths are proposed. The complexity of the algorithm needed to check the safeness of a path is $O(n^3)$. More complexity will be incurred since repetition is necessary. In the next section, an efficient algorithm is developed to determine a safe route

without time-consuming safeness validation.

To avoid deadlocks, real-time deadlock prediction is required. From these prediction results, the movement of vehicles is restricted to avoid future deadlocks. A vehicle can move to the next zone only if all deadlock possibilities are removed. From the literature review conducted, it was observed that research on the prediction of all types of deadlocks, including impending and restricted deadlocks, in AGV path networks consider only simple-cycle and multi-cycle deadlocks. At first, vehicle movement is restricted so as not to cause a cyclic deadlock. Since this control will also cause a restricted deadlock, a multi-cycle deadlock is considered. A common method of handling multi-cycle deadlocks is producing a safe route for a deadlock-causing vehicle (Fanti [2], Rajeeva et al. [7]).

This paper introduces an efficient real-time algorithm to predict deadlocks. This algorithm does not require new route determination to avoid multi-cycle deadlocks. Instead, the pre-computed shortest path in the network layout will always be given to vehicles as a route. In this environment, the prediction of all restricting deadlocks is indispensable.



<Figure 1> A bi-directional AGV path network

3. Deadlock prevention based on the cycle elimination algorithm

One method of preventing deadlocks is ensuring that a new route to be determined for an idle vehicle is safe. The cycle elimination algorithm proposed in this section is developed to provide such a route. Instead of exploiting time-consuming safeness validation procedure, the algorithm produces a graph from which a safe route is directly obtained. The following notations are used to describe the algorithm.

N_A : the number of vehicles in the system
$R_i = \{z_{i0}, z_{i1}, \cdots, z_{\in_i}\}$: current route of vehicle A_i
$G(V, E)$: directed graph representing an AGV path network
$G'(V, E')$: directed graph representing current routes
$G''(V, E'')$: directed graph to determine a safe route

It is assumed that the information on current routes of all N_A vehicles is available. A route consists of a sequence of zones to visit. Therefore, z_{i0} in the route denotes the zone in which vehicle A_i is currently positioned. For an idle vehicle A_k, the route contains only one zone, i.e., $R_k = \{z_{k0}\}$. The directed graph $G(V, E)$ that is an input to the algorithm is a weighted graph in which V represent all zones and E represent paths between adjacent zones in an AGV path network. Also, each edge has an attribute of distance or travel time. Two directed graphs, $G'(V, E')$ and $G''(V, E'')$, are created in the algorithm to find a safe route for an idle vehicle.

Whenever an idle vehicle A_k requests a new route, the following algorithm is invoked.

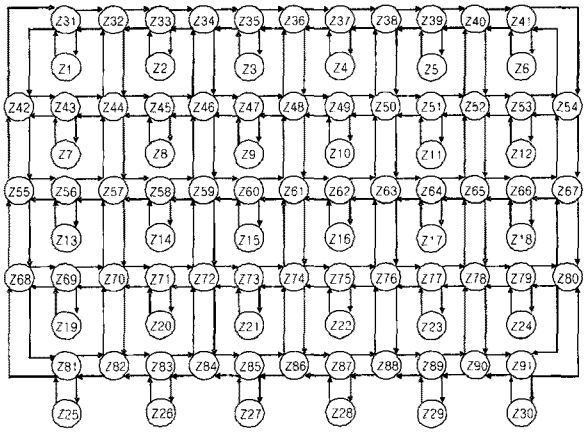
Algorithm : cycle elimination
Input : $G(V, E)$
$R_i, i = 1, \cdots, N_A$
Destination zone z_t for route-requesting vehicle A_k.
Process :
Step 1 : Construct a graph $G'(V, E')$ in which edges represent the routes of all vehicles. That is, for each route $R_i = \{z_{i0}, z_{i2}, \cdots, z_{in_i}\}$, $i = 1, \cdots, N_A$, edges $e = (z_{i,j}, z_{i,j+1})$, $j = 0, \cdots, n_i - 1$ are created.
Step 2 : Construct a weighted graph $G''(V, E'')$ by creating edges with the following manner :
For every edge $e \in E$, check if cycles containing e exist in $G'(V, E' + e)$. If no such cycles are found, add e in $G''(V, E'')$.
Step 3 : Find the shortest path from z_{k0} to z_t in $G''(V, E'')$.
Step 4 : If the path is found, assign the route to A_k.

Otherwise, wait until the next event occurs.

The graph $G'(V, E')$ constructed in step 1 represents current routes of all vehicles. This graph is used to check if cycles containing e are found by adding an edge $e \in E$. The cycles in $G'(V, E'+e)$ implies that there is possibility of cyclic deadlock if vehicle A_k follows the guided path represented by the edge e. By adding the edge e in $G''(V, E'')$ only when no such cycle exists in $G'(V, E'+e)$, the constructed graph provides a safe route for an idle vehicle.

Consider a graph $G(V, E)$ as shown in <Figure 2>. The graph represents all 91 zones and AGV path network of <Figure 1>. <Figure 3> shows a graph $G'(V, E')$ that is representing current routes of 6 busy vehicles described in <Table 1>. <Figure 4> shows the resulting graph $G''(V, E'')$ created in step 2 of the algorithm. For the construction of the graph, existence of cycles containing e is checked from the graph $G'(V, E'+e)$ where e is every edge in the graph of <Figure 2>. The edge is added to $G''(V, E'')$ only if no such cycle exists. It is easy to see that a cycle exists in $G'(V, E'+e)$ when the edge e is reverse directed for every edge in E'. For example, one of reverse directed edges of E' is (Z4, Z37). In the graph $G'(V, E'+(Z4, Z37))$, a cycle $Z4 \rightarrow Z37 \rightarrow Z4$ is found. Since the cycle contains the considered edge (Z4, Z37), the edge is not added to G''. Likewise, all reversed directed edges of E' are not added. Also, edge (Z60, Z59) is not added to $G''(V, E'')$ since a cycle $Z46 \rightarrow Z47 \rightarrow Z48 \rightarrow Z61 \rightarrow Z60 \rightarrow Z59 \rightarrow Z46$ is found in the graph of $G'(V, E'+(Z60, Z59))$.

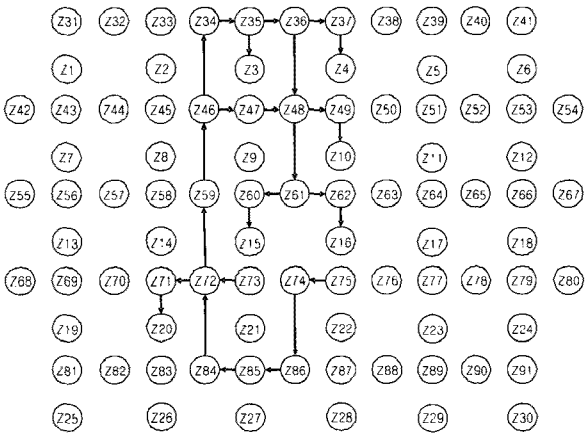
In step 3 of the algorithm, a shortest path from z_{k0} to z_l in graph G'' is determined for a new route of A_k. The route will be safe if cyclic deadlocks are not anticipated. The cyclic deadlock will occur only when there is a cycle in the graph representing current routes of busy vehicles and new routes, and every edge in the cycle is induced by different vehicle. For example, assume that the graph includes a cycle $Z46 \rightarrow Z47 \rightarrow Z48 \rightarrow Z61 \rightarrow Z60 \rightarrow Z59 \rightarrow Z46$ where every edge is induced by routing of different vehicle except edges (Z46, Z47) and (Z47, Z48). Only one vehicle induces these two edges. Thus, 6 edges in the cycle are induced by routings of 5 vehicles. This state does not lead to the cyclic deadlock. The cyclic deadlock occurs only when 6 different vehicles are positioning at each zone in the cycle, and are waiting to move to the next zone represented by the edge. Therefore, a necessary condition for future cyclic deadlock is the existence of a cycle in which every edge can be induced by different vehicle.



<Figure 2> A graph G representing path network

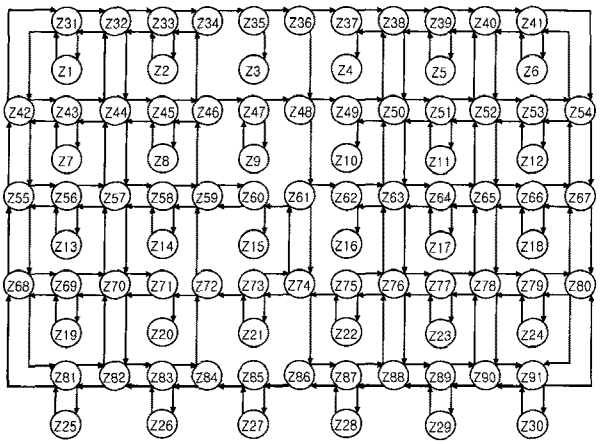
<Table 1> Current routes for cycle elimination algorithm

Vehicle	Current route
A1	Z36, Z48, Z61, Z60, Z15
A2	Z59, Z46, Z47, Z48, Z61, Z62, Z16
A3	Z85, Z84, Z72, Z59, Z46, Z34, Z35, Z36, Z37, Z4
A4	Z73, Z72, Z59, Z46, Z34, Z35, Z3
A5	Z47, Z48, Z49, Z10
A6	Z75, Z74, Z86, Z85, Z84, Z72, Z71, Z20



<Figure 3> A Graph G' representing current routes

Showing that the graph representing current and new routes has no such cycle will be enough to prove the safeness of new route determined by the algorithm. Assume that a graph $G'(V, E')$ representing only the current routes of busy vehicles does not contain such a cycle, i.e., there is no possibility of cyclic deadlock initially. In order to include the cycle



< Figure 4> A Graph G'' for safe route

of the necessary condition for the graph constructed by adding a new route to $G'(V, E')$, only one edge in the cycle should be induced by new route. However, it is impossible to find such a cycle when new route is determined by the cycle elimination algorithm. For example, assume that a graph $G'(V, E')$ includes a path $Z47 \rightarrow Z48 \rightarrow Z61 \rightarrow Z60 \rightarrow Z59 \rightarrow Z46$, and new route contains an edge (Z46, Z47). By adding edges in the new route to $G'(V, E')$, a cycle $Z46 \rightarrow Z47 \rightarrow Z48 \rightarrow Z61 \rightarrow Z60 \rightarrow Z59 \rightarrow Z46$ is found. This cycle will correspond to the necessary condition for cyclic deadlock if each edge in the path $Z47 \rightarrow Z48 \rightarrow Z61 \rightarrow Z60 \rightarrow Z59 \rightarrow Z46$ is induced by different vehicle. However, as described in step 2 of the algorithm, the edge (Z46, Z47) will not be included in the new route since a cycle containing (Z46, Z47) is found in $G'(V, E'+(Z46, Z47))$.

4. Deadlock avoidance based on graph reduction algorithm

The deadlock avoidance method in this paper is composed of two algorithms; graph reduction and vehicle control. At first, the graph reduction algorithm predicts all possible future deadlocks from the information on current routes of busy vehicles. And, the conditions for these future deadlocks are stored. The vehicle control algorithm exploits the stored information on deadlock conditions, and checks if requested movement of a vehicle satisfies the conditions. If any one of the deadlock conditions is satisfied, the requested vehicle will not be allowed to move to the next zone until the deadlock possibility is resolved. This method basically restricts the number of certain vehicles in a set of zones, thus, another restricted deadlock caused by restricting vehicle movement can occur. Therefore, all impending deadlocks and restricted deadlocks should be precisely detected to avoid deadlocks.

To predict all possible deadlocks and to determine the restricted number of specific vehicles in a specific set of zones, a graph-based method is used. Specifically, an extended graph, defined below, is used to represent current and future states of a system.

Definition : extended graph at level l

An extended graph at level l, $G_l(V_l, E_l)$, is a directed graph. In addition, each vertex v has the attributes of a set of zones $Z(v)$ included in the vertex, vehicles $H(v)$ that can be positioned in the vertex, and capacity $C(v)$ which is the restricted number of $H(v)$. Also, each edge e has an attribute of a passing vehicle $h(e)$.

Basically, an extended graph represents vehicles' routes. Additional attributes, $Z(v)$, $H(v)$, $C(v)$, and $h(e)$ make it possible to identify each vertex and edge with a specific vehicle. Vehicles that can be positioned in a specific vertex and vehicles that can move along the edge are identified.

The proposed graph reduction algorithm aims at predicting all impending and restricted deadlocks. Below, several definitions relating to the graph reduction are described.

Definition : cycle reduction condition

A cycle $(v_1, e_1, v_2, e_2, \cdots, v_n, e_n, v_1)$ in an extended graph at level l can be reduced to a vertex when the following condition is satisfied :

$h(e_1) \neq h(e_2) \neq \cdots h(e_n)$.

Definition : cycle reduction rules

When a cycle $(v_1, e_1, v_2, e_2, \cdots, v_n, e_n, v_1)$ in an extended graph at level l satisfies the cycle reduction conditions, it is reduced to a vertex v under the following rules :

1) $Z(v) = Z(v_1) \cup Z(v_2) \cup \cdots Z(v_n)$,

2) $H(v) \begin{cases} h(e_1) \cup h(e_2) \cup \cdots h(e_n), & \text{if } l = 0 \\ H(v_1) \cup H(v_2) \cup \cdots H(v_n), & \text{otherwise} \end{cases}$

3) $C(v) = |H(v)| - 1$.

Definition : reduced graph creation rules

An extended graph $G_{l+1}(V_{l+1}, E_{l+1})$ is created from $G_l(V_l, E_l)$ under the following rules :

1) Create vertices V_{l+1} by applying the cycle reduction rules to all cycles in $G_l(V_l, E_l)$.

2) When there is an edge $e=(v_a, v_b)$ in G in which $Z(v_a) \in Z(v_{l+1,i}) \cap Z(v_{l+1,j})$, $Z(v_b) \in Z(v_{l+1,i}) - Z(v_{l+1,i})$, and $h(v_a, v_b) \in H(v_{l+1,i}) \cap H(v_{l+1,j})$, connect the edges from $v_{l+1,i}$ to $v_{l+1,j}$ with $h(v_{l+1,i}, v_{l+1,j}) = h(v_a \cdot v_b)$.

Like the cycle elimination algorithm in the previous section, the graph reduction algorithm is invoked whenever an idle vehicle A_k requests a new route.

Algorithm : graph reduction

Input : $G(V, E)$

$R_i, i=1, \cdots, N_A$

Destination zone z_t for route-requesting vehicle A_k.

Process :

Step 1 : Find the shortest path from z_{k0} to z_t in $G(V, E)$, then assign the path to requesting vehicle A_k as a route R_k.

Step 2 : Set l to 0.

Construct $G_0(V_0, E_0)$ by creating edges $e=(z_{i,j}, z_{i,j+1})$, $j=0, \cdots, n_i-1$ for every route $R_i = z_{i0}, z_{i1}, \cdots, z_{in_i}$, $i=1, \cdots, N_A$. In addition, the following attributes are assigned :

1) Each vertex $z_{i,j}$ is given the attributes of $Z(z_{i,j}) - z_{i,j}$, $H(z_{i,j}) = \phi$, and $C(z_{i,j}) = 1$

2) Each edge $e=(z_{i,j}, z_{i,j+1})$ is given the attributes of $h(e) = A_i$

Step 3 : Construct a reduced graph $G_{l+1}(V_{l+1}, E_{l+1})$ from $G_l(V_l, E_l)$ by applying the reduced graph creation rule. Store $Z(v)$, $H(v)$, and $C(v)$ for all $v \in V_{l+1}$.

Step 4 : Add l by 1.

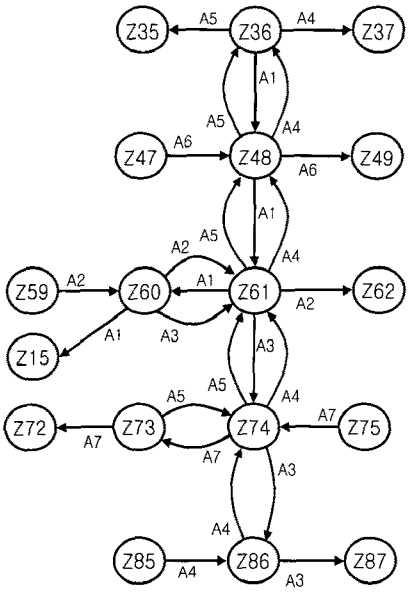
Go to step 3 if $G_l(V_l, E_l)$ is not null.

The graph $G(V, E)$ used in the algorithm represents an AGV path network as in the cycle elimination algorithm. From the graph, a shortest path is determined in step 1 and is assigned to the vehicle A_k as a new route. In step 2, an extended graph at level 0 is created. The graph represents current routes of busy vehicles and the just determined new route of idle vehicle A_k. In steps 3 and 4, a number of extended graphs are constructed sequentially by applying the reduced graph creation rules in order to obtain restrictions on vehicle movement.

An example will be provided to help explain the algorithm's intrinsic mechanism. <Table 2> shows the current routes and new route for 7 vehicles in the path network of <Figure 1>. <Figure 5> is an extended graph at level 0 which represents the routes in <Table 2>. In this figure, an attribute, $Z(v)$, is marked in the circle (vertex), and $h(e)$ is attached to each edge. Since $H(v)$ for every vertex in the graph at level 0 is null (see step 2.1), it is not included in the figure. From the graph, it is observed that cyclic deadlocks are anticipated. Each cycle satisfying cycle reduction conditions means there could be a deadlock if specific vehicles are positioned in the vertices (i.e., zones) included in the cycle. For example, one of 10 cycles is {Z36, Z48, Z36}, in which two edges have attributes of vehicles A1 and A4. If vehicles A1 and A4 are positioned at Z36 and Z48, respectively, at the same time, a deadlock occurs since no vehicle can move to the next zone. One deadlock avoidance method is restricting the number of vehicles so that at most only one vehicle of A1 and A4 should be positioned at zones Z36 and Z48 at any time. In this manner, the 10 cycles in the graph that satisfy the cycle reduction condition, and the related restrictions specified in the cycle reduction rule are determined, as shown in <Table 3>. From the results of $Z(v)$, $H(v)$, and $C(v)$, the vehicle's movement is so restricted that the maximum number of vehicles of $H(v)$ that are positioned at $Z(v)$ at any time is $C(v)$. It is not difficult to discover that additional deadlocks can occur by restricting the number of specific vehicles in a set of zones, as shown in <Table 3>. <Figure 6> shows the graph at level 1 that was derived by applying the reduced graph creation rule to the graph at level 0. In this figure, a box attached to each vertex represents the attribute $H(v)$. According to the cycle reduction rule, cycles in <Table 3> are represented as vertices.

<Table 2> Current routes for graph reduction algorithm

Vehicle	Current route
A1	Z36, Z48, Z61, Z60, Z15
A2	Z59, Z60, Z61, Z62
A3	Z60, Z61, Z74, Z86, Z87
A4	Z85, Z86, Z74, Z61, Z48, Z36, Z37
A5	Z73, Z74, Z61, Z48, Z36, Z35
A6	Z47, Z48, Z49
A7	Z75, Z73, Z72

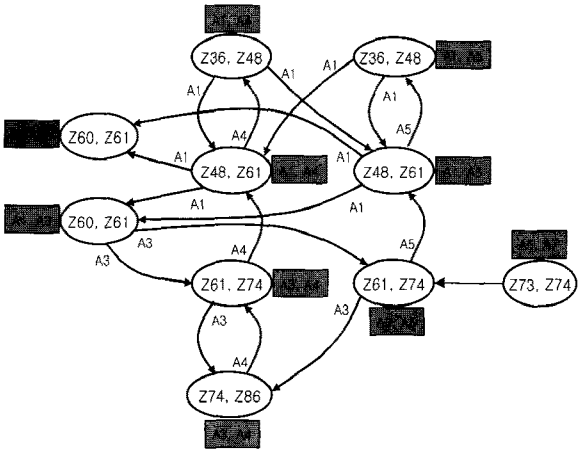


<Figure 5> Extended graph at level 0

<Table 3> Cycles in the graph at level 0

No	Cycle	$h(e)$	$Z(v)$	$H(v)$	$C(v)$
1	Z36, Z48, Z36	A1, A4	{Z36, Z48}	{A1, A4}	1
2	Z36, Z48, Z36	A1, A5	{Z36, Z48}	{A1, A5}	1
3	Z48, Z61, Z48	A1, A4	{Z48, Z61}	{A1, A4}	1
4	Z48, Z61, Z48	A1, A5	{Z48, Z61}	{A1, A5}	1
5	Z60, Z61, Z60	A2, A1	{Z60, Z61}	{A1, A2}	1
6	Z60, Z61, Z60	A3, A1	{Z60, Z61}	{A1, A3}	1
7	Z61, Z74, Z61	A3, A4	{Z61, Z74}	{A3, A4}	1
8	Z61, Z74, Z61	A3, A5	{Z61, Z74}	{A3, A5}	1
9	Z73, Z74, Z73	A5, A7	{Z73, Z74}	{A5, A7}	1
10	Z74, Z86, Z74	A3, A4	{Z74, Z86}	{A3, A4}	1

Notice that even though two vertices, v_1 and v_2, contain the same zones (i.e., $Z(v_1) = Z(v_2)$), these vertices are separately identified when the attributed vehicles are different (i.e., $H(v_1) \neq H(v_2)$). Also, as described in the reduced graph creation rule, an edge is created when there is a route from the zones in the source vertex to the zones in the destination vertex that is not included in the source vertex, by a vehicle appearing in both vertices. For example, consider two vertices, v_1 and v_2 in level 1 where $Z(v_1) = \{Z36, Z48\}$, $H(v_1) = \{A1, A4\}$, $Z(v_2) = \{Z36, Z61\}$, and $H(v_2) = \{A1, A4\}$. If there is a route in A1 from the zones in {Z36, Z48} to the zones in Z61 (= {Z48, Z61} - {Z36, Z48}), an edge with A1 is connected from {Z36, Z48} to {Z48, Z61}. Since A1 moves from Z48 to Z61 and A2 moves from Z48 to Z36, two edges are created between both vertices, as shown in <Figure 6>.



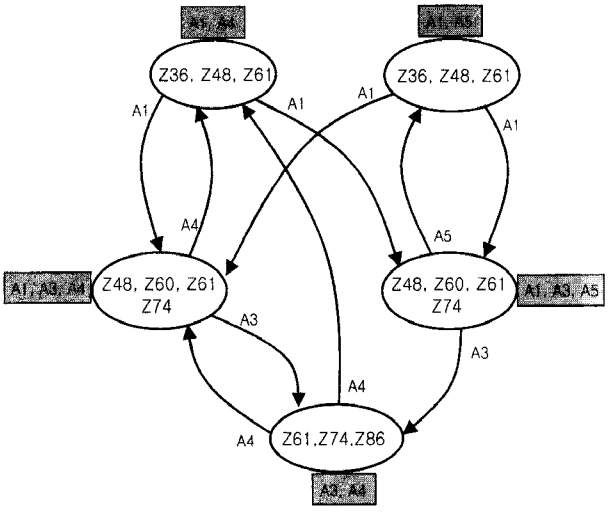
<Figure 6> Extended graph at level 1

<Table 4> Cycles in graph at level 1, 2 and 3

Level	Cycle	$h(e)$	$Z(v)$	$H(v)$
1	{Z36, Z48}{Z48, Z61}{Z36, Z48}	A1, A4	{Z36, Z48, Z61}	{A1, A4}
1	{Z36, Z48}{Z48, Z61}{Z36, Z48}	A1, A5	{Z36, Z48, Z61}	{A1, A5}
1	{Z48, Z61}{Z60, Z61}{Z61, Z74}{Z48, Z61}	A1, A3, A4	{Z48, Z60, Z61, Z74}	{A1, A3, A4}
1	{486, Z61}{Z60, Z61}{Z61, Z74}{Z48, Z61}	A1, A3, A5	{Z48, Z60, Z61, Z74}	{A1, A3, A5}
1	{Z61, Z74}{Z74, Z86}{Z61, Z74}	A3, A4	{Z61, Z74, Z86}	{A3, A4}
2	{Z36, Z48, Z61}{Z48, Z60, Z61, Z74}{Z36, Z48, Z61}	A1, A4	{Z36, Z48, Z60, Z61, Z74}	{A1, A3, A4}
2	{Z36, Z48, Z61}{Z48, Z60, Z61, Z74}{Z36, Z48, Z61}	A1, A5	{Z36, Z48, Z60, Z61, Z74}	{A1, A3, A5}
2	{Z48, Z60, Z61, Z74}{Z61, Z74, Z86}{Z48, Z60, Z61, Z74}	A3, A4	{Z48, Z60, Z61, Z74, Z86}	{A1, A3, A4}
2	{Z36, Z48, Z61}{Z48, Z60, Z61, Z74}{Z61, Z74, Z86}{Z36, Z48, Z61}	A1, A3, A4	{Z36, Z48, Z60, Z61, Z74, Z86}	{A1, A3, A4}
2	{Z36, Z48, Z61}{Z48, Z60, Z61, Z74}{Z61, Z74, Z86}{Z36, Z48, Z61}	A1, A3, A4	{Z36, Z48, Z60, Z61, Z74, Z86}	{A1, A3, A4, A5}
3	{Z36, Z48, Z60, Z61, Z74}{Z48, Z60, Z61, Z74, Z86}{Z36, Z48, Z60, Z61}	A3, A4	{Z36, Z48, Z60, Z61, Z74, Z86}	{A1, A3, A4}

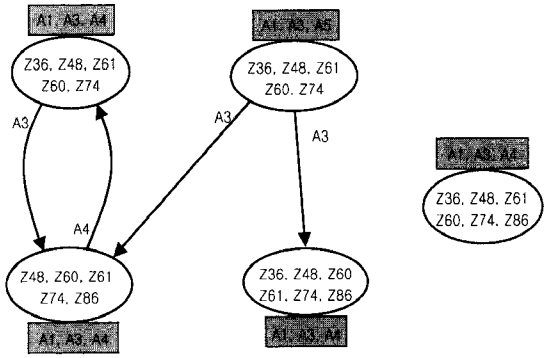
From the graph at level 1, several cycles are also found. Consider a cycle containing the two abovementioned vertices. The cycle means that a deadlock occurs if A1 is positioned at one of the zones {Z36, Z48} and A4 is positioned at one of the zones {Z48, Z61} under the restriction specified in <Table 5>. To avoid this restricted deadlock, at most one vehicle of {A1, A4} should be allowed to be positioned in zones {Z36, Z48, Z61} at any time. Likewise, 5 cycles satisfying the cycle reduction condition are determined, as depicted in <Table 4>.

<Figure 7> and <Figure 8> show extended graphs at subsequent levels derived from the graph at the previous level by applying the reduced graph creation rule. This cycle reduction procedure is repeated until there are no more cycles left in the graph. All cycles and related restrictions determined from the extended graph at levels 1~3 are included in <Table 4>. These results in <Table 3> and <Table 4> are used to effectively avoid deadlocks. In the appendix, it is shown that by restricting the capacity of zones $Z(v)$ such as $C(v) = |H(v)| - 1$, the restriction deadlock is avoided. Since the graph $G(V, E)$ used in step 1 of the algorithm represents all zones and guided paths in an AGV network, it is not changed over time. Therefore, the shortest paths between any two P/D stations can be determined prior to applying the algorithm. With this consideration, the burden of the shortest path algorithm in step 1 is removed. Then, the complexity is dominated by the repetition of steps 2, 3 and 4. Also, obtaining all cycles in the reduced graph creation rule will contribute mainly to the complexity of steps 2, 3 and 4.



<Figure 7> Extended graph at level 2

At last, the vehicle control algorithm will be explained. As stated before, a graph reduction algorithm is invoked whenever a vehicle requests a new route. From the algorithm, restrictions with $Z(v)$, $H(v)$, $C(v)$ are determined and are stored in order to be used for the vehicle control algorithm. Whenever a vehicle requests a move to the next zone from the current zone, the following algorithm is invoked.



<Figure 8> Extended graph at levels 3 and 4

Algorithm : vehicle control

Input : restrictions of $Z(v)$, $H(v)$, $C(v)$ determined from graph reduction algorithm

$R_i = \{z_{i0} \rightarrow z_{i1} \rightarrow \cdots \rightarrow z_{in_i}\}$, $i = 1, \cdots, N_A$

Vehicle A_k requesting a move to the next zone

Process :

Step 1 : For every v,

1) compute $n(v)$ that is the number of vehicles in $H(v)$ currently positioning at $Z(v)$ assuming that vehicle A_k has moved to the next zone

2) if $n(v)$ is greater than $C(v)$, do not allow the movement of A_k to the next zone and return.

Step 2 : Allow vehicle A_k to move to the next zone.

This algorithm checks if the movement of the requested vehicle violates the restrictions determined by the graph reduction algorithm. If all restrictions specified with $Z(v)$, $H(v)$, $C(v)$ are not violated, the movement of vehicle A_k to the next zone is allowed. Otherwise, the movement is prohibited to avoid deadlocks, and the vehicle must wait at the current zone until the next event occurs.

5. Simulation experiments

To evaluate the performance of the developed algorithms,

a simulation study was accomplished with the generic AGV path network in <Figure 1>. Simulation models were created using the ProModel™ software package, and executed in Pentium-4 3.2 GHz PC. Although the package provides a built-in AGV model, it is not be suitable to the purpose of this paper's simulation. In this simulation model, AGV control modules, including the developed algorithms, were created by regarding vehicles as entities, so that the movement of the entities is completely controlled.

Since the objective of the simulation study conducted was to evaluate the performance of the deadlock-free algorithms, the system was assumed to be busy. That is, it was assumed that there was always work for an idle vehicle to do.

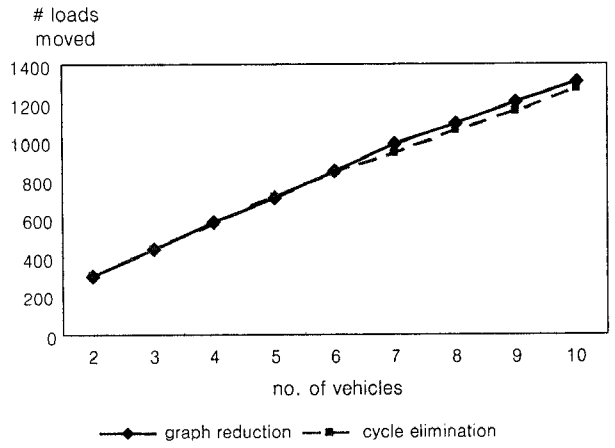
Both cycle elimination and graph reduction algorithm are invoked whenever an idle vehicle requests a new route. In applying these algorithms, it is assumed that the current system state is not a deadlock state. If the current state is a deadlock, either the shortest path for an idle vehicle A_k may not be determined for ever under the cycle elimination algorithm or no vehicle can move to the next zone under graph reduction and vehicle control algorithms. In this simulation study, the following provisions were made to prevent this state from occurring :

1) Initially, each idle vehicle was positioned in front of a randomly selected P/D station.
2) In assigning work to an idle vehicle, the destination is selected randomly among P/D stations, except for the following stations :
   - P/D stations that other vehicles currently serve, and
   - P/D stations in which any vehicle currently occupies the corresponding zone.

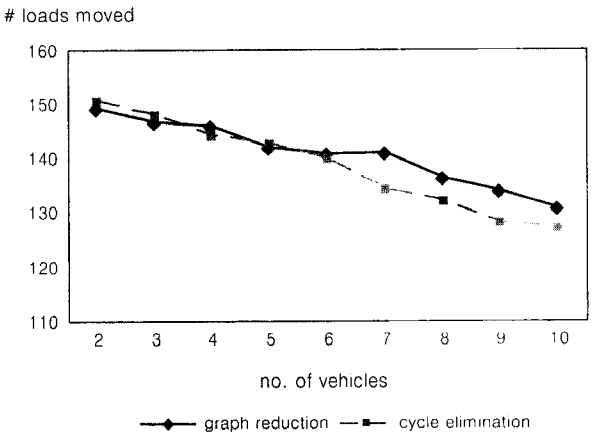
These provisions allow all vehicles to be positioned in front of the P/D stations without any conflict. In other words, the graph representing the current state does not contain a cycle with two zones, namely, the zone of the P/D station and the zone adjacent to the P/D station. Therefore, the vehicle at the P/D station can wait there safely if its movement to the next zone violates restrictions. The violation will always be resolved after other vehicles leave the considered set of zones.

During the execution of the simulation for 7 days, no deadlock incident occurred. Both the algorithms developed in this paper effectively provided deadlock prevention and deadlock avoidance. <Figure 9> shows the number of loads moved in a day as the number of vehicles in the system was changed from 2 to 10. The figure shows a natural increase in the number of loads moved as the number of vehicles increased. The deadlock avoidance based on the graph reduction algorithm yielded slightly better results when the number of vehicles exceeded 6. Even though the number of load movements increased when the number of available vehicles was increased, the load movement performed by each vehicle decreased due to congestion, as shown in <Figure 10>. The figure shows the average number of loads moved by a vehicle in a day. When there were less than 7 vehicles, the numbers of load movements with the cycle elimination algorithm and the graph reduction algorithm were almost the same. When the number of vehicles exceeded 6, however, the graph reduction algorithm produced more load movements than did the cycle elimination algorithm. This result is based on the fact depicted in <Figure 11>. When the number of vehicles exceeded 6, the durations of the load movement of the two algorithms differed. While more blocking and waiting times were incurred with the graph reduction algorithm as the number of vehicles increased, more moving time was required with the cycle elimination algorithm. Considering the results, whereby the total working time of the cycle elimination algorithm was greater than that of the graph reduction algorithm, the route created under the cycle elimination algorithm is considered to have deviated from the shortest path.

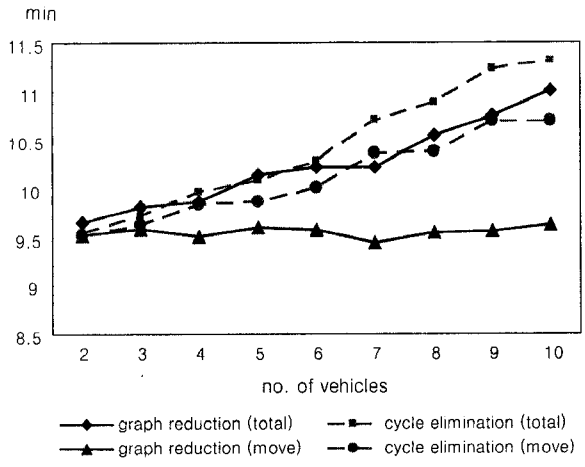
From the CPU time taken for the algorithm execution in <Figure 12>, it was observed that the graph reduction algorithm is much faster than the cycle elimination algorithm. When there were 10 vehicles in the system, only 0.023 second was required under the graph reduction algorithm. Notice that the CPU time includes the execution times of the simulation engine. With this consideration, the average 0.085 second under the cycle elimination algorithm will not be regarded as significant. As stated in Section 3, the time complexity of the cycle elimination algorithm depends on the number of vertices and edges in the graph representing a static AGV path network. This is why the CPU time of the algorithm shows an insensitive characteristic with an increase in the number of vehicles. However, the CPU time of the graph reduction algorithm increases according to the number of vehicles. As the number of vehicles increases, the first-level graph manipulated in the algorithm becomes bigger, with many vertices and edges. This implies that more CPU time is required with an increased number of vehicles.



<Figure 9> Number of loads moved in a day



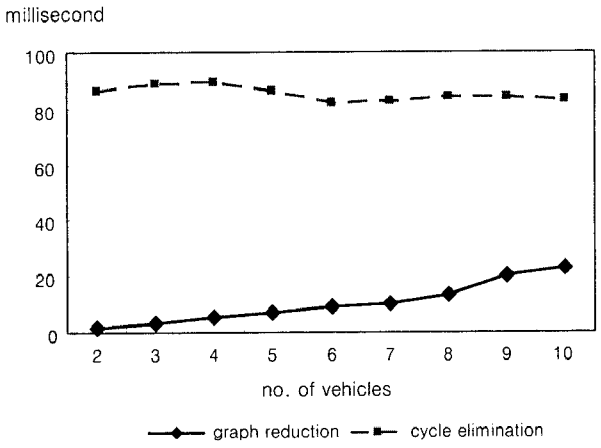
<Figure 10> Number of loads moved per vehicle in a day



<Figure 11> Working tomes for load movement

6. Conclusion

In this study, two deadlock-free algorithms in a zone-controlled AGV system were developed. These algorithms were based on graph models, and were designed to be effectively used in bi-directional AGV path networks. Specifically, the cycle elimination algorithm was implemented by adapting a deadlock prevention strategy. It determined safe routes for an idle vehicle, so that there was no cycle in the graph model. The second algorithm was based on a deadlock avoidance strategy, so that future deadlocks were predicted in real time with a dynamically constructed graph model. From information on these predicted deadlocks, vehicle movement was restricted to avoid future deadlocks.



<Figure 12> Computation times

Throughout the simulation study conducted, the graph reduction algorithm performed better than the cycle elimination algorithm. Due to the graph reduction algorithm's shorter moving time, more loads were transferred under it, especially when the number of vehicles was large. The CPU time also showed that the graph reduction algorithm is superior to the cycle elimination algorithm. While the CPU time of the cycle elimination algorithm was insensitive to the number of vehicles in the system, that of the graph reduction algorithm increased as the number of vehicles increased. It was also seen that the CPU times of both algorithms increased when the AGV path network became bigger, with many zones and guided paths. Considering that more than 100 vehicles with complex path networks are currently in operation in some container terminals, more research on time complexity will be required for these complex systems.

Reference

[1] Banaszak, Z. A. and Krogh, B. H.; "Deadlock avoidance in flexible manufacturing systems with con-

currently competing process flows," *IEEE Transactions on Robotics and Automation*, 6 : 724-734, 1990.

[2] Fanti, M. P.; "Event-based controller to avoid deadlock and collisions in zone-control AGVS," *International Journal of Production Research*, 40(6) : 1453 -1478, 2002.

[3] Kim, C. W. and Tanchoco, J. M. A.; "Conflict-free shortest-time bi-directional AGV routeing," *International Journal of Production Research*, 29(12) : 2377 -2391, 1999.

[4] Lee, C. C. and Lin, J. T.; "Deadlock prediction and avoidance based on Petri nets for zone-control automated guided vehicle systems," *International Journal of Production Research*, 33(12) : 3249-3265, 1995.

[5] Leung, Y. T. and Sheen, G. J.; "Resolving deadlocks in flexible manufacturing cells," *Journal of Manufacturing Systems*, 12 : 291-304, 1993.

[6] Oboth, C., Batta, R., and Karwan, M. H.; "Dynamic conflict-free routeing of automated guided vehicles," *International Journal of Production Research*, 37(9) : 2003-2030, 1999.

[7] Rajeeva, L. M., Wee, H. Ng, W., and Teo C.; "Cyclic deadlock prediction and avoidance for zone -controlled AGV system," *International Journal of Production Economics*, 83 : 309-324, 2003.

[8] Reveliotis, S. A.; "Conflict resolution in AGV systems," *IIE Transaction*, 32(7) : 647-659, 2000.

[9] Wysk, R. A., Yang, N. S., and Joshi, S.; "Detection of deadlocks in flexible manufacturing cells," *IEEE Transactions on Robotics and Automation*, 7 (6) : 853-859, 1991.

[10] Yeh, M. S. and Yeh, W. C.; "Deadlock prediction and avoidance for zone-control AGVs," *International Journal of Production Research*, 36(10) : 2879-2889, 1998.

[11] Yim, D. S., Kim, J. I., and Woo, H. S.; "Avoidance of deadlocks in flexible manufacturing systems using capacity-designated directed graph," *International Journal of Production Research*, 35(9) : 2459-2475, 1997.