Journal of the Korean Institute of Industrial Engineers
Vol. 26, No. 4, pp. 392-401, December 2000.

존 조정하에서의 AGV 고착 방지

임 동 순

한남대학교 산업공학과

AGV Deadlock Avoidance Under Zone Control

Dong-Soon Yim

In this work, a deadlock avoidance strategy is proposed in order to effectively handle conflicts and deadlocks occurring in zone-control AGV (Automated Guided Vehicle) systems. The basic idea is based on Capacity-designated Directed Graph (CDG) theory that was developed to avoid from deadlocks in manufacturing systems. However, to enforce the effectiveness of detecting impending and restricted deadlocks, AGV routings are explicitly described in Extended Directed Graph (EDG). From EDG, a non-conservative deadlock-avoidance strategy is derived. The superiority of the proposed strategy lies on the applicability to diverse AGV path configurations using zone control. Also, because of its insensibility and robustness, it can be effectively used when the system has randomness and stochastic nature.

1. 서 론

존 조정(zone contrl)은 자동반송차량(Automated Guided Vehicle: AGV) 간의 충돌과 상충을 방지하기 위해 AGV 패스 네트워크(path network)를 서로 중복되지 않는 지역으로 분할하여 한 존에 최대 하나의 차량 만을 허용하는 조정 정책이다. 그러나 이러한 조정이 차량 간의 일차적인 상충을 방지할 수 있을지라도 존 조정이 갖고 있는 근본적인 제약으로 인하여 고착(deadlock)이 발생될 수 있다. 존 조정 하에서 고착을 방지하는 방법을 도출하기 위하여 CDG (Capacity-designated Directed Graph) 이론(Yim *et al.*, 1997)을 적용할 수 있다. CDG 이론은 생산시스템의 작업물 이동에서 발생하는 고착으로부터의 회피를 목적으로 개발되었으나, 다양한 생산시스템에 적용될 수 있는 응용력을 갖고 있어 AGV 조정 시스템에서도 효과적으로 이용될 수 있다. CDG를 이용하는 정책은 고착 회피 방안의 결정 시점에 따라 정적인 것과 동적인 것이 있다. 정적인 정책은 운영전의 계획 단계에서 모든 가능한 차량 라우트(route: 차량에게 할당될 존의 순서)를 도출하여 이를 바탕으로 정적인 고착 방지 방안을 강구하는 것이다. 반면, 동적인 방법은 차량에게 명령을 하달할 때마다 운행 또는 대기 중인 차량들의 현재 주어진 라우트를 바탕으로 고착 방지 방안을 강구한다. CDG에 의한 방법은 각 차량의 라우트를 자세하게 표현하지 않는 단순함으로 인하여 보수적인 방안에 속한다. 즉, 고착이 발생하지 않는 경우에도 발생 가능성이 있다고 판단할 수 있다. 정적인 방법은 동적인 방법에 비해 보다 보수적이다.

본 연구에서는 존 조정 정책을 사용하는 다양한 형태의 AGV 패스 네트워크에서 이용될 수 있는 동적인 고착 방지 방안을 개발하는데 있다. 특히, 양 방향 또는 한 방향 패스 네트워크와 둘 이상의 어떤 차량의 수에도 적용될 수 있는 일반적인 방안을 제안한다. 기본적인 방법은 기존의 CDG 이론을 이용하나 보수적인 정책이 아닌, 고착이 발생되는 조건을 만족시키지 않도록 하는 보다 효과적인 정책을 유도한다. 이를 위하여 차량에게 할당된 라우트를 자세하게 묘사할 수 있도록 CDG를 변형한 확장된 방향성 그래프(Extended Directed Graph: EDG)를 이용한다.

2. 관련 연구

AGV 시스템의 상충(conflict)은 한 차선에서 두 차량이 동시에 운행하는 경우에 발생한다. 한 차선에 두 차량이 같은 방향으로 운행하나, 뒤 차량의 속도가 앞의 것 보다 빨라 충돌하는 경우가 있다. 또는, 한 차선에 두 차량이 서로 다른 방향으로 차선의 가운데 지점을 향해 운행한다면 충돌이 발생한다. 전자를 catch-up 상충, 후자를 head-on 상충이라고 한다(Broadbent *et al.*,

1985). 물론, head-on 상충은 양 방향 패스 네트워크에서만 발생된다. 이러한 상충을 방지하기 위하여, 일반적으로 존 조정이 이용된다. 존 조정은 AGV 패스 네트워크를 서로 중복되지 않는 존으로 나누어 각 존에 최대 하나의 차량만이 있도록 제한한다. 존 조정을 사용하지 않는 경우에는 차량의 라우트를 결정할 때 상충이 발생되지 않는 안전한 라우트를 선택하도록 하는 방법이 이용된다.

안전한 라우트를 결정하는 방법은 각 차량의 운행에 대한 확정적인 시간을 바탕으로 세부적인 스케줄을 결정하는 방법과 운행시간 등 시스템의 불확실성을 고려하는 방법이 있다. Oboth *et al.*(1999)은 Krishnamurphy *et al.* (1993)의 방법에 기초하여 양 방향 패스 네트워크에서 교차로를 vertex로, 교차로 간의 차선을 edge로 표현한 그래프 모델을 이용하여 상충이 없는 라우트를 결정하는 방법을 사용하였다. 그들의 방법은 확정적인 운행시간을 이용하여 고려되는 차량들이 주어진 운행을 모두 끝내는 시간을 최소화하는 라우트를 선정하였다. Kim and Tanchoco (1991) 역시 양 방향 패스 네트워크에서 확정적 운행시간을 이용하여 상충이 없는 최단 거리를 차량의 라우트로 결정하는 방법을 제시하였다. 이렇게 라우트를 계획하는 방법은 확정적인 운행시간과 개회로 조정(open-loop control)에 기초하므로, 시스템 불확실성에 대해 안정성을 보장하지 못하는 단점이 있다. Reveliotis(2000)는 시스템 불확실성을 고려한 라우트 계획 방법을 제안하였다. 존 조정을 사용하는 양 방향 패스 네트워크에서 그래프 모델을 이용하여 상충과 고착이 없는 안전한 라우트를 구하는 방법을 제시하였다. 특히, Banker's 알고리듬을 AGV 시스템에 알맞게 변형하여 라우트의 안전성을 조사하는 방법으로 사용하였다. 그러나 각 차량들의 운행시간을 고려하지 않으므로, 상충과 고착이 발생하지 않는 경우에도 발생 가능성이 있다고 판단할 수 있는 보수적인 방법에 속한다.

상충을 방지하기 위하여 존 조정을 사용하는 AGV 시스템에서는 존 조정 자체에 의한 고착이 발생할 수 있다. 고착은 AGV 시스템에서 환형 대기상태(circular wait state)를 의미하여 다음 존으로의 진행을 요구하는 차량들과 차량들이 공유하는 자원인 존 사이에 환형 관계를 이루는 상태이다. 고착을 방지하는 기본적인 방법은 이를 미리 예측하여 고착 가능성이 있을 경우, 해당 존에 있는 차량들에게 고착 가능성이 없어질 때 까지 그 존에서 대기토록 명령을 내리는 것이다. Lee and Lin(1995)은 한 방향 패스 네트워크에서 페트리 네트(Petri net)모델을 사용하여 고착회피를 위한 방법을 제시하였다. 차량이 한 존에서 다음 존으로 진행할 때 페트리 네트의 도달성(reachability)을 조사하여 고착 가능성을 결정하였다. Yeh *et al.* (1998)은 역시 한 방향 패스 네트워크에서 그래프 모델을 사용하여 고착 가능성을 조사하는 방법을 제시하였다. 존 조정하에서의 이러한 방법들은 한 방향 패스 네트워크에서의 일차적인 고착, 즉 존에 있는 차량들이 사이클을 이루는 기본적인 고착만을 고려하여 현재의 상태가 고착이 아니지만 이 상태가 미래의 고착을 유발하는 것을 방지하는 정책에 속한다. Wysk *et al.* (1991)은 이러한 미래의 고착을 임박고착(impending deadlock)으로 정의하였다. 한 방향 패스 네트워크일 경우에는 catch-up 상충만이 존재하여, 존 조정을 이용할 경우 고착은 환형 대기상태를 의미함은 자명하다. 그러나 환형 대기상태에 도달하지 않도록 존에 있는 차량들의 진행을 제한할 경우, 이 제한에 의한 부수적인 고착이 발생할 수 있다. Banaszak and Krogh(1990)의 연구에서는 이러한 부수적인 고착을 제약고착(restricted deadlock)으로 정의하였다. 물론, Lee *et al.* (1995) and Yeh *et al.*(1998)이 예제로 사용한 AGV 시스템에서의 존 구성은 이러한 제약고착이 발생하지 않는 특정한 경우에 해당한다.

본 연구에서 제시하는 고착 회피 방안은 존 조정을 사용하는 AGV 시스템에 한한다. 그러나 한 방향뿐만이 아니라 양 방향 패스 네트워크, 그리고 어떠한 존의 구성에서도 이용될 수 있는 일반적인 방법을 제시한다. 뿐만 아니라, 시스템의 불확실성에 대처할 수 있는 안정성을 갖고, 고착의 발생을 정확히 예측하여 이를 방지하는 방안을 제시한다.

3. EDG를 이용한 AGV 고착 방지

CDG 이론을 이용하여 고착의 발생을 방지하기 위하여는 동적인 CDG 생성 방법이 효과적이긴 하나, 보수적인 정책을 탈피하여 보다 정확히 고착 발생을 예측하고, 이를 미연에 방지하는 방안을 도출하기 위하여는 CDG 모델에 각 차량의 순차적인 라우트를 자세히 묘사하여야 한다. 자세한 모델의 작성을 위해 본 연구에서는 다음과 같은 AGV 시스템을 대상으로 한다.

1. AGV 네트워크는 한 방향 그리고/또는 양 방향이다.
2. AGV 간의 상충을 방지하기 위해 존 조정을 사용한다.
3. 쉬고 있는 차량들에게 이동명령을 하달할 때 차량의 라우트는 결정된다.
4. 차량의 라우트는 고정된 순서를 갖는다.

동적인 CDG 모델의 생성방법은 AGV 네트워크에서의 각 존을 vertex로 나타내고, 현재 명령을 수행 중인 차량들이 앞으로 이동하여야 할 라우트를 방향이 있는 edge로 나타낸다. 주어진 라우트는 명령 실행 도중에 변경될 수 없는 성질의 것으로 가정하고, 고정된 순서를 갖고 있어 유연적이거나, 여러 대안을 갖는 가능성을 배제한다. 때문에, 차량에게 주어진 라우트는 일련의 순차적인 존으로 구성되어 한 존에서 다음 존으로 이동하는 순번을 라우트 순번이라고 정의한다. 차량의 라우트를 자세히 표현하기 위하여 edge에 어느 차량의 몇 번째 라우트 순번에 의한 것인가를 나타내는 특성치를 부가한다. 이는 다음의 확장된 방향성 그래프로 정의된다.

정의: 확장된 방향성 그래프(EDG)

EDG는 그래프로 G=(V, E)로 V는 vertex의 집합이고, E는 두

vertex를 잇는 edge들의 family이다. E의 성분 $e=(v_i, v_j)$는 vertex v_i에서 v_j로 잇는 방향을 가지는 edge로서 차량 번호와 라우트 순번을 의미하는 $n(e)$와 $s(e)$를 특성치로 갖는다.

Edge는 집합이 아니라 family로 정의되므로 두 vertex를 잇는 다수의 edge를 허용한다. 그러나 한 차량이 수행하고 있는 현 라우트는 고정된 순서를 가지므로 EDG에서 하나의 패스로 표현되어 조합의 특정 값을 갖는 edge e는 유일하다.

존 조정하에서 차량의 라우트를 EDG로 모델링하는 경우 각 존에는 단지 하나의 차량만 허용하므로, CDG에서 고려한 vertex의 용량은 불필요하다. 모든 vertex는 각 존을 나타내어 1의 용량을 갖기 때문이다.

전체 AGV 네트워크 중 일부분을 나타낸 <그림 1>에서 4대의 차량이 이동하는 현재 상태를 EDG로 모델링 해보자. 굵은 선으로 표시된 AGV 패스는 양 방향을 허용하여 어느 방향으로도 진행이 가능하다. 안전을 고려하여 교차로를 중심으로 한 지역에 하나의 차량만 있도록 하였고, 기타 운행에 필요한 존을 Z_1부터 Z_9까지 설정하였다. A_1, A_2, A_3, A_4의 4차량이 현재 명령을 수행 중에 있으며, 앞으로의 라우트는 <표 1>과 같다.

이러한 상황을 EDG로 표현한 것은 <그림 2>와 같다. 각 존은 vertex로 표현되었고, 각 edge는 <표1>의 라우트를 나타내어 특성치($n(e)$, $s(e)$)가 포함되었다. 그림의 EDG에서 고착이 발생되는 경우는 현재 차량을 포함하고 있는 vertex들에 대하여, 차량들의 다음 라우트 순번에 해당하는 edge들에 의해 사

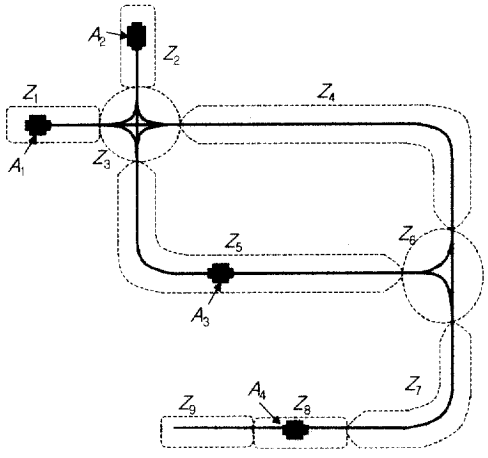

그림 1. AGV 패스 네트워크.

표 1. AGV 라우트

차량 번호	라우트
A_1	$Z_1 \rightarrow Z_3 \rightarrow Z_4 \rightarrow Z_6 \rightarrow Z_7 \rightarrow Z_8 \rightarrow Z_9$
A_2	$Z_2 \rightarrow Z_3 \rightarrow Z_4 \rightarrow Z_6 \rightarrow Z_7 \rightarrow Z_8 \rightarrow Z_9$
A_3	$Z_5 \rightarrow Z_3 \rightarrow Z_2$
A_4	$Z_8 \rightarrow Z_7 \rightarrow Z_6 \rightarrow Z_5 \rightarrow Z_3 \rightarrow Z_1$

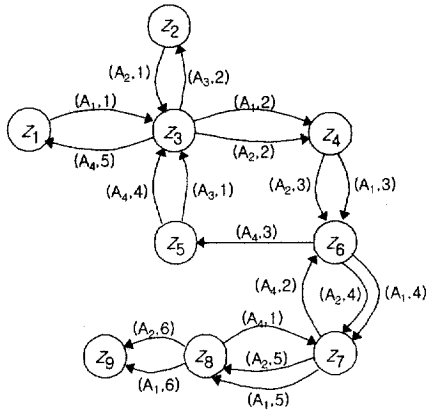

그림 2. EDG.

이클(cycle)을 이루는 환형 대기상태에 있는 경우이다. $v^-(e)$를 edge e의 입력 vertex라고 하자. 고착이 발생하는 조건은 다음과 같다.

고착 발생조건

다음 두 조건을 모두 만족하는 사이클 $(e_1, e_2, e_3, \ldots, e_l)$이 존재하면 고착이 발생한다.

조건1) $v^-(e_1), v^-(e_2), \ldots, v^-(e_l)$*에 차량들* $n(e_1)$, $n(e_2)$, $\ldots, n(e_l)$*이 동시에 있고,*

조건 2) 이들 차량들의 다음 라우트 순번이 각각 $s^-(e_1)$, $s^-(e_2), \ldots, s^-(e_l)$*이다.*

<그림2>에서 현재 차량들이 Z_1, Z_2, Z_5, Z_8에 있으므로, 이들 존에 해당하는 vertex들은 이러한 조건의 사이클을 이루고 있지 않다. 그러나 미래에 고착이 발생될 수 있는 가능성이 있다. 예를 들어, 주어진 라우트를 수행하는 도중 Z_6과 Z_7에 각각 다음 라우트 순번 4와 2를 수행하려고 하는 차량 A_1과 A_4가 있을 수 있고, 이럴 경우, 고착 발생조건을 만족하게 된다. 만약, 이러한 고착 발생조건을 만족하는 미래 상태가 필히 발생된다면 이는 임박고착이 된다. 이러한 임박고착은 차량의 운행 시간이 확정적이어서 미래 상태를 정확히 예측할 수 있는 경우에 쉽게 탐지될 수 있다.

만약, 차량의 운행시간을 정확히 예측할 수 없고, 운행 도중 차량들을 잠시 대기시킬 수 있어 운행시간에 내, 외적 불확실성을 갖는다면 임박고착을 정확히 예측할 수 없다. 그러나 임박고착을 방지할 수 있는 방법 중의 하나는 고착 발생 가능성이 있는 사이클의 존들에 대하여 위의 두 조건이 앞으로 만족되지 않도록 하는 것이다. 사이클을 이루는 존에서 한 차량이 동시에 두 존 이상에 있을 수 없으므로, 위의 고착 발생조건 1)은 다음 조건을 포함한다.

조건3) $n(e_1) \neq n(e_2) \neq \ldots \neq n(e_l)$

본 연구에서는 조건 3)을 만족하는 사이클, 즉 서로 다른 차

표 2. 유의사이클

유의사이클(S)	$V(S)$	$(n(e), s(e))$
$S_{0.1}$	Z_1, Z_3	(A_1,1), (A_4,5)
$S_{0.2}$	Z_2, Z_3	(A_2,1), (A_3,2)
$S_{0.3}$	Z_3, Z_4, Z_6, Z_5	(A_1,2), (A_2,3), (A_4,3), (A_3,1)
$S_{0.4}$	Z_3, Z_4, Z_6, Z_5	(A_2,2), (A_1,3), (A_4,3), (A_3,1)
$S_{0.5}$	Z_6, Z_7	(A_1,4), (A_4,2)
$S_{0.6}$	Z_6, Z_7	(A_2,4), (A_4,2)
$S_{0.7}$	Z_7, Z_8	(A_1,5), (A_4,1)
$S_{0.8}$	Z_7, Z_8	(A_2,5), (A_4,1)

량의 특정 라우트에 의한 사이클을 유의사이클(significant cycle)이라고 부르기로 한다. <그림 2>의 EDG로부터 구한 모든 유의사이클은 <표 2>와 같다. 이 유의사이클들을 구하는 방법은 단순하다. EDG에서 한 vertex가 시작과 끝이 되는 패스인 사이클을 모두 구한 후 고착 발생조건 3)을 만족하는 사이클만을 유의사이클로 한다. 예를 들어, <그림 2>에서의 사이클 (Z_3, (A_1, 2), Z_4, (A_1, 3), Z_6, (A_4, 3), Z_5, (A_4, 4), Z_3)는 사이클을 이루는 edge의 차량번호가 중복되어 조건 3)을 만족하지 않는다. 따라서 이러한 사이클은 유의사이클로 하지 않는다. 이에 대한 알고리듬 분석은 4.1절에서 설명된다.

위에서 설명한 동적인 EDG를 이용하여 임박고착을 방지하는 정책은 다음과 같다. EDG에서 l개의 vertex로 구성된 유의사이클 (e_1, e_2, e_3, ..., e_l)에 대하여 차량들 (n_1, n_2, n_3, ..., n_l)중 한 차량 $n(e_k)$가 다음에 수행하여야 할 라우트 순번이 $s(e_k)-1$이라고 하자. 만약, 이를 수행했다고 가정 했을 경우 (즉 $n(e_k)$가 $v^-(e_k)$에 있을 경우), 고착발생 조건 1), 2)를 만족하면, 그 라우트 순번을 수행하지 않고, 현재 존에 대기 시킨다. 즉, 사이클을 이루는 존들에 대하여 $s(e_k)-1$의 라우트 순번을 수행하는 차량 $n(e_k)$들이 최대 $l-1$개 있도록 수를 제한한다. 유의사이클을 EDG $G=(V, E)$라고 하면, 이 제한되는 AGV 수는 $|V|-1=|E|-1$과 동일하다.

현재 상태가 고착이 아니라면, 이러한 정책은 고착 발생 조건 1), 2)를 만족하는 상태에 도달하지 않도록 한다. 그러나 유의사이클을 이루는 존들에 차량의 수를 제한함으로써 이에 따른 부가적인 고착인 제약고착이 발생될 수 있다.

<그림 1>의 예에서 Z_6과 Z_7은 A_1과 A_4의 라우트에 의해 유의사이클($S_{0.5}$)을 이루고, 또한, Z_7과 Z_8역시 동일한 차량들에 의한 유의사이클($S_{0.7}$)을 이룬다. 만약, A_1이 Z_6에서 4번째 라우트 순번을 수행하려고 하고, A_4가 Z_8에서 1번째 라우트 순번을 수행하려고 한다고 하자. 고착 발생 조건을 만족시키지 않기 위하여 각 사이클에 하나의 차량만 있도록 제한하면 어느 차량도 Z_7으로의 이동이 불가능한 상태가 되어 제약고착이 발생된다. 이 경우는 두 사이클이 Z_7을 공유하고, Z_7에서의 edge들이 같은 차량의 연속적인 라우트 순번에 의해 입출력으로 작용할 때 발생한다. 구체적으로, 공유 vertex Z_7에서 $S_{0.5}$의 (A_1, 4)에 해당하는 입력 edge와 $S_{0.7}$의 (A_1, 5)에 해당하는 출력 edge가 A_1에 의한 연속적 라우트이고, $S_{0.7}$의 (A_4, 1)입력 edge와 $S_{0.5}$의 (A_4, 2)출력 edge가 A_4의 연속적 라우트이다. 이 경우에 제약고착을 방지하기 위한 방법은 다음과 같다. 두 유의사이클을 합친 vertex들, 즉 Z_6, Z_7, Z_8에 사이클 내의 edge에 해당하는 라우트 순번을 수행하는 A_1, A_4 중 한 차량만 있도록 수를 제한한다.

이와 같이 EDG에서 다수의 유의사이클들이 서로 차량과 vertex를 공유하는 경우, 제약고착을 방지하는 부가적인 절차가 필요하다. 이를 위하여 CDG에서는 사이클 축소 규칙(cycle reduction rule)에 따라 모든 사이클이 합해진 새로운 결합사이클(CDG에서는 이를 하나의 vertex로 축소하여 macro node라고 부르나, 여기서는 결합 사이클이라고 부르기로 한다)을 생성한다. 이때 결합사이클의 용량, 즉 결합사이클에 동시에 있을 수 있는 특정 차량의 수를 제한한다. 유사한 방법을 EDG에 적용할 수 있으나, 부가적인 절차가 요구된다. 우선 EDG에서의 결합사이클에 대한 정의를 다음과 같이 한다.

정의: 결합사이클(combined cycle)

결합사이클 $S=(V,E)$는 하나의 유의사이클 이거나 또는 두 개 이상의 유의사이클이 합해져 생성되는EDG로 다음의 특성치를 갖는다.

$Agv(S)$: $n(e), e\in E$들의 집합으로 S에 있을 수 있는 차량 번호 집합을 의미한다.

$C(S)$: S의 용량으로서 고착을 방지하기 위하여 제한되는 $Agv(S)$들의 수를 의미한다.

이 정의에 따르면, 하나의 유의사이클도 결합사이클에 속한다. 이미 설명된 바와 같이 유의사이클 S의 용량은 $C(S)=|V|-1=|E|-1=|Agv(S)|-1$이다.

EDG의 경우도 CDG에서와 같이 결합사이클들을 구할 수 있으나, 보다 제한적이다. 두 개의 결합사이클들이 합해져 새로운 결합사이클을 구성하는 생성 조건과 생성 규칙은 다음과 같다.

결합사이클 생성 조건

1) 두 결합사이클 S_1, S_2가 한 vertex, v_k만을 공통으로 포함하고 있다.

2) 다음 조건을 만족하는 S_1의 공통vertex, v_k에서의 입력과 출력edge인 $e_{11}=(x_1, v_k)$, $e_{12}=(v_k, y_1)$, 그리고 S_2의 공통 vertex, v_k에서의 입, 출력 edge인 $e_{21}=(x_2, v_k)$, $e_{22}=(v_k, y_2)$이 존재한다.

$$n(e_{11})=n(e_{22})$$
$$n(e_{12})=n(e_{21})$$
$$s(e_{22})=s(e_{11})+1$$
$$s(e_{12})=s(e_{21})+1$$

결합사이클 생성 규칙

두 결합사이클, $S_1 = (V_1, E_1)$, $S_2 = (V_2, E_2)$에 대하여 결합사이클 생성 조건 1), 2)를 만족할 때 이를 다음과 같은 결합사이클 $S_3 = (V_3, E_3)$로 생성한다.

$$V_3 = V_1 + V_2$$
$$E_3 = E_1 + E_2$$
$$C(S_3) = C(S_1) + C(S_2) - 1$$
$$Agv(S_3) = Agv(S_1) + Agv(S_2)$$

이미 설명한 바와 같이 임박고착을 방지하기 위하여 주어진 EDG에서 m개의 모든 유의사이클을 구한 후, 사이클의 존에 동시에 있을 수 있는 특정의 차량 수를 제한한다. 유의사이클에 차량 수를 제한함으로써 발생하는 제약고착을 방지하기 위하여는 두 유의사이클이 결합되는 경우를 조사하여 결합된 사이클의 존들에 특정 차량 수를 제한하여야 한다. 같은 논리로, 두 유의사이클이 결합된 존에 차량 수를 제한하여 발생하는 제약고착을 방지하기 위하여는 3개 또는 4개의 유의사이클이 합쳐지는 결합사이클을 구하여 제한될 특정 차량 수를 결정하여야 한다. 즉, 2, 3, 4, ⋯, m개의 유의사이클이 합쳐지는 모든 경우를 조사하여 제한될 차량 수를 결정함으로써 순차적으로 발생되는 모든 제약고착을 방지할 수 있다.

2개의 유의사이클이 합쳐질 수 있는 조건은 결합사이클 생성 조건 1), 2)를 모두 만족해야 함은 자명하다. 만약, 2개의 유의사이클 S_1, S_2에 결합사이클 생성조건 2)를 만족하는 vertex의 수가 조건 1)과 다르게 t개 존재한다면, 두 유의사이클을 합친 S_3의 용량 $C(S_3)$는 $C(S_1) + C(S_2) - t = |Agv(S_1)| + |Agv(S_2)| - 2 - t$이다(Yim *et al.*(1997)의 CDG 사이클 축소 규칙 참조). 또한, 두 사이클의 edge를 유발한 차량의 수, $Agv(S_3)$는 두 사이클이 $2 \times t$개의 차량을 공유하고 있으므로 $|Agv(S_1)| + |Agv(S_2)| - 2 \times t$이다. t가 2보다 크거나 같으면, $C(S_3) \geq |Agv(S_3)|$가 되어 용량을 설정할 필요가 없게 된다. 즉, 하나의 vertex만이 조건 2)를 만족하게 되는 경우만을 고려하면 된다.

이를 좀더 설명하기 위하여 <그림 3>의 EDG를 예로 들자. 이 EDG에서 모두 4개의 사이클이 존재하나 고착발생 조건 3)을 만족하는 유의사이클은 <표 3>에서 보듯이 S_1과 S_2 두 개이다. 이 두 유의사이클은 두 vertex Z_1, Z_3를 공유하고 있고, 두 공유 vertex가 결합사이클 생성 조건 2)를 만족하고 있다. 즉, 결합사이클 생성 조건 1)과 다르게 조건 2)를 만족하는 공유

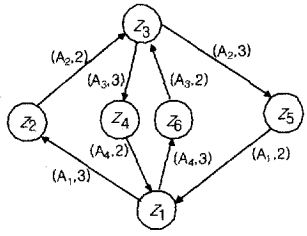

그림 3. 두개의 유의 사이클을 갖는 EDG.

vertex가 2개 존재한다(t=2). <표 3>에 나타난 바와 같이 유의사이클에서의 고착을 방지하기 위해 제한되어야 하는 차량의 수(용량)는 각각 3이다. 두 유의사이클이 합해진 결합사이클 S_3의 용량은 $4(C(S_3) = C(S_1) + C(S_2) - t = 3 + 3 - 2 = 4)$로써 결합사이클의 edge들을 유발한 차량의 수와 같다($|Agv(S_3)| = |Agv(S_1)| + |Agv(S_2)| - 2 \times t = 4 + 4 - 2 \times 2 = 4$). 따라서 결합사이클 S_3의 용량을 4로 설정할 필요가 없게 된다.

3개의 유의사이클을 합치기 위하여는 우선 두 개의 유의사이클을 합쳐 하나의 결합사이클을 만들고, 이것과 나머지 사이클을 합치면 된다. 결합사이클과 유의사이클이 합쳐질 때도 위의 결합 조건을 만족하는 경우만을 고려하면 된다. 하나가 아닌 두 개 이상의 vertex가 조건 2)를 만족하면, 유의사이클 결합의 경우와 같이 차량수들이 용량 보다 적거나 같게 되어 용량을 설정할 필요가 없다. 3개 이상의 유의사이클을 합치는 경우도 이와 동일하다.

그러나 결합사이클 생성 조건을 만족하지만, 차량의 수가 계산된 용량 보다 적거나 같게 되는 경우가 발생할 수 있다. 두 결합사이클 S_1, S_2가 합해지는 경우, 각 사이클에서 공유하는 결합사이클 생성 조건 1)의 vertex에서 입출력으로 작용하는 생성 조건 2)의 두 차량을 제외하고, 나머지 차량들이 서로 상이하다면, 두 사이클이 합쳐진 결합사이클, S_3의 용량은 결합사이클 생성 규칙에 따라 $C(S_3) = C(S_1) + C(S_2) - 1$로 이 수는 $|Agv(S_1)| + |Agv(S_2)| - 3 = |Agv(S_3)| - 1$과 같다. 그러나 동일한 차량이 3 이상 있다면, S_1과 S_2에 동시에 있을 수 있는 서로 다른 차량의 수 $|Agv(S_3)|$는 $C(S_3)$보다 적거나 같게 된다. 때문에 이러한 경우에 결합사이클의 용량을 설정할 필요가 없다.

또한, 두 유의사이클 S_i, i=1,2이 합쳐질 때 항상 $C(S_3) = |Agv(S_3)| - 1 \geq C(S_i) = |Agv(S_i)| - 1$이다. 만약, $C(S_3) = C(S_1)$이라면, S_3의 용량을 $C(S_3)$로 설정하는 것만으로 S_1의 용

표 3. 두 개의 공유 vertex를 갖는 두 유의 사이클의 결합

결합 사이클(S)	$V(S)$	$(n(e), s(e))$	$Agv(S)$	$C(S)$
S_1	Z_1, Z_2, Z_3, Z_4	(A_1,3), (A_2,2), (A_3,3), (A_4,2)	A_1, A_2, A_3, A_4	3
S_2	Z_1, Z_6, Z_3, Z_5	(A_4,3), (A_3,2), (A_2,3), (A_1,2)	A_1, A_2, A_3, A_4	3
S_3	Z_1, Z_2, Z_3, Z_4, Z_5, Z_6	(A_1,3), (A_2,2), (A_3,3), (A_4,2) (A_4,3), (A_3,2), (A_2,3), (A_1,2)	A_1, A_2, A_3, A_4	4

량을 $C(S_1)$으로 설정하는 것을 만족시킨다. 때문에, 이러한 유의사이클들의 용량을 설정할 필요가 없게 된다. 이 역시 어느 두 결합사이클이 합쳐질 때도 성립한다.

즉, 다음의 경우에 결합사이클들의 용량을 설정할 필요가 없다.

용량 설정이 불필요한 결합 사이클

두 결합사이클 S_1, S_2를 합쳐 새로운 결합 사이클 S_3를 생성했다면,

1) 만약 $|Agv(S_3)| \le C(S_3)$ 이면 S_3의 용량 설정은 불필요하다.

2) 1)에 해당하지 않고, 만약 $C(S_3) = C(S_i)$, ($i=1$ 또는 2)이면 S_i의 용량 설정은 불필요하다.

4. 알고리듬 분석

주어진 EDG에서 모든 결합사이클을 구하기 위하여는 다음 두 단계를 수행하여야 한다.

1. EDG에서 모든 유의사이클을 구한다.
2. 유의사이클로부터 모든 결합사이클을 구한다.

4.1 유의사이클 구하기

유의사이클을 구하기 위해서는 그래프에서의 일반적인 사이클 구하는 방법을 사용할 수 있다. 즉, 각 vertex에서 시작하여 깊이 우선 탐색(depth-first search) 방법에 의해 사이클을 구한 후 이 사이클이 고착 발생조건 3)을 만족하는지를 조사하는 것이다. 한 vertex v에서 시작하여 v로 끝나는 일반적인 사이클을 깊이 우선 탐색 방법에 의해 구하는 방법은 만약, EDG $G=(V, E)$에서 각 vertex의 출력 edge수가 평균적으로 $|E|/|V|$라면, $(|E|/|V|-1) \times (|E|/|V|-2) \times \cdots \times 1 = (|E|/|V|-1)!$의 복잡성을 갖는다. 따라서, 각 vertex에서 시작하는 모든 사이클을 구하기 위한 복잡성은 $|V| \times (|E|/|V|-1)!$이다. 그러나 유의사이클은 고착 발생조건 3)을 만족하는 것으로 서로 다른 차량의 라우트에 의한 사이클을 의미한다. 따라서, 한 유의사이클에 포함되는 vertex의 수는 전체 차량의 수 보다 적거나 같아야 한다. EDG에 포함된 라우트를 가지고 있는 차량의 수를 A라고 하면, 유의사이클을 구하는 복잡성은 평균적으로 $O(|V| \times (|E|/|V|)^{A+1})$이다.

4.2 결합사이클 구하기

m개의 유의사이클들을 이용하여 모든 결합사이클을 구하기 위하여는 전장에서 설명한 바와 같이 $1, 2, 3, \ldots, m$개의 유의사이클이 합해지는 모든 경우를 조사할 수 있다. 이 경우 복잡성은 다음과 같다.

$$\sum_{i=1}^{m} {}_mC_i = 2^m - 1 = O(2^m)$$

그러나 보다 효율적으로 구하는 방법은 우선 두 개의 유의사이클을 합쳐 생성 조건과 규칙에 따라 결합사이클을 구한 후, 이들을 이용하여 새로운 결합사이클을 반복적으로 구하는 것이다. 예를 들어, 두 유의사이클 S_{01}과 S_{02}를 합쳐 결합사이클 S_{11}을 생성하고, S_{01}과 S_{03}를 합쳐 S_{12}를 생성했다고 하자. S_{11}을 구성하는 S_{01}은 S_{03}와 결합될 수 있으므로, S_{01}, S_{02}, S_{03}을 하나의 결합사이클로 만들 수 있다. 다시 말하면, 결합사이클 S_{11}과 S_{12}는 모두 S_{01}을 포함하고 있으므로, 이를 합쳐 새로운 결합사이클을 만든다. 두 개의 유의사이클이 합쳐진 결합사이클들을 두 개 합치는 것은 결국 3개의 유의 사이클을 합치는 것과 같다. 다음의 알고리듬은 이러한 방법에 의한다.

Algorithm: ComputeCombinedCycles
Input: Significant cycles S_{0i}, $i=1, 2, \ldots, m$
$C(S_{0i})$, $Agv(S_{0i})$, $i=1, 2, \ldots, m$
Process :
Step 1:
1 $k=1$, $l=0$
2 for every (S_{0i}, S_{0j}) **do**
3 if both cycles satisfy the combined cycle creation condition,
4 $l=l+1$
5 Compute $S_{kl} = S_{0i} + S_{0j}$, $C(S_{kl})$, $Agv(S_{kl})$
according to the combined cycle creation rule
6 $U(S_{kl}) = \{i, j\}$, $F(S_{kl}) = FALSE$
8 endif
9 endfor
Step 2:
10 if $l=0$, go to Step 4
11 $l=0$
12 for every (S_{ki}, S_{kj}),
13 $P = U(S_{ki}) \cap U(S_{kj})$
14 if $P \neq \emptyset$,
15 $l=l+1$
16 $S_{k+1,l} = S_{ki} + S_{kj}$
17 $C(S_{k+1,l}) = C(S_{ki}) + C(S_{kj}) - C(S_{k-1,p})$,
where p is only one element in P
18 $Agv(S_{k+1,l}) = Agv(S_{ki}) + Agv(S_{kj})$
19 $U(S_{k+1,l}) = \{j, j\}$, $F(S_{k+1,l}) = FALSE$
20 if $|Agv(S_{k+1,l})| \le C(S_{k+1,l})$, $l=l-1$,
21 else if $C(S_{k+1,l}) = C((S_{ki})$, $F(S_{ki}) = TRUE$
22 if $C(S_{k+1,l}) = C((S_{kj})$, $F(S_{kj}) = TRUE$
23 endif
24 endif

25 **endfor**

26 $k = k+1$

Step 3:

27 Repeat step 2

Step 4:

28 Print S_{kl}, $C(S_{kl})$, $Agv(S_{kl})$ in which $F(S_{kl}) = FALSE$ for all k and l

<표 4>는 <표 2>의 유의사이클을 이용하여 위 알고리듬에 의해 구한 결합사이클들을 나타낸다. 알고리듬의 Step 1은 두 유의사이클을 합쳐 새로운 결합사이클을 생성하는 절차이고, Step 2는 기 생성된 두 결합사이클을 합치는 절차를 나타낸다. Step 1을 설명하기 위하여 <표 4>의 두 유의사이클 $S_{0,2}$와 $S_{0,4}$가 합쳐져 생성된 결합사이클 $S_{1,1}$의 경우를 설명하면 다음과 같다. 두 유의사이클은 한 vertex Z_3을 공유하고, Z_3에서의 입출력 edge들은 다음과 같다.

$S_{0,2}$에서의 입력 edge: $(A_2,1)$ $S_{0,2}$에서의 출력 edge: $(A_3,2)$

$S_{0,4}$에서의 입력 edge: $(A_3,1)$ $S_{0,4}$에서의 출력 edge: $(A_2,2)$

따라서 두 유의사이클은 알고리듬의 줄번호 3에서 명시된 결합사이클 생성조건을 만족하여 새로운 결합사이클 $S_{1,1}$을 결합사이클 생성 규칙에 따라 생성한다.

Step 2에서는 두 결합사이클이 어떠한 하위 결합사이클을 포함하고 있는지를 조사하기 위하여 리스트 $U(S_{kl})$이 사용되었다. 결합사이클 S_{kl}은 k 번째 반복에서 l번째로 생성된 결합사이클로써 $U(S_{kl})=\{i, j\}$는 결합사이클이 전 단계(Step 2의 $k-1$번째 반복)에서의 두 결합사이클, $S_{k-1,i}$와 $S_{k-1,j}$이 합쳐져 구성되어 있다는 것을 의미한다. k번째 반복에서 구한 두 (S_{ki}, S_{kj})에 대하여 $P = U(S_{ki}) \cap U(S_{kj})$는 공집합이거나 또는 성분이 있다면, 단지 하나의 성분만을 포함하고 있음을 유의하라. P가 성분 p를 포함한다면, 이는 두 결합사이클이 모두 $S_{k-1,p}$를 포함하고 있음을 의미한다. 따라서 이 두 결합사이클을 합쳐 새로운 $S_{k+1,\,l} = S_{ki} + S_{kj}$을 생성한다. 알고리듬은 용량 설정이 불필요한 결합사이클의 제거를 위하여 불린(boolean) 값을 갖는 $F(S_{kl})$를 각 결합사이클, S_{kl}에 할당하였다. 알고리듬 줄번호

표 4. 결합사이클

결합사이클(S)	$U(S)$	$V(S)$	$(n(e), s(e))$	$Agv(S)$	$C(S)$	$F(S)$
$S_{0,1}$		Z_1, Z_3	$(A_1,1)$, $(A_4,5)$	A_1, A_4	1	F
$S_{0,2}$		Z_2, Z_3	$(A_3,2)$, $(A_2,1)$	A_2, A_3	1	F
$S_{0,3}$		Z_3, Z_4, Z_6, Z_5	$(A_1,2)$, $(A_2,3)$, $(A_4,3)$, $(A_3,1)$	A_1, A_2 A_3, A_4	3	T
$S_{0,4}$		Z_3, Z_4, Z_6, Z_5	$(A_2,2)$, $(A_1,3)$, $(A_4,3)$, $(A_3,1)$	A_1, A_2 A_3, A_4	3	T
$S_{0,5}$		Z_6, Z_7	$(A_1,4)$, $(A_4,2)$	A_1, A_4	1	T
$S_{0,6}$		Z_6, Z_7	$(A_2,4)$, $(A_4,2)$	A_2, A_4	1	T
$S_{0,7}$		Z_7, Z_8	$(A_1,5)$, $(A_4,1)$	A_1, A_4	1	T
$S_{0,8}$		Z_7, Z_8	$(A_2,5)$, $(A_4,1)$	A_2, A_4	1	T
$S_{1,1}$	{2, 4}	Z_2, Z_3, Z_4, Z_6, Z_5	$(A_3,2)$, $(A_2,1)$, $(A_2,2)$, $(A_1,3)$ $(A_4,3)$, $(A_3,1)$	A_1, A_2 A_3, A_4	3	T
$S_{1,2}$	{3, 6}	Z_3, Z_4, Z_6, Z_5, Z_7	$(A_1,2)$, $(A_2,3)$, $(A_4,3)$, $(A_3,1)$ $(A_2,4)$, $(A_4,2)$	A_1, A_2 A_3, A_4	3	T
$S_{1,3}$	{4, 5}	Z_3, Z_4, Z_6, Z_5, Z_7	$(A_2,2)$, $(A_1,3)$, $(A_4,3)$, $(A_3,1)$ $(A_1,4)$, $(A_4,2)$	A_1, A_2 A_3, A_4	3	T
$S_{1,4}$	{5, 7}	Z_6, Z_7, Z_8	$(A_1,4)$, $(A_4,2)$, $(A_1,5)$, $(A_4,1)$	A_1, A_4	1	F
$S_{1,5}$	{6, 8}	Z_6, Z_7, Z_8	$(A_2,4)$, $(A_4,2)$, $(A_2,5)$, $(A_4,1)$	A_2, A_4	1	F
$S_{2,1}$	{1, 3}	Z_2, Z_3, Z_4, Z_6, Z_5, Z_7	$(A_3,2)$, $(A_2,1)$, $(A_2,2)$, $(A_1,3)$ $(A_4,3)$, $(A_3,1)$, $(A_1,4)$, $(A_4,2)$	A_1, A_2 A_3, A_4	3	T
$S_{2,2}$	{2, 5}	Z_3, Z_4, Z_6, Z_5, Z_7, Z_8	$(A_1,2)$, $(A_2,3)$, $(A_4,3)$, $(A_3,1)$ $(A_2,4)$, $(A_4,2)$, $(A_2,5)$, $(A_4,1)$	A_1, A_2 A_3, A_4	3	F
$S_{2,3}$	{3, 4}	Z_3, Z_4, Z_6, Z_5, Z_7, Z_8	$(A_2,2)$, $(A_1,3)$, $(A_4,3)$, $(A_3,1)$ $(A_1,4)$, $(A_4,2)$, $(A_1,5)$, $(A_4,1)$	A_1, A_2 A_3, A_4	3	T
$S_{3,1}$	{1, 3}	Z_2, Z_3, Z_4, Z_6, Z_5, Z_7, Z_8	$(A_3,2)$, $(A_2,1)$, $(A_2,2)$, $(A_1,3)$ $(A_4,3)$, $(A_3,1)$, $(A_1,4)$, $(A_4,2)$ $(A_1,5)$, $(A_4,1)$	A_1, A_2 A_3, A_4	3	F

20, 21, 22는 제거를 위한 절차를 나타낸다.

Step 2를 좀더 설명하기 위하여 <표 4>의 $S_{1,1}$과 $S_{1,3}$이 합쳐져 $S_{2,1}$을 생성하는 예를 들어보자. 두 리스트 $U(S_{1,1})=\{2,4\}$와 $U(S_{1,3})=\{4,5\}$로부터 $S_{1,1}$은 $S_{0,2}$와 $S_{0,4}$가 결합된 것이고, $S_{1,3}$은 $S_{0,4}$와 $S_{0,5}$가 결합된 것임을 알 수 있다. 즉, 두 결합 사이클은 $S_{0,4}$를 모두 포함하여 줄번호 13에서 $P = U(S_{1,1}) \cap U(S_{1,3}) = \{2,4\} \cap \{4,5\} = \{4\}$가 된다. 줄번호 16에서는 두 결합사이클을 합쳐 $S_{2,1}$의 vertex와 edge집합을 구한다. 줄번호 17에서는 $S_{2,1}$의 용량을 CDG의 사이클 축소 규칙(Yim *et al.*, 1997)에 따라 구하는 것으로 두 결합 사이클의 용량을 더한 값에 공유되는 결합사이클의 용량을 뺀 값이다($C(S_{2,1}) = C(S_{1,1}) + C(S_{1,3}) - C(S_{0,4}) = 3+3-3=3$). 줄번호 20에서 $S_{2,1}$의 차량수 4는 용량 3보다 크므로 줄번호 21과 22를 수행한다. 결합되는 두 사이클 $S_{1,1}$과 $S_{1,3}$의 용량이 모두 3이므로 $S_{2,1}$의 용량과 같다. 따라서 $S_{1,1}$과 $S_{1,3}$의 F값을 TRUE로 하여 줄번호 28에서와 같이 이 둘을 최종적인 결합사이클에서 제거하게 된다.

이 알고리듬이 Step 2의 제한된 반복 후에 종료되는 유한성을 갖는 것은 자명하다. 유의사이클(S_{0i})의 수를 m이라 하면, 이들 중 둘을 합친 Step 1에서의 결합 사이클 S_{1i}의 수 m_1은 최대 ${}_mC_2$이다. Step 2에서는 이들을 다시 반복적으로 합치므로, 두 S_{1i}를 합친 S_{2i}의 수 m_2는 최대 ${}_{m_1}C_2$이나, S_{2i}는 결국 3개의 S_{0i}들이 합친 것과 같으므로, 최대 ${}_mC_3$이다. 때문에, Step 2의 각 반복에서 구한 S_{ki}의 수 m_k는 최대 ${}_mC_{k+1}$이다. $m-1$번째 반복에서의 m_{m-1}은 최대 ${}_mC_m=1$이므로 더 이상의 반복은 없게 된다.

Step 2의 $k-1$번째 반복에서 구한 결합사이클 $S_{k-1,i}$의 수가 m_{k-1}이라면, k 번째 반복에서 조사하여야 할 경우의 수는 ${}_{m_{k-1}}C_2$이다. 즉, 위 알고리듬의 복잡성은 다음과 같다.

$$ {}_mC_2 + {}_{m_1}C_2 + {}_{m_2}C_2 \cdots + {}_{m_{m-1}}C_2 $$

만약, $m_1 \geq m_2 \geq m_3 \geq \cdots \geq m_m$이고, m_1이 ${}_mC_2$라기 보다 $c \times m$으로 m에 대해 선형관계를 갖는다면, 다음의 복잡성을 갖는다.

$$ O({}_mC_2 + (m-2)\,{}_{c\times m}C_2) = O(m^3) $$

한 EDG로부터 구한 유의사이클이 특정의 두 차량만에 의한 m개가 존재한다면, 이 유의사이클들이 가장 많이 합쳐질 수 있는 경우는 사이클들이 일렬로 결합되는 것으로 가장 앞의 S_{01}은 뒤에 있는 S_{02}와 결합될 수 있고, S_{02}는 앞에 있는 S_{01}, 그리고 뒤에 있는 S_{03}와 결합되고, S_{03}는 앞의 S_{02}, 그리고 뒤의 S_{04}와 결합되는 것이다. 최종적으로 가장 뒤의 S_{0m}은 앞에 있는 $S_{0,m-1}$과 결합된다. 이럴 경우, $m_1 \leq m-1$, $m_2 \leq m-2$, ..., $m_{m-1} \leq m-(m-1)$로 위의 가정을 만족한다. 3 이상의 차량들에 의한 유의사이클이 있을 때, 만약 사이클들이 결합사이클 생성 조건을 만족하여 합쳐지는 경우가 위의 두 차량의 경우와 유사하게 제한적이라면, 위의 가정에 따를 가능성이 커진다.

5. 결합사이클을 이용한 고착 방지 정책

결합사이클을 이용하여 고착을 방지하는 정책은 이미 언급되었으나, 이 장에서는 보다 세부적으로 정책을 설명하기로 한다. 고착 방지 정책은 다음의 두 가지로 구성된다.

1) 라우트 안전성 조사
2) 고착 방지

5.1 라우트 안전성 조사

쉬고 있는 차량들에게 명령을 하달할 때 명령에 상응하는 라우트를 선정하였다고 하자. 라우트 안전성 조사는 명령에 상응하는 라우트로 인하여 고착이 발생하고 있는지를 조사하는 것이다. 만약, 이미 고착이 발생되었다면, 다른 라우트를 선정하여야 하거나, 또는 다음 사건까지 라우트 선정을 미루는 방법이 있을 것이다. 라우트 안정성 조사 방법은 다음과 같다. 쉬고 있는 차량에게 주어질 라우트와 현재 운행 중인 차량들의 남아있는 라우트를 바탕으로 EDG를 생성한 후 모든 결합사이클을 구한다. 다음과 같은 시스템 상태를 실시간으로 알고 있다고 하자.

$status(Z_i)$: 존 Z_i에 있는 차량 수
$nStep(A_i)$: 차량 A_i의 현재 라우트 순번

라우트의 선정 대상이 되는 차량의 현재 라우트 순번은 초기치를 0으로 한다. 각 차량의 현 라우트 순번을 이용하여 각 결합사이클에 포함된 차량 수를 다음과 같이 초기화 시킨다.

$nAgvs(S_i)$: 결합 사이클 S_i에서 $s(e) = nStep(A_i) + 1$, $n(e) = A_i$, $e \in E(S_i)$를 만족하는 차량 A_i들의 수

만약, 어느 S_i에 대해 $nAgvs(S_i) > C(S_i)$이면, 이미 고착 또는 제약고착이 발생되었다는 것을 의미한다. 때문에, 쉬고 있는 차량에게 할당할 라우트를 취소시킨다.

<그림 1>의 예에서 3 차량들 A_2, A_3, A_4는 <표 1>에 명시된 라우트 명령을 방금 하달 받았다는 가정하에 A_1의 라우트에 대해 안전성을 조사한다고 하자. 현재 상태는 다음과 같다.

$$ nStep(A_i) = 0,\ i = 1,2,3,4 $$

$$ status(Z_i) = \begin{cases} 1, & i = 1,2,5,8 \\ 0, & i = 3,4,6,7,9 \end{cases} $$

이 상태 정보로부터 <표 5>와 같이 $nAgvs(S_i)$를 구할 수 있다. 이 표로부터 $nAgvs(S_i) > C(S_i)$를 만족하는 결합사이클이 없으므로 A_1의 라우트는 안전하다.

표 5. 차량 A_1의 라우트 안전성 조사를 위한 계산

결합 사이클(S)	$nAgv(S)$	$C(S)$
$S_{0,1}$	1	1
$S_{0,2}$	1	1
$S_{1,4}$	1	1
$S_{1,5}$	1	1
$S_{2,2}$	2	3
$S_{3,1}$	3	3

5.2 고착 방지

고착 방지는 라우트 안전성 조사를 통과한 차량들이 현 존에서 다음 존으로 이동할 때 마다 실행된다. 차량의 라우트와 안전성 조사에서 구한 결합사이클로부터 다음과 같은 정보를 추출할 수 있다.

$Zone(A_i, k)$: 차량 A_i의 k번째 라우트 순번의 목적지에 해당하는 존

$Arrival(A_i, k)$: 차량 A_i의 k번째 라우트 순번이 입력으로 작용하는 결합사이클 집합

$Depart(A_i, k)$: 차량 A_i의 k번째 라우트 순번이 출력으로 작용하는 결합사이클 집합

이 논문에서 사용된 예제로 설명하면, $Zone(A_i, k)$는 <표 1>로부터 구할 수 있다. 예를 들어, $Zone(A_3, 0)$는 Z_5, $Zone(A_3, 1)$은 Z_3, $Zone(A_3, 2)$는 Z_2로 라우트 순번은 0 부터 2까지 존재한다. $Arrival(A_i, k)$와 $Depart(A_i, k)$는 <표 6>에 나타난 바와 같이 <표 4>의 결합사이클로부터 구할 수 있다. 한 결합사이클 S_i에서 각 차량 $A_i \in Agv(S_i)$에 대해 $n(e)=A_i$인 edge 중에서 가장 작은 수의 $s(e)$를 찾아 S_i를 $Arrival(A_i, s(e))$에, 그리고 가장 큰 수의 $s(e)$를 찾아 S_i를 $Depart(A_i, s(e))$에 속하도록 한다.

표 6. 각 차량의 결합 사이클 입출력

차량 번호	입 력	출 력
A_1	$Arrival(A_1,1)=\{S_{01}\}$ $Arrival(A_1,2)=\{S_{22}\}$ $Arrival(A_1,3)=\{S_{31}\}$ $Arrival(A_1,4)=\{S_{14}\}$	$Depart(A_1,1)=\{S_{01}\}$ $Depart(A_1,2)=\{S_{22}\}$ $Depart(A_1,5)=\{S_{14}, S_{31}\}$
A_2	$Arrival(A_2,1)=\{S_{02}, S_{31}\}$ $Arrival(A_2,3)=\{S_{22}\}$ $Arrival(A_2,4)=\{S_{15}\}$	$Depart(A_2,1)=\{S_{02}\}$ $Depart(A_2,2)=\{S_{31}\}$ $Depart(A_2,5)=\{S_{15}, S_{22}\}$
A_3	$Arrival(A_3,1)=\{S_{22}, S_{31}\}$ $Arrival(A_3,2)=\{S_{02}\}$	$Depart(A_3,1)=\{S_{22}\}$ $Depart(A_3,2)=\{S_{02}, S_{31}\}$
A_4	$Arrival(A_4,1)$ $=\{S_{14}, S_{15}, S_{22}, S_{31}\}$ $Arrival(A_4,5)=\{S_{01}\}$	$Depart(A_4,2)=\{S_{14}, S_{15}\}$ $Depart(A_4,3)=\{S_{22}, S_{31}\}$ $Depart(A_4,5)=\{S_{01}\}$

만약, 차량 A_i가 다음 존으로의 이동을 요청하였다고 하자. 고착 방지 알고리듬은 다음과 같다.

Algorithm : DeadlockAvoidance

Step 1:

1 $k = nStep(A_i)+1$
2 if $status(Zone(A_i, k))=1$, go to step 3
3 for every combined cycle S_i in $Arrival(A_i, k+1)$
4 if $nAgvs(S_i) \geq C(S_i)$, go to step 3
5 endfor

Step 2:

6 $status(Zone(A_i, k-1))=0$
7 $status(Zone(A_i, k))=1$
8 for every combined cycle S_i in $Arrival(A_i, k+1)$
9 $nAgvs(S_i) = nAgvs(S_i)+1$
10 endfor
11 for every combined cycle S_i in $Depart(A_i, k)$
12 $nAgvs(S_i) = nAgvs(S_i)-1$
13 endfor
14 $nStep(A_i) = nStep(A_i)+1$
15 Give a command "Move to the next zone, $Zone(A_i, k)$" to A_i
16 Stop

Step 3:

17 Give a command "Wait at the current zone, $Zone(A_i, k)$" to A_i
18 Stop

이 알고리듬을 설명하기 위하여 <그림 1>의 예를 보자. 네 차량 모두 <표 1>의 라우트를 명령 받아 현재 라우트 순번은 0이다. 차량 A_1이 다음 라우트 순번의 수행을 시도한다고 하자. Step 1은 A_1이 다음 라우트 순번을 수행하였을 때 고착 또는 제약고착을 발생시키는지를 조사하는 것이다. 알고리듬의 줄번호 1에서 $k=1$이다. 줄번호 2에서 $Zone(A_1, 1)$은 이동할 다음 존인 Z_3으로 현재 $status(Z_3)$은 0이다. 만약 $status(Z_3)$이 1이라면 이미 Z_3에 다른 차량이 있다는 결과가 되어 존 조정에 의해 A_1의 이동은 유보된다. 줄번호 3에서 $Arrival(A_1, 2)=\{S_{2,2}\}$이므로 <표 5>로부터 $nAgvs(S_{2,2})=2 < C(S_{2,2})=3$이 성립하여 고착 발생이 없다는 결과를 가져온다. 즉, A_1이 Z_3로 이동했다고 가정했을 때 이 이동에 의해 영향을 받는 결합사이클의 차량수가 용량보다 적다면 고착이 없으나 크거나 같다면 고착이 발생했다는 결과가 되어 차량의 이동은 유보된다. 이러한

정책은 근본적으로 한 차량의 이동이 고착 발생 조건을 만족하는지 그리고, 제약고착을 발생시키는지를 정확하게 예측하는 방법론에 근거하여 필히 고착이 발생되는 경우만을 고려한 방법으로 보수적인 방법에 비해 보다 효과적이다.

알고리듬의 Step 2는 Step 1의 고착 발생 조사를 안전하게 통과한 차량들에게 이동 명령을 하달하여 이에 따른 상태 변수들을 수정하는 절차를 나타낸다.

6. 결 론

본 연구에서는 존 조정을 사용하는 AGV 시스템에서 상충과 고착을 방지하는 정책을 제시하였다. 기본적인 방법은 기존의 CDG를 이용하나, 차량의 라우트에 대한 자세한 정보를 포함한 EDG를 제안하여 보수적인 정책이 아닌 보다 효과적인 정책을 유도하였다.

제안된 정책은 두 가지로 구성되어, 라우트의 안전성 조사와 운행 중에 발생할 수 있는 고착을 미리 예측하여 방지토록 하는 실시간 정책으로 구성되어 있다. 라우트의 안전성 조사의 경우 약간의 복잡성을 갖지만, 실제 시스템에서 한 네트워크에서 운영하는 차량의 수가 10이하이고, 그래프 모델로 표현할 때 vertex와 edge의 수가 제한적이라는 점을 감안하면, 실시간적으로 사용하기에 큰 부담이 없을 것이다.

EDG를 이용한 임박고착 및 제약고착의 방지 방법론은 존 조정을 사용하는 어떠한 AGV 패스 네트워크에서도 사용될 수 있는 적용력을 갖고 있다. 때문에, AGV 시스템 뿐만 아니라, 유연생산 시스템에서 작업물과 자원간의 상충에서 발생하는 고착의 방지를 위해서도 효과적으로 이용될 수 있을 것으로 믿어진다. 이에 대한 연구는 계속적으로 수행될 예정이다.

참고문헌

Banaszak, Z. A. and Krogh, B. H. (1990), Deadlock Avoidance in Flexible Manufacturing Systems with Concurrently Competing Process Flows, *IEEE Transactions on Robotics and Automation*, 6, 724-734.

Broadbent, A. J., Besant, C. B., Premi, S. K. and Walker, S. P. (1985), Free Ranging AGV Systems: Promises, Problems and Pathways, *Proc. 2nd Int. Conf. On automated Materials Handling*, IFS(publication) Ltd., UK, 221-237.

Kim, C. W. and Tanchoco, J. M. A. (1991), Conflict-free Shortest-time Bidirectional AGV Routeing, *International Journal of Production Research*, 29 (12), 2377-2391.

Krishnamurphy, N. N., Batta R. and Karwan, M. H. (1990), Developing Conflict-free Routes for Automated Guided Vehicles , *Operations Research*, 41, 1077-1090.

Lee, C. C. and Lin, J. T. (1995), Deadlock Prediction and Avoidance Based on Petri Nets for Zone-Control Automated Guided Vehicle Systems, *International Journal of Production Research*, 33(12), 3249-3265.

Oboth, C., Batta, R. and Karwan, M. H. (1999), Dynamic Conflict-free Roueting of Automated Guided Vehicles, *International Journal of Production Research*, 37(9), 2003-2030.

Reveliotis, S. A. (2000), Conflict Resolution in AGV Systems, *IIE Transaction*, 32(7), 647-659.

Wysk, R. A, Yang, N. S. and Joshi, S. (1991), Detection of Deadlocks in Flexible Manufacturing Cells, *IEEE Transactions on Robotics and Automation*, 7(6), 853-859.

Yeh, M. S. and Yeh, W. C. (1998), Deadlock Prediction and Avoidance for Zone-control AGVS, *International Journal of Production Research*, 36(10), 2879-2889.

Yim, D. S., Kim, J. I. and Woo, H. S. (1997), Avoidance of Deadlocks in Flexible Manufacturing Systems Using Capacity-designated Directed Graph, *International Journal of Production Research*, 35(9), 2459-2475.