제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

스케줄링 분석 문제의 복잡도 :추이적 행렬을 이용한 분할 알고리즘

이주현°, 송유진, 이종근
창원대학교 컴퓨터공학과
e-mail : henah@cosmos.changwon.ac.kr

Study on the Reduction of Complexity in Scheduling Analysis Problem: A slice algorithm using the Transitive matrix

Ju-Hyun Lee*, Yu-Jin Song*, Jong-Kun Lee*
*Dept. of Computer Engineering, Changwon University

요 약

유연생산 시스템에서 스케줄링 문제는 기본적으로 조합 최적화 문제로서 NP-hard problem 으로 알려져 있으며 문제의 크기에 따라 복잡도가 지수적으로 증가하게 된다. 이러한 복잡도를 줄이기 위해 우리는 넷의 행위적인 속성에 따른 제어 흐름들의 집합을 병행적 기본 단위(BUC)로 정의하고, 이들을 종합화 함으로 최종적인 스케줄링을 얻게 된다. 본 연구에서는 이러한 병행적 기본 단위로 넷을 분할하여 스케줄링을 분석하는 알고리즘을 제안한다.

1. 서론

유연 생산 시스템을 비롯한 생산 및 제조 시스템에서 주요한 과제는 최적화된 작업 환경을 구축하여 경제성을 추구하도록 스케줄링을 하는 것이다.

이러한 스케줄링 문제는 주로 시뮬레이션[1]이나 선형계획법[2], 대기이론[3]을 이용하여 해결한다. 특히 이러한 스케줄링 문제의 연구를 위하여 패트리넷을 이용한 스케줄링 분석 알고리즘 연구가 활발하다[4-7]. 이 연구에서는 패트리넷 모델의 추이적 행렬식을 이용해 자원을 공유하는 기기를 중심으로 여러 개의 서브넷으로 분리하여서 각각의 서브넷을 분석하여 스케줄링을 얻은 후 이를 종합하여, 최종적인 스케줄링을 구하는 분석 알고리즘을 제시 한다.

2. 타임 패트리넷과 추이적 행렬

2.1 타임 패트리넷

[정의 2.1] 타임 패트리넷(TPN) [8]

$TPN = (P, T, E, S, M_0, \tau)$

여기에서,

$P = \{p_1, p_2, \ldots\ldots, p_n\}$: 플레이스의 유한 집합 $(n \geq 0)$

$T = \{t_1, t_2, \ldots\ldots, t_m\}$: 트랜지션의 유한 집합 $(m \geq 0)$

$P \cap T = \phi$

$E: P \times T \rightarrow N$, E는 입력 함수

$S: T \times P \rightarrow N$, S는 출력 함수

$P \neq 0$ *and* $T \neq 0$

$M_0 \in M = \{M | M: P \rightarrow N\}$, M_0는 초기 토큰 상태

τ는 시간 함수 : $\tau \rightarrow ((f), f$는 실수, $f \geq 0)$

N: 음수를 제외한 정수의 집합(양의 정수의 집합)

2.2 추이적 행렬 [9]

[정의 2.2] 표식화 플레이스 기반 추이적 행렬

L_{BP}는 표식화 플레이스 기반 추이적 행렬이라고 하며 다음과 같이 정의된다.

$L_{BP} = B^{-} diag(t_1, t_2, t_3, \ldots\ldots, t_n)(B^{+})^{T}$ .....(식 2.1)

[정의 2.3] 가중적 플레이스 기반 추이적 행렬

L_{BP}^{*}는 $m \times m$ 가중적 플레이스 기반 추이적 행렬이라고 정의한다. 만약 트랜지션 t_k가 L_{BP}의 같은 열에 s 번 나타난다면 L_{BP}에 있는 t_k를 L_{BP}^{*}에서는 t_k / s로 표시한다.

2.3 병행적 기본 단위(BUC)

병행적 기본 단위는 모델에 내재한 구조적인 병행성을 기반으로 독립적으로 수행될 수 있는 제어 흐름들을 말한다.[10]

[정의 2.4] 병행적 기본 단위로 분할

병행적 기본단위 (BUC)로 분할하기 위해서는 해당 플레이스의 행 방향에 있는 모든 트랜지션에 대해서, 다음의 조건을 만족하여야 한다.

$\Sigma(t_{k_i}/d_i) \geq 1$ .....(식 2.2)

여기서, t_{k_i} 는 행 방향에 있는 같은 이름을 가진 트랜지션을 지칭하며, d_i 는 같은 트랜지션의 동일한 분모 값을 나타낸다.

3. 추이적 행렬을 이용한 BUC 단위의 분할 알고리즘

3.1 BUC 단위의 서브넷 생성 알고리즘

```
char Psubnet[ ], Tsubnet[ ]; /* 트랜지션과 플레이스의 저장배열 */
void search() {
   for (플레이스의 행 테이블에 트랜지션이 존재) {
      트랜지션을 읽어 온다;
      while (같은 행 테이블에 트랜지션이 존재) {
         Tsubnet[ ] 배열에 트랜지션을 넣는다;
          if (Psubnet[ ]에 그 트랜지션의 행방향의 플레이스가
              존재하지 않는다면) {
              Psubnet[ ] 배열에 그 트랜지션의 행방향의
              플레이스를 넣는다.;
         } /* if */
         행 테이블의 다른 트랜지션을 읽어 온다.;
      } /* while */
      다음 행 테이블로 옮긴다.;
   } /* for */
   for (Psubnet[ ] 배열에 플레이스가 존재) {
       Psubnet[ ]에 있는 플레이스의 행방향의 트랜지션을 읽어
       온다.;
      Tsubnet[ ] 배열에 트랜지션을 넣는다.;
   } /* for */
} /* search() */
main()
{
   열의 플레이스와 테이블의 값을 읽어 온다.;
   for (테이블의 열에 플레이스가 존재) {
      열의 플레이스를 읽어 온다.;
      if(열의 플레이스가 초기 마킹 플레이스) {
          Psubnet[ ] 배열에 그 플레이스를 넣는다.;
          search();
      } /* if */
      if(Σ t_k/d ≠ 1) {
      /*  t_k  Tsubnet[ ] 배열에 있는 트랜지션이다. */
      트랜지션(t_k )의 행의 플레이스를 읽어 온다.;
      search();
      } /* if */
      printf (Psubnet[ ], Tsubnet[ ]);
   } /* for */
   } /* main */
```

3.2 분할 알고리즘의 검증

[정리 3.1] 통합

$\{\forall P_i \in P, \forall T_i \in T, \forall I_i \in I, \forall O_i \in O, \forall M_i \in M\}$ 에 대하여

$BUC_1 = \{P_1, T_1, I_1, O_1, M_1\}$, $BUC_2 = \{P_2, T_2, I_2, O_2, M_2\}$,.....

....., $BUC_n = \{P_n, T_n, I_n, O_n, M_n\}$ 일 경우,

$BUC_1 \cup BUC_2 \cup ..... \cup BUC_n =$

$((P_1 \cup .... \cup P_n),(T_1 \cup .... \cup T_n),(I_1 \cup .... \cup I_n),(Q \cup .... \cup O_n),(M_1 \cup .... \cup M_n))$

$\bigcup_{i=1}^{n} BUC_i = N(P,T,I,O,M)$ 이다.

[증명] $N = \{P,T,I,O,M\}$ 을 패트리넷이라 하고, N 을 분할 알고리즘에 의해 분할하여 얻은 서브넷 $BUC_i = \{P_i, T_i, I_i, O_i, M_i\}$ $(i = 1,.....,n)$ 이라 하면, 분할된 서브넷의 $\bigcup BUC_i$ 는 분할 알고리즘의 정의에 의하여 $N \subseteq \bigcup_{i=1}^{n} BUC_i$ 이고, 또한 $\bigcup_{i=1}^{n} BUC_i \subseteq N$ 이면 상등의 법칙에 의해 $N = \bigcup_{i=1}^{n} BUC_i$ 이다.

(1) $N \subseteq \bigcup_{i=1}^{n} BUC_i$ 의 경우

$\bigcup_{i=1}^{n} BUC_i$ 은 $N \subseteq \bigcup_{i=1}^{n} BUC_i$ 이다.

(2) $\bigcup_{i=1}^{n} BUC_i \subseteq N$ 의 경우

만일 BUC_i 에서, $p_i \in P_i$, $t_i \in T_i$ 이면, $\exists p_i \in \bigcup_{i=1}^{n} P_i$, $\exists t_i \in \bigcup_{i=1}^{n} T_i$ 이며, $p_i \in {}^{\bullet}t_i$ 이면, $p_i \in \bigcup_{i=1}^{n} ({}^{\bullet}T_i)$ 이다.

분할 알고리즘에 의해 얻은 BUC_i 는 모든 플레이스와 트랜지션의 삭제나 추가가 없으므로, 각 BUC_i 의 플레이스의 통합과 트랜지션의 통합은

$\bigcup_{i=1}^{n} P_i \subseteq P$, $\bigcup_{i=1}^{n} T_i \subseteq T$ 이므로, $p_i \in \bigcup_{i=1}^{n} ({}^{\bullet}T_i) \in {}^{\bullet}T$ 이다.

따라서, $p_i \in {}^{\bullet}t_i$ 이면, $p_i \in {}^{\bullet}t \in T^{\bullet}$ 이고,

$p_i \in t_i{}^{\bullet}$ 이면, $p_i \in t^{\bullet} \in T$ 이다.

그러므로, $\bigcup_{i=1}^{n} BUC_i \subseteq N$ 이다.

이를 종합하면,

$N \subseteq \bigcup_{i=1}^{n} BUC_i$, $\bigcup_{i=1}^{n} BUC_i \subseteq N$ 이므로 $N = \bigcup_{i=1}^{n} BUC_i$ 이다.

[정리 3.2] 통합된 넷의 특성

패트리넷 N 에서 분할 알고리즘으로 얻은 $\bigcup_{i=1}^{n} BUC_i$ 는 N 이 가지고 있는 성질(즉, 도달 가능성, 생동성, 유한성 등)과 일치한다.

[증명] 정리 3.1 에 의하여 성질에 변동이 없다.

4. 분할 알고리즘을 이용한 스케줄링 알고리즘

서브넷 1 과 서브넷 2 가 동시 수행되는 서브넷이라고 할 때, 각 서브넷들의 플레이스를 다음과 같이 정의한다. :

서브넷 1 의 입력 플레이스 : $In_{sub1}[i]$

서브넷 2 의 입력 플레이스 : $In_{sub2}[i]$

서브넷 1 의 출력 플레이스 : $Out_{sub1}[i]$

서브넷 2 의 출력 플레이스 : $Out_{sub2}[i]$

먼저 비효율적인 스케줄들을 제거하는 과정이 필요하다. 다음과 같은 경우는 고려 대상에서 제외시킨다.

[조건 4.1] $In_{sub1}[i][0] = In_{sub2}[j][0]$

[조건 4.2]

```
For (i=0; i++; i<n)   {
For (j=0; j++; j<m)   {
 If( In_sub1[i]!= Out_sub1[j] )
 {
   for(k=0; k++; k<l) {
    if ( In_sub1[i] == In_sub2[k] )
       ErrorMsg("동시 수행이 가능하지 않다.");
} } } }
```

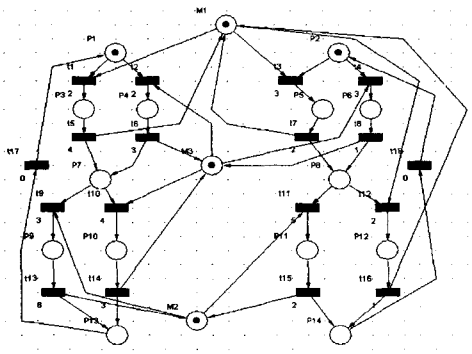
[조건 4.3] $Out_{sub1}[i][n-1] = In_{sub2}[j][0]$

[조건 4.4] 한 서브넷의 출력 플레이스와 다른 서브넷의 입력 플레이스가 같은 경우, 출력 플레이스가 있는 테이블이 우선한다.

위의 조건 4.1, 조건 4.2, 조건 4.3, 조건 4.4 들을 적용하여 스케줄링을 찾아내는 알고리즘을 작성하면 다음과 같다.

```
패트리넷 모델의 추이적 행렬식 L_BP* 를 구한다.
L_BP* 를 이용한 분할 알고리즘을 적용하여 모델을 분할한다.
분할된 서브넷의 처리 순서를 정한다.
분할된 서브넷 각각의 sequence 테이블을 만든다.
동시 수행하는 서브넷들을 추출한다.
do {
for (i=0; i++; i<n)
{
for(j=0; j++; j<m)
{
    for (k=0; k++; k<h)
    {
      배열 subnet1[i][j]와 배열 subnet2[j][k]를 비교한다.;
      If (조건[4.1][4.2][4.3]에 부합하면) {
        해당 서브넷 테이블들을 제외한다.;
      }
    }
}
서브넷 테이블(subnet1[i], subnet2[j])를 얻는다.
}while(동시 수행하는 서브넷 조합들이 있는 동안)
서브넷 조합들의 스케줄링 시간을 계산하여 테이블로 작성한다.
동시 수행이 아닌 서브넷들을 테이블의 해당 위치에 끼워넣는다
스케줄링을 얻는다.
```

5. 사례 연구

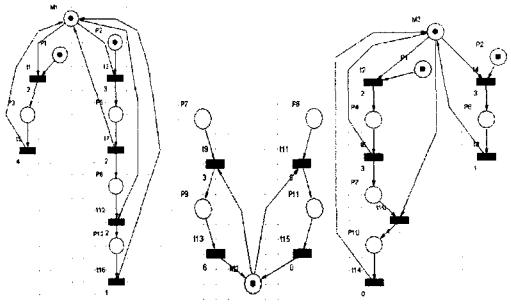


<그림 5.1> FMS 의 패트리넷 모델

<표 5.1> 그림 5.1 의 L_{BP}^{*}

$$L_{BP}^{*} = \begin{bmatrix}
0 & 0 & t1 & t2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & 2 & 2 & & & & & & & & & & & & \\
0 & 0 & 0 & 0 & t3 & t4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & 2 & 2 & & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & t5 & 0 & 0 & 0 & 0 & 0 & 0 & t5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & t6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & t7 & 0 & 0 & 0 & 0 & 0 & t7 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & t8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t9 & t10 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & & 2 & 2 & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t11 & t12 & 0 & 0 & 0 & 0 \\
 & & & & & & & & & & 2 & 2 & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t13 & 0 & 0 & t13 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t14 & 0 & 0 & 0 & t14 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t15 & 0 & t15 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t16 & t16 & 0 & 0 \\
t17 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & t18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & t1 & 0 & t3 & 0 & 0 & 0 & 0 & 0 & 0 & t12 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & 2 & & 2 & & & & & & & 2 & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t9 & 0 & t11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & & 2 & & 2 & & & & & & & \\
0 & 0 & 0 & t2 & 0 & t4 & 0 & 0 & 0 & t10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

FMS 의 패트리넷 모델은 그림 5.1 과 같으며, 이를 추이적 행렬로 나타내면 표 5.1 과 같다. 이 추이적 행렬을 이용하여 앞 장에서 제안한 분할 알고리즘을 적용하면 그림 5.2 와 같이 세 개의 서브넷으로 분할되며, 이들의 구성은 표 5.2 에 있다.



(a) 서브넷 1 (b) 서브넷 2 (c) 서브넷 3

<그림 5.2> 그림 5.1 의 서브넷

<표 5.2> 그림 5.1 의 서브넷의 구성

서브넷	플레이스와 트랜지션
서브넷 1	M1-P1-P2-P3-P5-P8-P12, t1-t3-t5-t7-t12-t16
서브넷 2	M2-P7-P8-P9-P11, t9-t11-t13-t15
서브넷 3	M3-P1-P2-P4-P6-P7-P10, t2-t4-t6-t8-t10-t14

5.1 서브넷의 스케줄링 테이블

5.1.1 "서브넷 1(sub1)" 스케줄링 테이블

<표 5.3> 그림 5.2(a) 스케줄링 테이블 (1)

Sub1[0]	Sub1 [0][0]	Sub1 [0][1]	Sub1 [0][2]	Sub1 [0][3]	Sub1 [0][4]	Sub1 [0][5]
입력 플레이스	$In_{sub1}[0][0]$ =P1		$In_{sub1}[0][2]$ =P2		$In_{sub1}[0][4]$ =P8	
점화 순서	T1(2)	T5(4)	T3(3)	T7(2)	T12(2)	T16(1)
출력 플레이스		$In_{sub1}[0][1]$ =P7		$Out_{sub1}[0][3]$ =P8		$Out_{sub1}[0][5]$ =P2

Subn[i][j]

여기서 :

i : Subn 의 스케줄링 테이블의 경우의 수.

j : Subn 의 스케줄링 테이블 내에서의 구분자.

$In_{subn}[i][j]$: Subn 의 입력 플레이스 배열.

$Out_{subn}[i][j]$: Subn 의 출력 플레이스 배열

<표 5.4> 그림 5.2(a) 스케줄링 테이블 (2)

Sub1[1]	Sub1 [1][0]	Sub1 [1][1]	Sub1 [1][2]	Sub1 [1][3]	Sub1 [1][4]	Sub1 [1][5]
입력플레이스	P2		P8		P1	
점화 순서	T3(3)	T7(2)	T12(2)	T16(1)	T1(2)	T5(4)
출력플레이스		P8		P2		P7

<표 5.5> 그림 5.2(a) 스케줄링 테이블 (3)

Sub1[2]	Sub1 [2][0]	Sub1 [2][1]	Sub1 [2][2]	Sub1 [2][3]	Sub1 [2][4]	Sub1 [2][5]
입력플레이스	P2		P1		P8	
점화 순서	T3(3)	T7(2)	T1(2)	T5(4)	T12(2)	T16(1)
출력플레이스		P8		P7		P2

5.1.2 "서브넷 2(sub2)" 스케줄링 테이블

<표 5.6> 그림 5.2(b) 스케줄링 테이블 (1)

Sub2[0]	Sub2[0][0]	Sub2[0][1]	Sub2[0][2]	Sub2[0][3]
입력 플레이스	P7		P8	
점화 순서	T9(3)	T13(6)	T11(5)	T15(2)
출력 플레이스				P2

<표 5.7> 그림 5.2(b) 스케줄링 테이블 (2)

Sub2[1]	Sub2[1][0]	Sub2[1][1]	Sub2[1][2]	Sub2[1][3]
입력 플레이스	P8		P7	
점화 순서	T11(5)	T15(2)	T9(3)	T13(6)
출력 플레이스				P1

5.1.3 "서브넷 3(sub3)" 스케줄링 테이블

<표 5.8> 그림 5.2(c) 스케줄링 테이블 (1)

Sub3[0]	Sub3 [0][0]	Sub3 [0][1]	Sub3 [0][2]	Sub3 [0][3]	Sub3 [0][4]	Sub3 [0][5]
입력플레이스	P1		P7		P2	
점화 순서	T2(2)	T6(3)	T10(4)	T14(3)	T4(3)	T8(1)
출력플레이스		P7		P1		P8

<표 5.9> 그림 5.2(c) 스케줄링 테이블 (2)

Sub3[1]	Sub3 [1][0]	Sub3 [1][1]	Sub3 [1][2]	Sub3 [1][3]	Sub3 [1][4]	Sub3 [1][5]
입력플레이스	P1		P2		P7	
점화 순서	T2(2)	T6(3)	T4(3)	T8(1)	T10(4)	T14(3)
출력플레이스		P7		P8		P1

<표 5.10> 그림 5.2(c) 스케줄링 테이블 (3)

Sub3[2]	Sub3 [2][0]	Sub3 [2][1]	Sub3 [2][2]	Sub3 [2][3]	Sub3 [2][4]	Sub3 [2][5]
입력플레이스	P2		P1		P7	
점화 순서	T4(3)	T8(1)	T2(2)	T6(3)	T10(4)	T14(3)
출력플레이스		P8		P7		P1

5.2 서브넷의 스케줄링 처리 과정

분할된 각 서브넷들에 대하여 아래 알고리즘을 수행한다.

단계 1) Sub1 과 Sub3 이 동시에 처음 실행된다.
단계 2) 따라서 Sub1 과 Sub3 의 조합을 살펴본다.
단계 3) 알고리즘에 의하여 동시 수행이 아닌 조합은 제거한다.
단계 4) Sub1 과 Sub3 의 조합에서 동시 수행할 수 없는 조합을 제거하면 다음의 조합만을 고려한다.
- Sub1[1]과 Sub3[0]
- Sub1[1]과 Sub3[1]

단계 5) 위의 조합에서 동시 수행 가능한 부분만을 추출하여 계산한다.
단계 6) Sub2 를 고려한다.
단계 7) 넷의 전체 스케줄링에서 가장 최적의 스케줄링 시간 값을 계산한다.
단계 8) 전체의 스케줄링 흐름을 나타낸다.

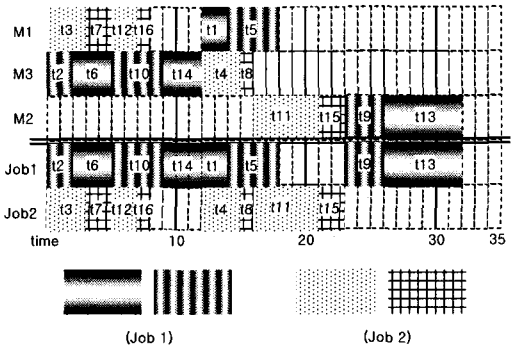
<표 5.11> 그림 5.1 의 스케줄링 결과

Sub1 [1][0]	Sub1 [1][1]	Sub1 [1][2]	Sub1 [1][3]	Sub1 [1][4]	Sub1 [1][5]				
T3(3)	T7(2)	T12(2)	T16(1)	T1(2)	T5(4)				
Sub3 [0][0]	Sub3 [0][1]	Sub3 [0][2]	Sub3 [0][3]	Sub3 [0][4]	Sub3 [0][5]	Sub2 [1][0]	Sub2 [1][1]	Sub2 [1][2]	Sub2 [1][3]
T2(2)	T6(3)	T10(4)	T14(3)	T4(3)	T8(1)	T11(5)	T15(2)	T9(3)	T13(6)

6. 결론

분할 스케줄링 알고리즘은 모델을 분할하여 좀 더 작은 모델들을 가지고 분석을 시도하고, 또한 비효율적인 스케줄들을 미리 제거함으로써, 문제의 크기에 따라 복잡도가 지수적으로 증가하게 되는 NP-hard problem 을 보완하고자 시도한 알고리즘이며, 이 분석 방법은 정확한 스케줄을 찾아냄과 동시에 많은 경우의 수를 미리 제거함으로써 복잡도를 줄일 수 있는 효과를 가져온다.

그러나 이 논문에서 제안한 알고리즘은 모델의 마킹 상태가 각 플레이스에서 하나 이하인 경우를 고려하여 작성한 것이다. 또한 배열을 이용함으로 해서 메모리 사용의 효율이 떨어지는 단점이 있다. 따라서, 메모리 상의 단점을 보완할 수 있는 알고리즘의 개발과 함께 마킹의 상태가 하나 이상인 경우도 고려할 수 있는 알고리즘의 개발이 필요하다.



<그림 5.3> 스케줄링을 표현한 간트 챠트

참고문헌

[1] Y.L. Chang and R.S. Sullivan, Using SLAM to Design the Material Handling System of a Flexible Manufacturing System, International Journal of Production Research, Vol.24, pp.15-26, 1986.

[2] Y.U. Chen and R.G. Askin, A Multiobjective Evaluation of Flexible Manufacturing System Loading Heuristics, International Journal of Production Research, Vol.28, pp.895-911, 1990.

[3] Y. Cai, T. Sekiguchi, H. Tanaka, M. Hikichi and Y. Maruyama, A method for analyzing Petri net structure and adding counter-place to its incidence matrix, The Transactions of the SICE of Japan, Vol.29, No. 12, pp. 1458-1464, 1993.

[4] H.Hillion,J-M Proth, and X-L Xie, A Heuristic Algorithm for Scheduling and Sequence Job-Shop problem, In: proceeding 26th CDC,pp.612-617, 1987.

[5] J.K. Lee, O. Korbaa, and J.C. Gentina, Modeling and analysis of Cycle scheduling using Petri nets unfolding, In: proceeding IEEE SMC'01, pp.2611-2616, 2001.

[6] J.K. Lee, Une méthode d'analyse d'ordinnancement des systems fléxibles de production manufacturière utilisant le dépliage des réseaux de Petri, Thèse de doctorat EC Paris, mars 2002.

[7] O. Korbaa, H. Camus and J-C. Gentina, FMS Cyclic Scheduling with Overlapping production cycles, In: proceeding ICATPN'97, pp.35-52, 1997.

[8] J. Carlier, P. Chretienne and C. Girault, Modeling Scheduling Problems with Timed Petri Nets, in Advanced in Petri Nets 1984, Lecture Notes in Computer Science, G. Goes and J.Hartmanis(eds), Springer-Verlag Pub.Co., pp. 62-82, 1985.

[9] J. Liu, Y. Itoh, I. Miyazawa and T. Sekiguchi, A Research on Petri Net Properties using Transitive Matrix, IEEE International Conference on Systems, Man, and Cybernetics, 1999.

[10] 송유진, 김종욱, 이종근, 유연생산 시스템 스케줄링 분석을 위한 추이적 행렬을 이용한 페트리 넷의 분할, 제어 자동화 시스템공학회, 제 8 권 제 4 호, pp.292-298, 2002. 4.