한국시뮬레이션학회 논문지
Vol. 18, No. 1, pp. 9-16 (2009. 3)

패트리 넷에서의 교착 상태 확인 알고리즘 성능분석

김종욱[1†] · 이종근[1]

An Performance Evaluation of the Deadlock Detection Algorithm in Petri Nets

Jongwoog Kim · Jongkun Lee

ABSTRACT

Since a deadlock is a condition in which the excessive demand for the resources being used by others causes activities to stop, it is very important to detect and prevent a deadlock. About the deadlock detection analysis methods are may divide like as Siphon, DAPN and transitive matrix, but it's very difficult to evaluate the performance. Since DES (Discrete Event Systems) is NP-hard, and these detection and avoidance methods used various factors in each technique, it's made difficult to compare with each other's. In this paper, we are benchmarked these deadlock detection analyze methods based on the complexity, the detection time and the understanding after approached to one example.

Key words : Algorithm, DAPN, Deadlock, Deadlock free, Petri nets, Siphon, Transitive matrix

요 약

본 연구에서는 교착상태 확인 알고리즘의 성능분석을 위하여 사이폰(siphon) 알고리즘, DAPN알고리즘과 추이적 행렬 알고리즘을 상호 비교한다. 이를 위하여 비교 모델을 설정하여 각 알고리즘을 활용한 결과를 복잡도, 이해도 그리고 신속성 등의 3가지 함수를 이용하여 성능을 분석한다. 서로 다른 개념의 알고리즘을 비교분석에 한계성이 있으나, 동일한 모델에 적용하여 그 효율성을 비교 분석하여 각 알고리즘의 특성들을 분석한다.

주요어 : 알고리즘, 교착회피 패트리 넷, 교착상태, 교착자유상태, 패트리 넷, 사이폰, 추이적행렬

1. 서 론

교착상태란 시스템의 공정 중에 포함 된 여러 작업에서 공동으로 사용되는 공유 자원을 각 작업에서 상호적으로 사용을 기다려 시스템의 작업 공정이 중단 된 상태를 의미하며, 현대의 많은 기술 시스템인 자동생산시스템, 데이터 통신, 다중처리 운영시스템과 분산 데이터베이스 시스템 등에서 가장 잘 알려진 문제이다. 이러한 교착 문제를 해결하기 위하여 패트리 넷(Petri Net)을 활용 한 교착상태 확인 및 회피 방법 연구가 활발하다. 패트리 넷을 이용한 교착 상태 확인 알고리즘에는 특히 사이폰(siphon)을 이용한 교착상태 확인 기법이 가장 많이 사용되고 있으며[1-4,7-10], 관계 행렬을 이용한 기법[6] 그리고 추이적 행렬을 이용한 기법[11,12] 등이 있다. 사이폰이란 입력 플레이스에 의하여 삽입 된 토큰이 다시 입력플레이스로 주입되는 특성을 가진 서브 넷 개념으로 이러한 과정을 통하여 토큰의 수가 줄어들어 교착상태의 발생 여부를 확인할 수 있는 알고리즘이다. 페트리 넷을 관계 행렬로 정의하고 동시 작업 개념으로 이를 통합하여 자원분석을 통하여 교착 여부를 확인하고 방지하기 위한 알고리즘이 DAPN으로 제안되었다[6]. 플레이스와 플레이스간의 관계를 종합적으로 표현이 가능한 추이적 행렬을 이용하여 특히 자원 공유 플레이스에 기초하여 산술적인 해석으로 교착 상태를 확인하는 알고리즘을 추이적 행렬 알고리즘으로 제안되었다[11]. 교착 확인 알고리즘 제안에 있어서 좋

2008년 11월 11일 접수, 2009년 3월 19일 채택
[1)] 창원대학교 컴퓨터공학과
주 저 자 : 김종욱
교신저자 : 이종근
E-mail; jklee@cwnu.ac.kr

은 알고리즘이란 상태 확인을 위한 계산 과정을 단순처리 과정으로 제안되고 이해가 용이하며 보다 짧은 계산 시간을 통해 모든 가능한 해결안을 얻도록 구성되어진 알고리즘이 효율성이 높다고 하겠다[12]. 지금까지 교착상태에 관한 알고리즘들이 많은 연구를 통하여 제안 되었으나 유사한 개념에 의한 상호 비교는 많이 이루어졌다, 예를 들면 사이폰 알고리즘의 경우 Ezpleta[4]의 개념을 Li,J와 Zhou에[8,10]의하여 지속적인 비교를 통하여 발전시키고 있다. 그러나 서로 다른 개념의 교착상태확인 알고리즘의 성능분석 연구는 저자의 검색 노력에도 불구하고 어떠한 연구결과를 찾아 볼 수가 없었다. 따라서 본 연구에서는 처음으로 서로 다른 개념의 교착 상태 확인 알고리즘의 분석을 위하여 가장 많이 연구되고 있는 사이폰, 관계행렬 그리고 본 연구팀에서 제안한 추이적행렬 알고리즘을 성능분석하기 위하여 비교 모델을 설정하고 해결을 위한 복잡도와 이해도 그리고 신속성을 정성적, 정량적으로 비교분석한다.

논문의 구성은 2장에서, 예시 모델에 대해 기술하고, 3장에서는 각 제안 알고리즘의 분석 기법을 정리하고, 4장에서는 성능 평가 계수를 사용하여 분석 한 후 벤치마킹 결과를 보여준다. 마지막으로, 5장에서 결론과 향후 연구과제 등을 보인다.

2. 예시 모델

비교 분석을 위하여 가장 잘 이용되고 있는 모델중의 하나로[6,11,12], 두 개의 작업 유형(J_1과 J_2)과 2대의 기계 r1 및 r2 를 공유하는 모델을 적용한다. 작업 J_1은 세 가지 수순작업을 r2, r1 기기를 각각 사용하며, J_2는 r1, r2 의 순서로 기기를 각각 사용하여 세 가지 수순작업을 행한다. 병합된 작업과정과 기기의 관계를 정리하면 다음과 같다 (그림 1,2):

작업 1(Job 1) : {J1-r2, J1-r1, J1-idle}
작업 2(Job 2) : {J2-r1, J2-r2, J2-idle}

즉 작업 1은 기기 r2를 사용하여 수순작업 t1을 행하고 t2작업을 기기 r1을 사용하고 기기들을 사용가능상태로 만들고(t3) 다시 새 작업을 수행하게 된다. 작업 2는 이와 반대로 기기 r1을 수순작업 t4, 기기 r2를 수순작업 t5에 그리고 다시 기기들을 사용가능하게(t6) 하고 새 작업을 수행하게 된다. 이제 이 모델의 작업 개념도와 패트리 넷을 나타내면 그림 1과 그림 2와 같다.

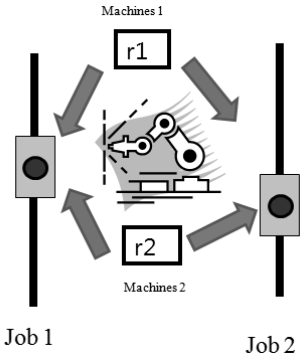

그림 1. 예시 모델 개념도

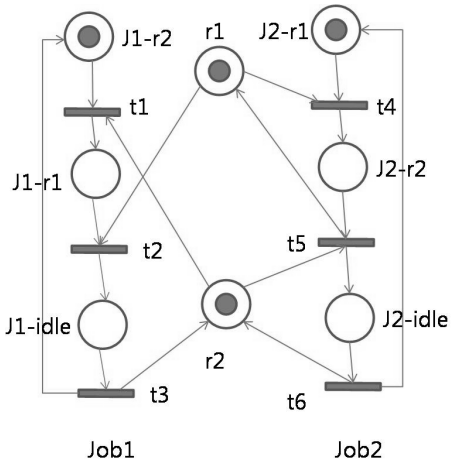

그림 2. 예시 모델의 패트리 넷

3. 비교분석 알고리즘

3.1 사이폰

사이폰과 트랩(trap)은 패트리 넷의 교착상태 특성으로부터 도입된 특별한 구조이다. 사이폰과 트랩을 통하여 패트리 넷의 모든 도달 가능한 마킹을 조사하지 않고도 패트리 넷의 생동성(liveness)을 분석 할 수 있는 특징적 구조이다[3].

[정의 3.1] 사이폰과 트랩

어느 플레이스 집합 S에 대하여 •S⊆ S•를 만족하는 집합 S를 사이폰이라고 한다. 또한 S•⊆•S를 만족하는 집합 S를 트랩 이라고 한다.

사이폰과 트랩을 통하여 교착상태에 대한 정의를 다음과 같이 정리한다[3].

[정의 3.2] 토큰이 모두 사라지는 사이폰 S를 잠재적인

교착(potential deadlock)이라한다.

[정의 3.3] 최소 사이폰만일 하나의 사이폰 S1이 다른 사이폰을 포함하고 있지 않다면 사이폰 S1은 최소 사이폰이라 한다.

[정의 3.4] 어느 패트리 넷에 만일 최소 사이폰이 없다면 이 패트리 넷은 교착상태가 아니며 이를 교착자유상태라 한다.

일반적으로 사이폰을 이용하여 교착상태를 확인하기 위하여서는 그림과 같이 주어진 패트리 넷 모델에서 사이폰가능성의 사이클 서브 넷을 구하여 사이클 서브 넷이 사이폰이나 트랩의 성질을 갖는지 여부를 판별하여 이를 이용하여 교착 상태를 분석한다. 사이폰 알고리즘을 이용하여 교착상태 확인 분석에는 두 단계를 거치는데, 먼저 사이클 상태를 찾아서 사이폰의 부분 넷을 구하고, 구하여진 서브 넷에서 사이폰의 성질 분석을 통하여 교착상태 확인을 꾀한다.

1) 비교 모델(그림 2)에서 사이클 상태를 찾아보면:

S1={J1-r1, r2, J1-idle}
S2={r1, J1-idle}
S3={r2, J1-r1, J1-idle, J2-idle}
S4={r1, J1-idle, J2-r2}
과 같이 4개를 구 할 수 있다.

2) 이제 각 사이클의 성질분석을 하여 보면:

S1:S1은 사이폰:
•S1={t1,t2,t3}
S1•={t1,t2,t3}
•S1⊆S1•
S1은 트랩:
•S1={t1,t2,t3,t6}
S1•={t1,t2,t3}
S1•⊆•S1
S1은 사이폰이며 트랩의 성질을 갖는다.
S2와 S3은 사이폰의 성질만을 갖는 사이클이다.
S2는 사이폰:
•S2={t2,t3}
S2•={t5,t2,t3}
•S2⊆S2•
S3:•S3={t1,t2,t3,t5,t6}
S3은 사이폰:
•S3={t1,t2,t3,t5,t6}
S3•={t1,t2,t3,t5,t6}
S3•⊆•S3
S4도 S1과 같이 사이폰이며 트랩의 성질을 갖는다.
S4는 사이폰:
•S4={t2,t3,t4,t5}
S4•={t2,t3,t4,t5}
•S4⊆S4•
S4는 트랩:
•S4={t2,t3,t4,t5}
S4•={t2,t3,t4,t5}
S4•⊆•S4

S1,S3,S4는 최소 사이폰이면서 초기 마킹 된 트랩을 가지고 있으므로 교착 상태가 아니나, S2는 모든 마킹이 빠져나가는 사이폰이므로 [정의 3.2]에 의하여 이 모델에서 교착상태가 있음을 알 수 있다.

3) 교착상태 확인 알고리즘

사이폰과 트랩을 이용한 교착상태 확인 알고리즘은 특별하게 표현되어진 알고리즘은 없어서 확인하여 가는 과정을 중심으로 작성하여 비교하기로 한다.

<사이폰과 트랩을 이용한 알고리즘>

Input: 패트리 넷
Output: 사이폰, 트랩
step 1. 패트리 넷에서 시작 마킹을 시작으로 사이클 프로세스를 찾는다;
step 2. 마지막 마킹까지를 시작으로 모든 사이클 프로세스를 찾을 때까지 step 1을 반복한다;
step 3. 모든 사이클 프로세스 n개를 구한다;
step 4. 각 사이클 프로세스에서 사이폰과 트랩의 성질을 분석한다;
step 5. 사이클의 성질 분석에서 최소 사이폰이면 교착상태로, 없으면 교착자유상태로 확인;
step 6. 모든 사이클 프로세스 n까지 step 4-5를 반복한다;

위와 같은 알고리즘으로 수행하면 비교 모델의 경우 비교횟수를 추정하면 다음과 같다.

1) 사이클 프로세스 추출 비교 횟수: 12회
2) 사이폰과 트랩성질 분석:
만일 n_i : i 사이클 내의 플레이스 수
i : 사이클 수

비교횟수: $\sum(2*n_i + 2)$

$=32$

결론적으로 최소한 32회의 비교 횟수를 추정 할 수가 있다. 그러나 사이클을 찾는 프로세스 방법에 따라, 비교 횟수는 다양하게 변화 할 수가 있다.

따라서 알고리즘의 복잡성을 분석하면,

1) 사이폰 후보군 검색:

$n(n-1)$

$= n^2 - n$

여기서, n: 플레이스의 수

2) 사이폰과 트랩의 성질을 비교:

m: 사이폰 후보군의 수

pi: 각 i 후보군의 플레이스 수

$$\sum_{i=1}^{m}(2*p_i)+2$$

결국, 사이폰 알고리즘은 $O(n^2)$의 복잡도를 갖는다.

3.2 DAPN

DAPN 알고리즘은[6]에 의하여 패트리 넷 모델의 관련 행렬식에 기초하여 제안되었다. DAPN 알고리즘은 모델을 인접행렬로 표현하고, 차기 마킹을 중심으로 행과 열의 마킹의 흐름을 통하여 0 보다 작은 마킹이 발생 할 때까지 교착 상태를 찾는 알고리즘이다. 이후 알고리즘에서는 그림 2에서의 플레이스 표기를 J1-r2를 p1,J1-r1을 p2,J1-idle를 p3로, J2-r1을 p4,J2-r2를 p5,J2-idle을 p6으로 표시한다.

<DAPN 알고리즘>

Input: 인접행렬

Output: 교착상태

step 1. 행렬의 과정 별로 인접행렬을 작성;

step 2. B_step1을 SV_0로 하여 차례로 자원상태 값 SV(State Value)테이블 작성;

step 3. 자원상태값 테이블에서 자원의 값이 음수인 것을 찾는다. 자원상태값이 음수이면 점화하기 위해 필요한 자원들이 없다는 것을 나타내는 것이며 이는 결국 이 부분에서 교착상태가 발생;

step 4. 교착상태를 일으키는 음수 값을 없앤다. 이를 위해 그 이전 단계에서 자원을 공급하여 음수 값이 나오지 않아야 하므로 SV_i의 교착상태를 없애기 위해 SV_{i-1}의 상태값을 바꾸어 SV_i에서 음수 값이 나오지 않도록 한다;

$$M_{PR} = \begin{array}{c} \begin{matrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & r_1 & r_2 \end{matrix} \\ \begin{bmatrix} 0 & 1_{j1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1_{j1} & 0 & 0 & 0 & 0 & 0 \\ 1_{j1} & 0 & 0 & 0 & 0 & 0 & 1_{j1} & 1_{j1} \\ 0 & 0 & 0 & 0 & 1_{j2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1_{j2} & 1_{j2} & 0 \\ 0 & 0 & 0 & 1_{j2} & 0 & 0 & 0 & 1_{j2} \\ 0 & 0 & 1_{j1} & 0 & 1_{j2} & 0 & 0 & 0 \\ 0 & 1_{j1} & 0 & 0 & 0 & 1_{j2} & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ r_1 \\ r_2 \end{matrix}$$

그림 3. 비교모델의 인접행렬

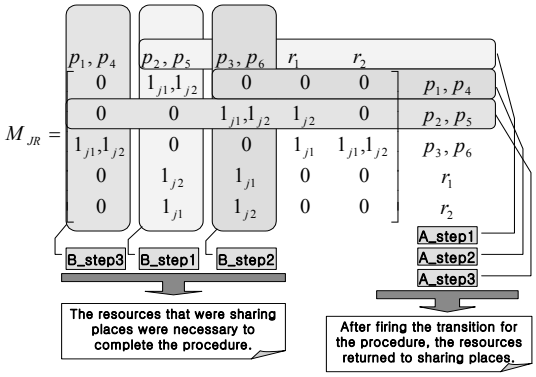

그림 4. 예시의 알고리즘에 대한 단계

step 5. 위 step 3, 4의 과정을 반복;

step 6. 자원 상태값 테이블에서 마지막 자원 상태 값이 SV_0의 자원 상태값과 동일;

step 7. 만약 마지막 자원 상태값이 SV_0의 상태값과 같지 않다면 이는 계속적인 프로세스 진행을 위한 보존성을 만족하지 않으므로 마지막 자원 상태 값을 SV_0의 값과 같도록 수정;

step 8. 수정된 최종 결과를 행렬에 반영하여 이를 토대로 그래프를 작성하면 교착상태를 탐지하여 그 상태가 회피된 최종그래프 표현;

예시 모델 내의 작업1과 작업2의 시작점이 각 작업에 존재하므로, 작업1과 작업2는 동시에 시작되어 수행 될 수 있다. 작업1과 작업2의 동시 과정을 고려하여, 관계행렬을 정리하면 M_{PR}로 표현된다(그림 3). 이를 다시 작업과 관계행렬을 합병하여 정리하여 분석 행렬 : M_{JR}을 얻게 되는데(그림 4) M_{JR}의 상태와 자원 공유 장소는 교착상태, 그 확인 및 방지의 해석이 이들로부터 얻어질 수 있으므로 함께 고려되어야 한다. 이 행렬을 통하여 각 단계의 자원 상태 값을 다음 단계에 기초하여 계산할 수

표 1. 충돌이 발생한 슬롯 당 평균 대역폭 요청 메시지 수

자원공유 장소	초기자원 상태	B_1단계	A_1단계	B_2단계	A_2단계	B_3단계	A_3단계
r1	1 (j1또는j2)	0	0	-1 (j1또는j2)	0	0	1 (j1또는j2)
r2	1 (j1또는j2)	0	0	-1 (j1또는j2)	-1 (j1또는j2)	-1 (j1또는j2)	1 (j1또는j2)

있다(표 1).

B_*i*단계의 상태값 = 사전 단계의 상태값 - B_*i*단계의 상태값

A_*i*단계의 상태값 = 사전 단계의 상태값 + A_*i*단계의 상태값

자원 번호 다음에 있는 괄호 안의 표기된 것은 어떤 작업에 자원이 필요한지를 표시한다. 여기서, B_1단계에서 r1 또는 r2의 값이 음수이면, 작업1과 작업2는 자동으로 음수가 된다. 이들 작업이 음수의 결과를 갖는 것은 필요한 자원이 제공되지 못함을 나타내므로 교착 상태의 이유가 된다. 이 예제에서는 B_2단계에서 r1과 r2가 공히 음수가 나타나므로 교착상태임을 나타내고 있다.

비교모델에서의 DAPN알고리즘의 비교분석은 인접행렬 구성이후에 자원공유 플레이스를 찾고 나서 각 자원공유 플레이스 프로세스 표를 작성하므로 비교가 시작되어지는 것부터 비교횟수를 산정하면 다음과 같다:

1) n: 자원공유 플레이스

2) 자원공유 표 작성: n * 트랜지션수 * job수 * 마킹수
= (2 * 6 * 2) * 3 = 72

DAPN의 경우, Job의 수와 트랜지션의 수(각 Job의 프로세스 수)와 음수 마킹을 찾아 낼 때까지의 마킹 수가 비교 횟수를 결정하게 된다. 따라서 간단한 모델의 경우 1-2 마킹으로 음수 마킹을 찾을 경우 비교 횟수가 적게 나타낼 수 있으나, 복잡한 모델이나 Job이 여러 개나 프로세스가 복잡 할 경우 그 비교 횟수는 비례적으로 크게 증가할 수가 있다.

일반항으로 DAPN 알고리즘을 분석하면:

$n*(t * k) * M$

n : 자원공유 플레이스 수

t : 트랜지션의 수

k : 작업의 수

M : 마킹의 수

따라서 DAPN의 복잡도는 최대 $O(n^4)$가 될 수가 있다.

표 2. 예시 모델의 TM

$M_{PR}'=$

p1	p2	p3	p4	p5	p6	r1	r2		Σ
0	1/2	0	0	0	0	0	0	**P1**	1/2
0	0	1/2	0	0	0	0	0	**P2**	1/2
1	0	0	0	0	0	0	1	**P3**	2
0	0	0	0	1/2	0	0	0	**P4**	1/2
0	0	0	0	0	1/2	1/2	0	**P5**	1
0	0	0	1	0	0	0	1	**P6**	2
0	0	1/2	0	1/2	0	0	0	**r1**	1
0	1/2	0	0	0	1/2	1/2	0	**r2**	3/2
1	1	1	1	1	1	1	2		Σ

3.3 추이적행렬 알고리즘(TM:Transitive matrix)

추이적 행렬 알고리즘은[5,11,12] 패트리 넷에서의 자원공유에 기초하여 모든 플레이스와 플레이스간의 관계를 정리하여 제안하였다. 특히 자원공유 플레이스를 중심으로 열과 행의 토큰 흐름의 값을 산정하여 비교하는 교착상태 여부 판별식을 설정하여 교착상태를 감지하는 알고리즘을 제안하였다.

<추이적행렬 알고리즘>

Input: 인접행렬

Output: 교착상태

step 1. 초기 패트리 넷 PN의 추이적 행렬정리: L^*_{BP}

step 2. 추이적 행렬 L^*_{BP} 의 행의 모든 자원 공유 플레이스를 구함:Rc

step 3. 추이적 행렬 L^*_{BP} 의 열의 모든 자원공유 플레이스를 구함:Rr

step 4. 교착 생태 확인 판별식 $Dr(p_i)=Rr - Rc$ 구함

step 5. 모든 자원공유 플레이스만큼 step 2 - 4를 반복

step 6. 판별식 $\sum_{i=1}^{n} Dr(p_i) = k$ 에서, 만일 k = 0 이나 k가 정수이면 교착자유상태, 그렇지 않으면 교착상태

TM[17]의 정의를 통해 다음과 같은 예시 모델의 추이적 행렬을 얻을 수 있다(표 2). 이 표 M_{PR}에서, 우리는 J1: p1-p2-p3 및 J2: p4-p5-p6와 같은 두 가지 작업을 알 수 있으며, 이 관계는 하나의 주기로 설명된다. 또한, 자원 공유 장소 r1 및 r2 사이의 어떠한 관계를 알 수 있다. 사례 모델에서의 자원공유 플레이스 r1과 r2를 각각 정리하면 다음과 같다.

1) 플레이스 r1:
Rr(r1) = 1/2+1/2=1,
Rc(r1) = 1/2+1/2=1
Dr(r1) = 1 - 1 = 0
2) 플레이스 r2:
Rr(r2) = 1+1= 2
Rc(r2) = 1/2+1/2+1/2=3/2
Dr(r2) = 2 - 3/2 = 1/2

여기서 플레이스 r2는 r1과 달리 판별식이 정수가 아니므로 r2는 교착상태가 되어 사례 모델은 교착상태이다.

이제 마킹의 흐름을 통한 검증을 하여 보면 플레이스 p2와 p4의 경우 정상적으로 토큰을 가지게 되나, p3, p5, p6의 경우 비정상적이 되어 교착상태가 발생하게 된다.

추이적 행렬을 이용한 알고리즘의 비교 횟수를 산정하가 위하여 DAPN과 같이, 추이적 행렬을 구한 후부터 각 자원공유를 이용하여 교착상태를 찾아내는 과정의 비교 횟수를 산출하여보면 다음과 같다:

1) n: 자원공유 플레이스 수
2) 자원공유를 제외한 일반 플레이스 수: m
3) 자원공유 플레이스의 마킹 계산: n*(n+m)
= 2 * (2+6) = 16
4) 음수나 0의 플레이스 확인: n = 2

추이적 행렬 알고리즘에서 최저로 비교횟수는 18회에 달한다. 또한 자원공유 플레이스가 아닌 다른 플레이스의 합에서 음수가 나타 날 경우에도 교착 가능성이 있음을 상기 할 수 있다.

일반항으로 정리하여 보면:

$$n*(n + k) = n2 + n \cdot k$$

n : 자원공유 플레이스 수
k : 자원공유 플레이스를 제외한 플레이스의 수

따라서 추이적 행렬알고리즘의 복잡도는 최대 $O(n^2)$가 될 수가 있다.

4. 알고리즘 벤치마킹

예시를 통해 우리는 다음 인자와 같은 일부 결과를 얻을 수 있다.

표 3. 교착 확인 알고리즘의 복잡성

	siphon	DAPN	추이적행렬
복잡도	$\sum_{i=1}^{n} 2 * P_i$ m: 사이폰의 수 n: 장소 번호 p_i: 각 사이폰의 장소 번호	$N_{*(t*k)*m}$ N: 자원 공유 장소의 번호 T: 전이 회수 K: 작업 번호 M: 표시 번호	n^2+n*k n: 자원 공유 장소 번호 k: 기타 장소 번호
예시 모델	32	72	18

표 4. 알고리즘 효율성 비교표

	사이폰	DAPN	추이적행렬
이해도	보통	난해	용이
복잡도	$O(n^2)$ 32	$O(n^4)$ 72	$O(n^2)$ 18
신속성	보통	느림	빠름

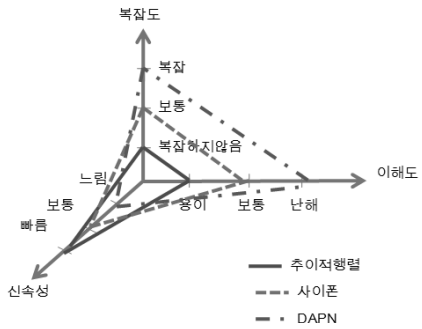

그림 5. 효율성 비교도

1) 이해도(Understand)

사이폰의 서브셋을 구하기 위하여 사이폰의 특성을 분석하는 과정의 알고리즘이 복잡하다. DAPN의 경우 모델에서 작업흐름을 구하고 이를 병렬로 처리하고 자원상태표를 구하여 분석하므로 이해하는 것은 매우 어렵다. TM의 경우 추이적 행렬을 구하고 자원공유부문의 산술식을 해석하여 교착 여부를 결정하므로 알고리즘의 이해가 용이하다.

2) 복잡도(Complexity)

효과성에 관해 알고리즘의 복잡도 값을 고려하는 것은 매우 어렵다. 우리는 각 알고리즘에서 과정의 흐름에 의해서만 이 문제를 피할 수 있다. 사이폰 알고리즘의 복잡도는 $O(n^2)$이며 효과는 예시 모델에서 32이다. 또한, DAPN에서, 복잡도는 $O(n^4)$이며 효과는 72이고, 추이적행렬의 복잡도에서는 $O(n^2)$이며 효과 값은 18이다.

3) 신속성(Detection time)

신속성의 경우 복잡도에서 얻은 것과 같이 DAPN의 처리시간이 72, 사이폰이 32 그리고 추이적 행렬이 18을 예제 모델에서 시간 비교를 얻었다.

결론적으로, 우리는 비교 대상 결과를 표 4로 요약할 수 있다. 이러한 요점에 기초하여 우리는 3개의 알고리즘 사이의 전체 관련 그래프를 도시 할 수 있다(그림 5). 이 그래프를 통해 우리는 추이적 행렬의 최적화가 양호하며 사이폰의 접근 방식은 시간상으로 효과적이라는 것을 알 수 있다.

5. 결론 및 향후 연구

이 연구는 패트리 넷의 교착 확인 알고리즘의 효율성 분석에 초점을 두었다. 우리는 성능 평가를 위해 사이폰, DAPN과 추이적 행렬 알고리즘과의 세 개 형태의 알고리즘을 비교 분석하였다. 여러 다른 개념을 통하고 통일 된 알고리즘 소스가 없어 물리적인 직접 비교 분석방법은 어렵다. 따라서 하나의 예시 모델을 선택하였고 3개의 서로 다른 알고리즘을 적용하여, 신속성, 복잡도 및 이해도와 같은 세 가지 비교 함수에 기초하여 비교하였다.

추이적 행렬 알고리즘이 시스템 내에 많은 자원 공유가 있는 모델에서 사이폰 및 DAPN 접근방식보다 처리 속도가 빠르다고 평가하였지만, 소형의 경우는, 사이폰의 접근 방식이 효율적이다. 이것은 새로운 접근 방식이 모든 경우에서 그리고 모델이 소형 자원 공유 기계 및 짧은 작동 부서를 가지고 있는 경우 많은 자원 공유 기계를 보유한 모델에 더욱 유용하다. 그리고 사이폰에 접근하는 것이 유용하며, DAPN의 알고리즘은 연구 시간이 양호하지만, 이들 접근방식이 최적화를 고려하지 않고, 복잡한 모델이나 각 기계 안의 많은 자원 공유에 사용되는 것은 좋지 않다. 앞으로의 연구방향은 성능분석 함수를 개발하고 교착을 회피한 효율적인 스케줄링 알고리즘의 연구가 필요하다.

참 고 문 헌

1. Corbett JC (1996), “Evaluating Deadlock detection methods for concurrent software”, IEEE tr. Sof. Eng. Vol. 22 (3), pp. 161-180.
2. Damasceno BC. And Xie X. (1999), “Petri nets and deadlock-free scheduling of multiple-resource operations”, *In IEEE SMC’99*, pp. 878-883.
3. F. Chu and X-L. Xie (1997), “Deadlock Analysis of Petri nets using Siphon and Mathematical Programming”, IEEE Tr on Robotics and Automation, Vol. 13, No. 6, pp. 793-804.
4. Ezpleta J.,Colom JM,Martinez J. (1995), “A Petri net based deadlock prevention policy for flexible manufacturing systems”, IEEE tr. Robotics and Automat., Vol. 11, No. 2, pp. 173-184.
5. Liu J.,Itoh Y., Miyazawa I., Seikiguchi T., (1999). “A Research on Petri nets Properties using Transitive matrix”, In: *Proceeding IEEE SMC’99*, pp. 888-893.
6. 송유진, 이종근 (2006), “DAPN과 인접행렬을 이용한 교착상태 회피에 대한 연구”, 한국시뮬레이션학회 논문지, 15권 (1호), pp. 1-10.
7. Y-Sheng Huang (2007), “Design of deadlock prevention supervisors using Petri nets”, In: J.Adv. manuf. Tech., Vol. 35, pp. 349-362.
8. ZW Li, M.Uzam, MCZhou (2008), “Deadlock control of concurrent manufacturing processes sharing finite resources”, In J. Adv. manuf. Tech., Vol. 38, pp. 787-800.
9. 김정철 외 2 (2007), “Siphon특성을 이용한 FMS의 Deadlock 해석과 제어”, 제어자동시스템공학 논문지, 13권 (7호), pp. 677-682.
10. ZW Li,MC Zhou,MD Jeng (2008), “A Maximally Permissive deadlock prevention policy for FMS based on petri nets Siphon control and the theory of regions”, In: IEEE tr. on Automation Scie. and Eng., Vol. 5, No. 1, pp. 182-188.
11. 김종욱, 이종근 (2008), “자원공유플레이스 추이적행렬을 이용한 효율적인 교착상태 확인정책”, 한국시뮬레이션학회 논문지, 17권 (3호), pp. 75-83.
12. 김종욱, 정상운, 이종근 (2008), “추이적 행렬을 이용한 교착상태확인 알고리즘의 성능 분석”, ’08추계학술대회논문집, 한국시뮬레이션학회, 서울산업대, pp. 98-102.

김 종 욱 (jwkim@changwon.ac.kr)

1992 경상대학교 전자계산학과 학사
2002 창원대학교 컴퓨터공학과 석사
2006 창원대학교 컴퓨터공학과 박사 수료
1992～현재 삼일 정보 기술(주) 대표이사

관심분야 : 패트리 넷, 성능분석, 정보보호, 스케줄링 연구

이 종 근 (jklee@cwnu.ac.kr)

1974 숭실대학교 전자계산학과 학사
1978 고려대학교 경영학과 경영학석사
1987 숭실대학교 컴퓨터공학과 공학 석사
2002 Ecole Centrale Paris 컴퓨터공학 공학박사
1987～1990 LSI / Univ. de Montpellier II 연구원 역임
1983～현재 창원대학교 컴퓨터공학과 교수

관심분야 : 패트리 넷, FMS 스케줄링 분석, 성능분석, 정보보호 관련 연구