90 대한기계학회논문집 A권 제22권 제1호, pp. 90~101, 1998

공구셋업시간을 고려한 유연생산시스템의 스케쥴링

임 성 진* · 이 두 용*

(1997년 4월 22일 접수)

Scheduling of Flexible Manufacturing Systems with the Consideration of Tool Set-up Times

Seong Jin Yim and Doo Yong Lee

Key Words: Flexible Manufacturing Systems(FMS : 유연생산시스템), Tool Set-up(공구셋업), Petri Nets(페트리넷), Heuristic Search(경험적 탐색)

Abstract

This paper presents a scheduling method that uses Petri net modeling and heuristic search to handle the tool setup. In manufacturing systems, a tool is attached to a particular machine to process a particular operation. The activity to attach a tool to a particular machine and detach the tool from the machine requires time. The processing time of operations varies according to the attached tool and the machine used. The method proposed in this paper uses Petri net to model these characteristics and applies a search algorithm to the reachability graph of the Petri net model to generate an optimal or near-optimal schedule. New heuristic functions are developed for efficient search. The experimental results that show the effectiveness of the proposed method are presented.

1. 서 론

급변하는 시장의 상황에 유연하게 대처하기 위해 생산시스템은 이전의 소품종 대량생산방식에서 다품종 소량생산방식으로 변화하고 있다. 컨베이어를 이용하는 대량생산시스템의 경우 새로운 제품을 생산하기 위해 시스템의 구성을 바꾸려면 매우 많은 시간과 비용이 소요되므로 급변하는 시장의 수요에 대응하기 어렵다. 따라서 시장의 수요에 민첩하게 대응하기 위해서는 새로운 제품에 대해 시스템의 구성이 쉽게 변경이 가능해야 한다. 이를 위해서는 여러 가지 요구사항이 만족되어야 한다. 가장 먼저 하나의 기계가 여러 가지 부품을 처리할 수 있어야 한다. 그리고 각 기계 사이에서 부품의 운반은 유연성이 떨어지는 컨베이어나 크레인 대신에 자율유도차량(automated guided vehicles)과 같은 자동화된 시스템에 의해 이루어져야 한다. 유연생산시스템(Flexible Manufacturing System : FMS)은 이러한 요구를 만족시키기 위해 개발되었다.

FMS에서 수치제어 공작기계는 하나 이상의 작업을 수행할 수 있어서 기존의 job-shop과는 달리 주어진 부품(job or part)이 하나 이상의 기계에서 가공될 수 있다. 따라서 부품이 여러 개의 작업경로(routing)를 가지게 한다. 이러한 다양한 작업경로 속에서 스케쥴링은 선택된 목적함수를 최적화시키고 모든 job을 완수하기 위해 어떤 부품의 어떤 작업을 어떤 기계에서 언제 수행할 것인가를 결정하는 것이다. 스케쥴링 문제는 기본적으로 조합최적화 문제로서 NP-hard 문제로 알려져 있다.

특정기계에서 작업을 수행하기 위해서는 해당기

*회원, 한국과학기술원 기계공학과

계는 수행하는 작업에 맞추어 미리 셋업이 되어 있어야 한다. 셋업은 특정의 작업을 위한 기계에서의 준비작업으로서 공구를 교환하거나 새로운 작업을 위한 프로그램을 입력하는 등의 활동을 말하며 특정한 양의 시간을 요구한다. 보통 셋업시간이 작업을 수행하는 시간보다 극히 작은 경우 셋업시간을 무시하고 스케쥴링을 하게 된다. 하지만 셋업시간이 작업수행시간과 같거나 큰 경우 셋업시간을 무시하면 큰 오류가 된다. 따라서 셋업시간이 큰 경우 셋업시간을 고려한 스케쥴링방법이 필요하게 된다. 그리고 셋업문제를 다룰 때 고려해야 하는 또 다른 사항은 같은 셋업을 필요로 하는 작업이 같은 기계에 연속적으로 들어오는 경우에는 셋업이 필요없다는 점이다.

이 논문에서는 셋업을 기계에 공구를 장착하고 탈착하는 과정이라고 가정하기로 한다. 이 과정은 각 공구와 각 기계에 따라 각기 다른 시간이 소요된다. 어떤 기계에 어떤 공구가 장착되고 탈착되는가에 따라서 해당기계에서 수행되는 작업은 작업수행시간이 달라진다. 특정기계에 공구가 이미 장착되어 있는 상황에서 다른 공구를 요구하는 작업이 들어오는 경우 이미 기계에 있는 공구는 탈착되고 필요한 공구가 기계에 장착되어야 한다. 또 다른 경우 이미 장착된 공구와 같은 공구를 요구하는 작업이 들어오는 경우 공구를 탈착할 필요는 없게 된다. 이 문제는 고려해야 할 제한조건이 너무 많아서 기존의 스케쥴링문제보다 훨씬 더 복잡하게 된다.

페트리넷(Petri nets)은 비동기적이며, 동시발생적인 이벤트(event)에 의해 시스템의 상태가 변화하는 이산이벤트시스템(Discrete Event System : DES)을 모델링하는데 아주 적절한 도구이다. 페트리넷은 FMS 환경에 고유한 유연한 작업경로, 자원공유, 공유된 자원간의 상호배제, 변화하는 로트크기와 같은 특징들을 모델링하는 데 적절한 도구가 된다. 일단 FMS를 페트리넷을 이용하여 모델링하면 페트리넷의 도달가능그래프(reachability graph)는 시스템의 전체상태를 표현하게 되며 트랜지션(transition)의 점화순서(firing sequence)는 그대로 FMS 운영에 있어서의 스케쥴이 된다. 시스템의 시간에 따른 변화를 보기 위해 보통 트랜지션(transition)이나 플레이스(place)에 시간을 부가한 timed Petri nets를 이용한다.[1~3]

일단 FMS를 페트리넷으로 모델링한 후 최적의 스케쥴을 구하기 위해 경험적 탐색(heuristic search)을 이용한다. 경험적 탐색은 그래프 탐색의 하나로 해를 탐색하는 과정에서 사전에 주어진 경험적인 정보를 이용하는 방법이다. 그리고 국소최적에 빠지지 않기 위해 과거에 지나간 경로를 기억하고 있다. 새로운 노드(node)를 확장할 때 확장하는 노드가 얼마나 유망한 노드인가를 예측하기 위해 평가함수(evaluation function)를 이용한다. 경험적 탐색은 예측된 평가함수가 실제 평가함수값보다 작다면 유한한 시간 내에 최적해를 찾는다는 것이 보장되어 있다.

Reddy et al.[4]은 페트리넷 모델을 이용하여 공구관리에 관한 주제를 연구하였다. Lee and DiCesare[5~7]는 FMS 스케쥴링 문제를 풀기 위해 timed-place Petri net과 경험적 탐색을 이용하였고 이 방법은 셋업을 고려한 스케쥴링 문제에도 적용되었다. 셋업시간을 고려하기 위해 Lee and DiCesare[7]의 논문에서는 플레이스에 시간을 부가한 timed-place Petri net(TPPN)이 이용되었다. TPPN을 이용하는 경우 모든 트랜지션은 시간이 없이 순간적으로 점화하는 것으로 간주되며, 특정의 place에 시간이 부가된다. TPPN으로 셋업시간을 모델링하는 경우 FMS 스케쥴링을 위한 기존의 페트리넷 모델에 부가적인 넷을 추가하여야 하며 결과적으로 넷이 복잡해진다. 이에 비해 timed-transition Petri net(TTPN)을 이용하면 특정 opration의 수행과 셋업시간을 간단하게 표현할 수 있게 된다. 따라서 이 논문에서는 TTPN을 이용하기로 한다.

이 논문에서는 공구셋업시간을 고려한 스케쥴링 문제를 풀기 위한 페트리넷 모델링과 경험적 탐색을 이용하는 방법을 제안한다. 공구셋업을 고려한 스케쥴링문제는 페트리넷을 이용하여 모델링한다. 페트리넷을 이용해 얻어진 모델의 도달가능그래프에 경험적 탐색 알고리듬을 적용하여 최적 또는 준최적의 스케쥴을 구한다. 최적화하려는 스케쥴링 목표는 모든 job을 처리하는데 소요되는 시간의 최소화, 즉 전체 작업시간(makespan)의 최소화이다. 경험적 탐색 알고리듬의 성능이 선택된 경험적 함수의 성능에 크게 의존하므로 알고리듬의 성능을 개선하기 위한 새로운 경험적 함수가 제시된다.

2. FMS 스케쥴링문제를 위한 페트리넷 모델링

2.1 스케쥴링을 위한 페트리넷 모델링

페트리넷은 시스템 내에서 발생하는 이벤트들이 비동기성과 동시성, 선행조건을 가지는 DES에서 이벤트의 제어와 정보의 흐름을 모델링 하는 도구이다. 페트리넷을 이용하면 조건을 나타내는 플레이스(place)와 조건의 변화를 나타내는 트랜지션(transition), 그리고 플레이스 안에 위치한 점으로서 플레이스가 나타내는 조건의 유효성을 나타내는 토큰(token)을 이용하여 DES를 아주 적절히 묘사할 수 있다.[8] 페트리넷의 마킹(marking)은 모델링된 시스템의 상태를 나타내며 이러한 상태는 트랜지션의 점화를 통해 변화하므로 트랜지션의 점화순서를 제어하면 원하는 상태와 시스템 거동을 얻을 수 있다.

Fig. 1의 (a)는 한 대의 기계로 두 가지 작업을 처리하는 과정을 페트리넷을 이용하여 모델링한 것을 보여 준다. Fig. 1에서 플레이스 p1, p3, p5는 각각 job J_1의 가공대기, 가공, 가공완료를 나타내며 트랜지션 t1, t3는 각각 job의 가공시작과 가공완료를 나타낸다. 플레이스 p2, p4, p6과 트랜지션 t2, t4는 job J_2의 경우로서 물리적인 의미는 job J_1의 경우와 같다. 플레이스 p7은 J_1과 J_2를 처리할 수 있는 기계의 이용가능성을 나타내며, 이 기계는 각 job에 공유된 자원(shared resource)이다. 각 플레이스에 있는 토큰은 해당 플레이스가 나타내는 조건이 유효하다는 것을 나타낸다. 즉 플레이스 p1과 p2에 있는 토큰은 J_1과 J_2가 가공 대기중이라는 것을 나타내며 플레이스 p7에 있는 토큰은 기계가 이용가능하다는 것을 나타낸다. 이러한 상황에서 하나의 job이 기계를 이용하는 경우 기계가 이 job의 가공을 마칠 때까지 다른 job은 이 기계를 이용하지 못한다.

Fig. 1의 (b)는 (a)의 페트리넷 모델의 도달가능 그래프를 나타낸다. Fig. 1의 (b)의 벡터들은 각 플레이스에 토큰이 얼마인지를 나타내는 마킹이고 아크에 부가된 ti는 트랜지션 i가 점화하는 것을 나타낸다. 예를 들어 [1, 1, 0, 0, 0, 0, 0, 1]에서 트랜지션 t1이 점화하면 [0, 1, 1, 0, 0, 0, 0]이 된다. 이렇듯 페트리넷의 도달가능그래프는 마킹과 트랜지션의 점화에 따른 마킹의 변화를 통해 페트리넷이

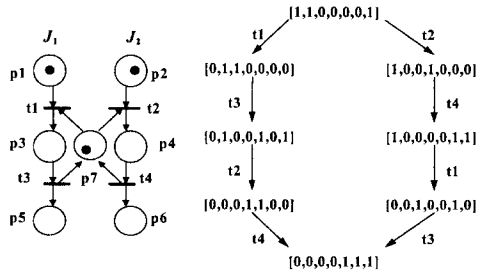

Fig. 1 Petri net modeling of shared resources.

모델링하는 시스템의 전체상태를 표현하게 된다.

Timed Petri net(TPN)은 시스템의 조건을 의미하는 플레이스나 사건을 의미하는 트랜지션에 실제 시스템에 대응하는 시간을 부가해서 시간을 통한 성능해석이 가능하게 한 것이다. Timed Petri net에는 트랜지션에 시간을 부가하는 timed-transition Petri net(TTPN)과 플레이스에 시간을 부가하는 timed-place Petri net(TPPN)이 있다. TPPN의 경우 시스템의 시간진행을 추적할 때 임의의 시점에서 마킹이 확실하며 시간의 진행을 추적하기 위해서는 토큰을 가진 플레이스만 고려하면 된다.[5, 6] 하지만 서론에서도 언급했듯이 동일한 대상을 TPPN과 TTPN으로 모델링할 때 TPPN이 TTPN보다 더 복잡하게 된다. 즉, TPPN의 마킹이 많아지게 되어 탐색으로 해를 찾는데 TTPN보다 더 많은 시간이 소요된다. 따라서 이 논문에서는 TTPN을 이용하기로 한다.

예제 1(Example 1) : FMS에서는 하나의 기계가 여러 가지 작업을 수행할 수 있기 때문에 부품의 작업경로가 고정되어 있지 않고 유연하다. Table 1에서 job은 두 개, 즉 J_1과 J_2이고 서로 독립적이다. 즉, 순서상 서로 관련이 없다. 각 job은 2개의 작업(operation)을 순서대로 거쳐야 하며 각 작업은 기계 M_1, M_2, M_3에서 수행 가능하다. 괄호 안에 있는 숫자는 각 기계에서 작업의 수행시간을 나타내며 미리 주어지게 된다. 예를 들어 job J_1의 첫번째 작업을 수행하는 데에는 기계 M_1에서 3단위시간, 또는 기계 M_3에서 5단위시간이 소요된다. 그리고 작업들은 특정기계를 공유하고 있어서 하나의 작업이 특정기계를 이용하면 이 기계를 이용하는 작업들은 이 기계가 이용 가능할 때까지 대기하거나 다른 기계를 이용해야 한다. 예를 들어 Table 1에서 J_1의 1st operation이 기계 M_1을 이용

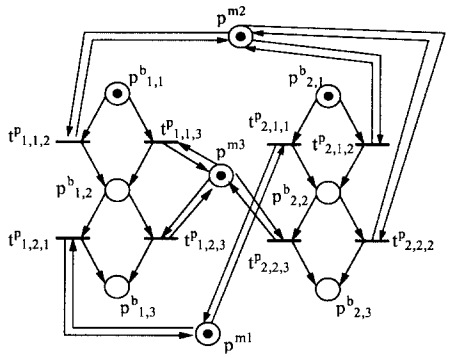

Fig. 2 The Petri net model of Example 1.

하는 경우 J_2의 1st operation은 기계 M_1을 이용하지 못하며 기계 M_3를 이용해야 한다. Table 1에서 주어진 조건은 페트리넷으로 모델링하면 Fig. 2와 같이 된다.

Fig. 2에서 플레이스 p^{mi}는 기계 i의 이용가능성을 나타낸다. 플레이스 $p^{b}_{i,j}$는 job i의 j번째 버퍼 플레이스(buffer place)를 나타낸다. 그리고 플레이스 $p^{b}_{i,1}$에 있는 토큰의 개수는 job i의 로트크기를 나타낸다. 즉, 이 FMS에서 각 job마다 처리해야 할 수량이 1이다. 트랜지션 $t^{p}_{i,j,k}$는 job i의 j번째 operation이 기계 k에서 수행되는 것을 나타낸다. Fig. 2에서 플레이스 $p^{b}_{1,j}$는(j=1, 2, 3)와 트랜지션 $t^{p}_{1,j,k}$(j=1, 2 ; k=1, 2, 3)은 job J_1의 job 모듈(module)이고, 플레이스 $p^{b}_{2,j}$는(j=1, 2, 3)와 트랜지션 $t^{p}_{2,j,k}$(j=1, 2 ; k=1, 2, 3)은 job J_2의 job 모듈(module)이다. job J_1의 job 모듈(module)에서 플레이스 $p^{b}_{1,3}$은 job J_1의 목표플레이스(goal place)이고, job J_2의 job 모듈(module)에서 플레이스 $p^{b}_{2,3}$은 job J_2의 목표플레이스이다. 플레이스 $p^{b}_{1,1}$에 있던 토큰이 플레이스 $p^{b}_{1,3}$까지 가면 job J_1은 가공이 완료된 것이며, 플레이스 $p^{b}_{2,1}$에 있던 토큰이 플레이스 $p^{b}_{2,3}$까지 가면 job J_2는 가공이 완료된 것이다. 따라서 목표마킹은 플레이스 $p^{b}_{1,3}$과 $p^{b}_{2,3}$에 토큰이 한 개씩 있는 경우이다. Table 2는 Fig. 2에서 각 트랜지션에 할당된 시간을 보여 준다.

Fig. 2와 같은 페트리넷의 도달가능그래프는 Table 1에서 주어진 FMS가 가질 수 있는 상태들을 표현한다. 스케줄은 트랜지션의 점화순서로 주어지게 된다. 스케줄을 구하는 과정은 모델링된 페트리넷의 도달가능그래프에서 탐색을 통해 주어진 목적함수를 최적으로 하는 트랜지션의 점화순서를 구하는 것이다. 페트리넷의 도달가능그래프를 탐색하기 위

Table 1 Job requirements of Example 1.

Job	J_1	J_2
1 st operation	M_1(3) / M_3(5)	M_1(8) / M_3(3)
2 nd operation	M_2(7) / M_3(4)	M_1(6) / M_2(2)

Table 2 The processing times associated with transitions.

Transition	Assigned time	Transition	Assigned time
$t^{p}_{1,1,1}$	3	$t^{p}_{2,1,1}$	8
$t^{p}_{1,1,3}$	5	$t^{p}_{2,1,3}$	3
$t^{p}_{1,2,2}$	7	$t^{p}_{2,2,1}$	6
$t^{p}_{1,2,3}$	4	$t^{p}_{2,2,2}$	2

해서는 도달가능그래프의 아크는 두 가지 정보를 가지고 있어야 한다. 즉, 점화하는 트랜지션과 점화가능시간의 정보를 가지고 있어야 탐색을 통해 최적값을 가지는 트랜지션의 점화순서를 구할 수 있게 된다.

Timed-transition을 점화하는 데에는 점화시간(firing time)과 점화가능시간(enabling time)을 이용하는 방법이 있다. 점화시간을 이용하는 경우 트랜지션은 점화할 때 입력 플레이스에서 토큰을 제거하고 트랜지션에 할당된 시간이 지난 후에 출력 플레이스로 토큰을 추가한다. 이렇게 되는 경우 특정의 시점에서 마킹이 불명확하게 되고 모든 트랜지션의 시간을 추적해야 하므로 전체 시스템의 시간진행을 추적하기가 어렵게 된다. 점화가능시간을 이용하는 경우 트랜지션은 점화가능하게 된 이후 트랜지션에 할당된 시간이 지난 후에 점화하게 된다. 이 경우 마킹은 특정시점에서 명확하게 유지되며, 전체 시스템의 시간을 추적하기가 쉽게 된다.

TTPN에서 전체 시스템의 시간을 추적하는 방법은 다음과 같다. 먼저 특정시점에서 점화가능하게 된 트랜지션을 찾는다. 점화불가능한 트랜지션의 점화가능시간은 무한대로 놓는다. 이전부터 점화가능한 트랜지션을 제외하고 새롭게 점화가능하게 된 트랜지션의 점화가능시간은 트랜지션에 할당된 시간으로 놓는다. 임의의 점화가능한 트랜지션에 대해 점화를 하고, 즉 입력플레이스에서 토큰을 제거하고 출력플레이스로 토큰을 추가한다. 그리고 모든 점화가능한 트랜지션에 대해 점화가능시간에

서 대해 점화한 트랜지션의 점화가능시간을 빼 준다. 만약 이 시간이 0보다 작은 경우 0으로 놓는다. 이 과정에서 점화하는 트랜지션때문에 점화불가능하게 된 트랜지션의 점화가능시간은 무한대로 놓는다. 점화하는 트랜지션의 할당된 시간이 아닌 점화가능시간이 도달가능그래프의 아크에서의 점화가능시간이 되며, 이 시간을 통해 도달가능그래프를 탐색하게 된다.

2.2 공구셋업시간을 고려한 FMS 스케쥴링을 위한 페트리넷 모델링

특정공구는 특정의 기계에만 장착될 수 있다. 특정공구가 특정기계에 장착된 상태에서 작업(operation)이 수행된다. 예제 2는 이와 같은 상황을 나타낸다.

예제 2(Example 2) : 여기서 다루는 FMS에는 job은 두 개, 즉 J_1과 J_2이고 서로 독립적이다. 각 job은 2개의 작업(operation)을 순서대로 거쳐야 하며, 각 작업은 기계 M_1, M_2, M_3에서 수행 가능하다. 작업에 이용되는 공구는 4개이다.

각 공구와 각 기계의 결합관계는 Table 3에 있다. Table 3에서 $a_{i,j}$는 기계 i에 공구 j가 장착되는 시간을, $d_{i,j}$는 기계 i에서 공구 j를 탈착하는 시간을 나타낸다. '×'는 해당 공구가 해당기계에 장착될 수 없음을 나타낸다. 각 job의 각 작업이 어떤 기계에서 어떤 공구를 이용하여 가공될 수 있는지는 Table 4에 있다. O_{ijkl}은 job i의 j번째 operation이 기계 k에서 공구 l을 이용하여 가공하는 것을 나타낸다. O_{ijkl}이 수행되기 위해서는 기계 k에 공구 l이 미리 장착되어 있어야 한다. 만약 기계가 연속적인 작업을 처리할 때 이미 장착되어 있는 공구를 필요로 하는 경우 공구를 탈착할 필요가 없다. 따라서 공구를 기계에 장착하는 시간은 고려되지 않는다.

Table 3 The combination of machine and tools in Example 2.

	T_1	T_2	T_3	T_4
M_1	$a_{1,1}/d_{1,1}$	$a_{1,2}/d_{1,2}$	×	$a_{1,4}/d_{1,4}$
M_2	×	$a_{2,2}/d_{2,2}$	×	$a_{2,4}/d_{2,4}$
M_3	$a_{3,1}/d_{3,1}$	$a_{3,2}/d_{3,2}$	$a_{3,3}/d_{3,3}$	×

앞에서 Table 3과 4에서는 두 가지 종류의 자원공유가 일어난다. 먼저 Table 3에서와 같이 기계는 제한된 공구를 공유한다. 그리고 Table 4에서 job의 operation은 각 기계를 공유한다. 따라서 위와 같은 문제를 페트리넷을 이용하여 모델링할 때도 기계가 공구를 공유하는 상황과 작업이 공구를 공유하는 상황을 따로 모델링할 수 있다.

먼저 Table 3에 있는 공구와 기계의 결합은 페트리넷을 이용하여 Fig. 3과 같이 모델링된다. Fig. 3에서 플레이스 p^{mi}와 p^{tk}는 각각 기계 i와 공구 k의 이용가능성을 나타낸다. 플레이스 p^{mi}에는 토큰이 하나만 들어간다. 즉 하나의 공구만이 기계에 장착될 수 있다. p^{tk}에 있는 토큰은 공구 k의 개수를 나타낸다. 따라서 이 모델에서는 공구의 종류에 따른 공구 개수를 쉽게 모델링할 수 있다.

플레이스 p^{mitk}는 기계 i에 공구 k가 장착된 상태를 나타낸다. 트랜지션 $t^a_{i,j}$와 $t^d_{i,j}$는 각각 기계 i에 공구 j를 장착하는 것과 탈착하는 것을 나타내며 Table 3에서 $a_{i,j}$와 $d_{i,j}$에 대응한다. Fig. 3에 있는 페트리넷 모델의 초기 마킹은 각 기계가 아무런 공구도 장착하지 않은 상황을 나타낸다.

Table 4에 있는 작업요구사항은 페트리넷을 이용하여 Fig. 4와 5에서와 같이 모델링된다. Fig. 4는 job J_1의 경우를, Fig. 5는 job J_2의 경우를 나

Table 4 Job requirements of Example 2.

	J_1				J_2			
1st operation	M_1		M_3		M_1			M_2
	T_2	T_4	T_1	T_3	T_1	T_2	T_4	T_4
	O_{1112}	O_{1114}	O_{1131}	O_{1133}	O_{2111}	O_{2112}	O_{2114}	O_{2124}
2st operation	M_1		M_3		M_2		M_2	
	T_1	T_3	T_2	T_3	T_2	T_4	T_1	T_3
	O_{1211}	O_{1214}	O_{1232}	O_{1233}	O_{2222}	O_{2124}	O_{2231}	O_{2233}

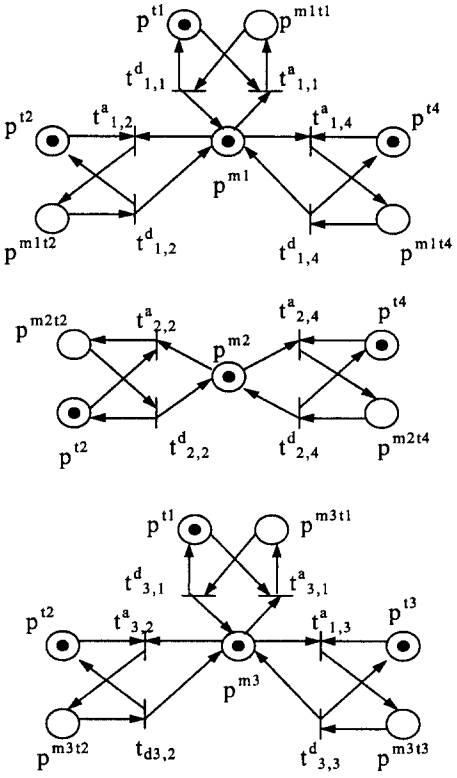

Fig. 3 The Petri net model for the combination of tool and machine in table 3.

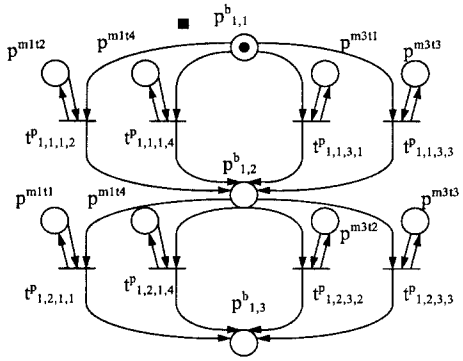

Fig. 4 The Petri net model of J_1

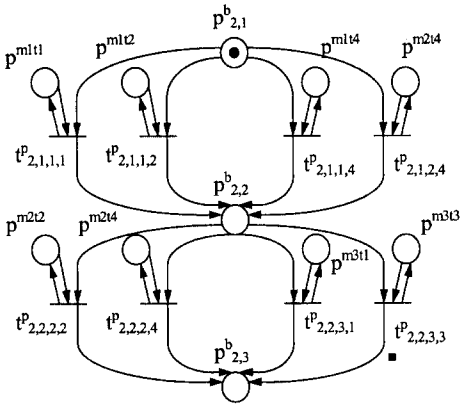

Fig. 5 The Petri net model of J_2.

타낸다. 각 그림에서 플레이스 $p^b_{i,j}$는 job i의 j번째 버퍼플레이스를 나타낸다. 그리고 palce $p^b_{i,1}$에 있는 토큰의 개수는 job i의 로트크기를 나타낸다. 플레이스 p^{mltk}는 기계 i에 공구 k가 장착된 상태를 나타내며 Fig. 3에 있는 플레이스와 같은 플레이스이다. 트랜지션 $tt^p_{j,k,l}$은 Table 4에서 O_{ijkl}에 대응한다. 즉 job i의 j번째 operation이 기계 k에서 공구 l을 이용하여 가공되는 것을 나타낸다.

3. 경험적 탐색 알고리듬

FMS를 모델링한 페트리넷의 도달가능그래프를 탐색하기 위해 다음과 같은 경험적 탐색(heuristic search)을 이용한 스케쥴링 알고리듬을 이용한다.

스케쥴링 알고리듬 :

(1) 초기 마킹 m_0를 리스트 OPEN에 넣는다.

(2) 만약 OPEN이 비어 있다면 실패로 끝낸다.

(3) OPEN에서 첫번째 마킹 m을 제거하고 m을 CLOSED에 넣는다.

(4) 만약 m이 목표마킹이면 초기 마킹에서 목표 마킹으로의 최적경로를 출력하고 끝낸다.

(5) 마킹 m에서 점화가능한 모든 트랜지션을 찾는다.

(6) 점화가능한 트랜지션 각각을 점화시켜 생겨나는 마킹 m'를 구하고, 이 마킹으로부터 m까지 포인터들을 지정한다.

(7) m의 모든 계승자 마킹 m'에 대해 다음과 같이 한다.

ⓐ 만약 m'이 이미 OPEN이나 CLOSED에 있으면 가장 작은 $g(m')$를 갖는 경로를 따라 그 포인터를 향하게 한다.

ⓑ 만약 m'이 CLOSED에 있고 포인터의 수정이 요구되면 m'를 OPEN에 넣는다.

ⓒ 만약 m'이 OPEN이나 CLOSED에 있지 않으면 $h(m')$과 $f(m')$를 계산하고 m'를 OPEN에 넣는다.

(8) 각 마킹의 f값의 순서로 OPEN에 있는 마킹들을 재배열한다.

(9) 단계 2로 간다.

경험적 탐색은 탐색과정에서 현재 탐색하고 있는 마킹은 리스트 OPEN에, 이전에 탐색된 마킹은 리스트 CLOSED에 넣는다. 새롭게 만들어진 마킹은 리스트 OPEN과 CLOSED에 저장된 마킹들과 비교된다. 경험적 탐색은 이렇게 과거의 경로를 기억함으로써 일정한 조건 하에서 국소최적(local optima)에 빠지지 않고 전체최적(global optimum)을 구하고자 한다.

경험적 탐색은 탐색을 수행하기 위해 세 가지 함수 f(m), g(m), h(m)를 이용한다. 함수 f(m)은 마킹 m을 통과하는 최적경로의 초기 마킹에서 목표 마킹까지의 비용의 예측치이다. 함수 f(m)은 두 부분으로 이루어진다. 즉 f(m)=g(m)+h(m)이다. 함수 g(m)은 지금까지 구해진, 초기 마킹에서 현재 마킹 m까지의 가장 낮은 비용이다. 함수 h(m)은 마킹 m을 통과하는 최적경로에서 마킹 m에서부터 목표마킹까지의 비용의 예측치이다. 현재의 마킹 m에서 목표마킹까지의 비용을 모르기 때문에 h(m)을 이용하여 비용을 예측하는 것이며, 이렇게 예측된 값 h(m)과 현재까지의 값 g(m)을 더해서 전체 비용의 예측치 f(m)를 구하게 된다. 탐색 알고리듬은 가장 작은 f(m)을 가지는 마킹 m을 이후 확장하게 된다.

경험적 탐색에서 현재 마킹에서 목표마킹까지의 비용의 예측치 h(m)이 실제 목표마킹까지의 비용 $h^*(m)$에 대해 다음과 같은 관계 (1)이 만족되면 탐색은 한정된 시간 내에 최적해를 찾는다는 것이 보장되어 있다.[11]

$$0 \le h(m) \le h^*(m) \quad \text{for all markings} \tag{1}$$

스케쥴링 알고리듬은 TTPN의 도달가능그래프를 탐색한다. TTPN의 도달가능그래프의 아크는 점화하는 트랜지션, 지연시간, 작업비용을 탐색을 위한 정보로 가지고 있다.

경험적 탐색에서는 탐색의 효율을 높이기 위해 경험적 지식을 h함수의 형태로 이용한다. 본 논문에서 제시하는 경험적 함수 h는 다음과 같다.

$$h1 = w \cdot [\text{estimate cost}] \cdot [\text{current marking}]^T \tag{2}$$

$$h2 = dw \cdot \sum(\text{Differences of Machine Utilization}) \tag{3}$$

식 (2)에서 [estimate cost]는 현재의 마킹에서 목표마킹까지의 비용을 예측하는 벡터이다. 벡터 estimate cost의 각 요소(element)는 마킹의 각 요소, 즉 플레이스에 대응하며 요소가 가지는 값의 의미는 해당플레이스에 있는 토큰이 목표플레이스에 가는데 소요되는 비용의 최소값이다. [estimate cost]를 구하는 방법은 다음과 같다. 페트리넷 모델에서 임의의 플레이스에 대해 플레이스가 속해 있는 job module이 식별된다. 이후 이 플레이스에서 해당 job module의 목표플레이스까지 트랜지션 점화순서 중에서 최소비용을 구하면 이 값이 이 플레이스의 estimate cost이다.

예를 들어 Fig. 2에서 플레이스 $p^b_{1,1}$은 job J_1의 module에 속하고, 플레이스 $p^b_{1,3}$은 job J_1의 목표플레이스이다. 플레이스 $p_{1,1}{}^b$에서 플레이스 $p^b_{1,3}$까지 트랜지션의 점화순서 중에서 최소의 비용을 가지는 것은 $t^b_{1,1,1}(3) \rightarrow t^b_{1,2,3}(4)$이며, 플레이스 $p^b_{1,1}$의 estimate cost는 3+4=7이 된다. 이것은 만약 플레이스 $p^b_{1,1}$에 토큰이 있는 경우 이 토큰은 목표플레이스 $p^b_{1,3}$까지 가는데 적어도 7의 비용을 필요로 한다는 것을 의미한다. 기계를 표현하는 플레이스의 estimate cost의 값은 0으로 한다. 이렇게 해서 Fig. 2에서 estimate cost의 벡터는 [7 4 0 5 2 0 0 0 0]이 된다.

공구셋업을 고려한 스케쥴링문제를 모델링한 Figs. 3, 4, 5에 있는 페트리넷 모델의 경우 estimate cost를 계산하는 방법이 달라진다. Fig. 4와 5에 있는 페트리넷 모델에서 estimate cost를 계산하는 방법은 Fig. 2에 있는 모델에서 estimate cost를 계산하는 방법과 같다. Fig. 3에 있는 페트리넷 모델의 경우 임의의 플레이스에 대해 출력 트랜지션중 최소의 점화시간을 가지는 트랜지션을 선택해서 그 트랜지션의 점화시간을 해당 플레이스의 estimate cost로 한다. 이렇게 하는 경우 Fig. 3과 같은 페트리넷 모델에서 h함수는 현재의 마킹에서 목표마킹까지 가는데 소요되는 비용을 예측하는 것이 아니라 현재의 마킹에서 점화가능한 트랜지션의 점화시간중 가장 작은 값을 예측하는 것이 된다. 즉, 현재의 마킹에서 다음 마킹까지 가는데 소요되는 비용만을 예측하는 것이다.

estimate cost 벡터에 현재의 마킹을 곱하면 현재의 마킹에서 목표마킹까지 가는데 필요한 비용의 예측치가 나오게 된다. 하지만 이렇게 구한 값은 각 job이 순차적으로 처리되었을 때의 비용을 나타낸다. 즉 가공시점에 있어서 job 사이의 중첩을 나타내지 못한다. job의 병렬적인 처리를 고려한 값

이 w이다.

식 (3)의 h2에서 Difference of Machine Utilization은 각 기계의 누적가공시간들, 즉 기계들의 부하(workload)의 차이이다. 각 기계당 가공시간은 다음과 같이 구해진다. 먼저 점화하는 트랜지션의 입력플레이스중에서 기계의 이용가능성을 나타내는 플레이스가 있으면 해당 트랜지션의 입력플레이스의 가공시간을 해당기계의 가공시간으로 더해서 구해진다. h2는 매 iteration마다 각 기계의 누적된 가공시간을 서로 빼서 절대값을 취한 후(=Differences of Machine Utilization) 이들을 더해서 이용하는 것이다. 이것은 기존의 FMS 스케쥴링연구에서 기계간의 이용부하가 균형을 이루면 전체작업시간을 줄이는 데 도움이 된다는 사실을 이용한 것이다. 다른 말로 하면 특정기계의 부하가 다른 기계에 비해 크게 증가하는 경우 이 기계에 병목현상이 생겨 전체 작업시간이 증가한다는 사실을 이용한 것이다. 이 함수는 작업시간에 대해서만 이용되면 다른 경험적 함수와 결합해서만 이용된다.

h2에서 dw는 h2가 다른 경험적 함수들과 결합할 때 크기를 조절한다. h2에서는 기계들의 작업부하 또는 이용시간들 사이의 차이를 더하게 되므로 실제 h2의 값은 매우 커지게 된다. 이 경우 h2가 다른 경험적 함수와 결합하게 되면 경험적 함수 자체가 너무 커지게 되어 목표까지의 실제 비용을 예측하지 못하게 된다. 따라서 dw를 이용하여 h2의 값을 다른 경험적 함수에 비해 상당히 작게 설정하게 된다.

이 논문에서 제시하는 경험적 함수 h1은 조건 (1)을 만족시키는 지를 알지 못한다. 따라서 정해진 시간 내에 최적해를 찾는 것을 보장하지는 않는다. w값을 조절함으로써 조건 (1)을 만족시킬 수는 있지만 조건 (1)을 만족시키는 w를 해석적으로 구하는 방법은 아직까지 연구되지 않았으며, h2에서 dw의 정확한 값을 해석적으로 구하는 방법도 아직까지 연구되지 않아서 여기서는 제시하지 않는다. 이 값들은 실험을 통한 시행착오를 통해 결정하게 된다.

4. 스케쥴링 예

본 논문에서는 세 가지 스케쥴링 사례연구를 제시한다. 첫번째 사례연구로서 앞에서 제시한 예제 2가 이용된다. 두번째 사례연구로 예제 2와 유사하지만 규모가 큰 문제로서 예제 3이 정의되고 스케쥴링 예로서 이용된다. 세번째 사례연구로 Tamaki et al.[9]의 Example E1이 스케쥴링 예로서 이용된다. 스케쥴링은 Intel Pentium 120 MHz 프로세서와 16MB의 메모리를 가진 IBM PC에서 수행되었다.

탐색에서 알고리듬과 경험적 함수의 성능을 평가하는 데에는 알고리듬의 메인루틴의 iteration횟수와 탐색을 통해 얻어진 전체 작업시간, 탐색에 있어서의 깊이, 그리고 계산에 소요되는 시간을 구해서 성능지표로 이용한다. 전체 작업시간은 트랜지션의 점화순서에 따라 각 트랜지션의 점화가능시간을 더하면 얻어진다. 계산시간은 해를 구하는 데까지 소요된 시간으로서 CPU time이 아니라 경과시간이다.

탐색에 있어서의 깊이는 해를 구할 때까지의 트랜지션의 점화횟수를 나타낸다. 트랜지션의 점화에는 job의 operation을 수행하는 것과 기계에 공구를 장착하거나 탈착하는 것을 포함한다. 위의 그림들에서 보았듯이 job의 operation의 수행을 나타내는 트랜지션의 점화횟수는 변하지 않으며 항상 일정한 횟수만큼 점화해야 job이 완수된다. 하지만 Fig. 3에서와 같이 공구를 기계에 장착하거나 탈착하는 경우 operation은 수행하지 않고 공구를 기계에 장착하거나 탈착하는 과정만 반복하게 되는 경우가 생길 수 있다. 이렇게 되면 탐색에 있어서의 깊이, 즉 트랜지션의 점화횟수가 증가한다. 따라서 탐색에 있어서의 깊이, 즉 트랜지션의 점화횟수가 적은 것은 공구를 기계에 자주 장착하거나 탈착하지 않고 job을 완수한 것으로 해석되며, 이것은 더 좋은 결과를 의미한다.

4.1 사례연구 1

예제 2의 Table 3과 4에 구체적인 값을 주면 Table 5와 6과 같다. Table 5는 공구셋업시간을 나타내고 Table 6은 특정공구를 장착한 상태에서 해당 operation의 수행시간을 나타낸다. 각 job이 처리해야 할 부품의 개수, 즉 로트크기는 각 job마다 10이다. 즉 페트리넷 모델에서 각 job마다 10개의 토큰을 가지고 시작하게 된다.

Table 7과 8은 h1, h1+h2를 이용해서 스케쥴링을 수행한 결과를 보여 준다. Table 7과 8에서 "Time", "ITER", "Depth", "MS"는 각각 계산시간(초), 메인루프의 iteration 횟수, 탐색에 있어서

의 깊이, 그리고 구해진 전체 작업시간(makespan)을 나타낸다. Table 7과 같이 w가 감소할수록 iteration과 시간은 증가하고 구해진 전체 작업시간은 감소한다. Table 8의 결과를 보면 h1만을 이용한 것보다 계산시간이 오래 걸리는 대신 전체 작업시간은 더 감소한다.

탐색에 있어서의 깊이는 w가 변해도 거의 같은 것을 알 수 있다. Fig. 4와 5에서 10개의 토큰이 목표 플레이스로 이동하기 위해서는 20번의 트랜지션점화가 필요하다. 따라서 길과된 탐색의 깊이 47에서 20을 빼면 27번의 트랜지션이 되는데, 이것은 공구를 기계에 장착하거나 탈착하는 과정이 27번 수행되었다는 것을 의미한다.

4.2 사례연구 2

사례연구 2는 Table 3과 4에서와 같은 문제를 다룬다. 즉, Table 3에서와 같은 방식으로 6개의 공구는 각각 4대의 기계에 장착가능하며 특정공구와 기계의 조합의 따라 장착시간과 탈착시간이 달라진다. 그리고 Table 4에서와 같은 방식으로 각각 4개의 연속적인 operation을 가지는 5개의 job이 처리된다. 각 operation은 3개에서 5개까지의 공구-기계의 결합 하에서 수행가능하다. 이 문제에 대한 페트리넷 모델은 Figs. 3, 4, 5와 같은 방식으로 쉽게 구성될 수 있으며 지면관계상 그림은 생략하였다. 스케쥴링은 job의 로트크기가 5인 경우와 10인 경우에 대해 수행된다.

Tables 9, 10, 11은 h1과 h2를 이용해서 각 job의 로트크기가 5인 경우와 10인 경우에 대해 스케쥴링을 수행한 결과를 보여 준다. Table 9, 10, 11에서 "Time", "ITER", "Depth", "MS"는 Table 7에서와 같은 의미로 사용된다. Table 7에서와 같이 w가 감소할수록 iteration과 시간은 증가하고 구해진 전체작업시간은 감소한다. Tables 9, 10, 11에서 w가 특정 값 이전까지 감소하면 계산시간이 크게 증가하지 않으면서 전체작업시간은 감소한다. 대신 w가 계속 감소하면 매우 많은 계산시간을 요구한다. 예를 들어 Table 9와 10에서 w가 0.5까지 계산시간이 크게 증가하지 않지만 0.4가 되면 크게 증가한다. 그리고 h1과 h2를 이용한 결과인 Table 11에서는 Table 10보다 더 좋

Table 7 The results with heuristic h 1.

w	Time	ITER	Depth	MS
1.0	0.25	89	47	132
0.9	1.48	468	47	131
0.8	1.31	390	47	129
0.7	9.86	1416	49	121

Table 8 The results with heuristic h 1+h 2.

w	dw	Time	ITER	Depth	MS
1.0	0.03	1.74	562	47	118
0.9	0.03	31.67	2611	47	119
0.8	0.03	31.67	2611	47	119
0.7	0.03	362.46	9609	49	113

Table 5 The combination of machine and tools in Example 2.

	T_1	T_2	T_3	T_4
M_1	7 / 3	10 / 6	×	5 / 9
M_2	×	12 / 6	×	8 /10
M_3	3 / 7	6 / 9	10 / 5	×

Table 6 Job requirements of Example 2.

	J_1				J_2			
1st operation	M_1		M_3		M_1			M_2
	T_2	T_4	T_1	T_3	T_1	T_2	T_4	T_4
	8	4	10	15	15	6	9	3
2st operation	M_1		M_3		M_2		M_2	
	T_1	T_3	T_2	T_3	T_2	T_4	T_1	T_3
	7	5	10	12	4	12	8	5

은 결과를 보이지만 그리 큰 향상은 아니다. 탐색에 있어서의 깊이는 w가 변해도 거의 같은 것을 알 수 있다. 대신 w가 증가할수록 미약하게나마 증가한다. 이러한 경향은 전체 작업시간을 줄이기 위해 기계에 공구를 장착하거나 탈착하는 과정이 더 필요하다는 것을 의미한다.

4.3 사례연구 3

사례연구 3으로서 Example E1은 플라스틱 성형 플랜트(plastic forming plant)에 대한 실제 예이다. 이 플랜트에는 4 대의 플라스틱 성형기계가 있고 이 기계에 장착하여 특정 형상의 플라스틱 제품을 만드는 4 종류의 금형이 있으며 각 금형마다 개수가 정해져 있다. 생산되는 플라스틱 제품의 종류는 10가지이며, 각 제품의 양이 정해져 있다. 이에 대한 정보는 Table 12에 제시되어 있다. 특정 금형은 특정기계에 따라 장착하는 시간과 탈착하는 시간이 달라진다. 이에 대한 정보는 Table 13에 제시되어 있다. 그리고 특정기계에 어떤 금형이 장착되었는가에 따라 성형 가공속도가 달라진다. 이에 대한 정보는 Table 14에 제시되어 있다. Table 12에서 보듯이 이 예제는 Table 3과 Table 4의 예제에서 각 job이 하나의 operation만 가지는 경우이다.

Tamaki et al.[9]은 이 문제를 정형화하기 위해 혼합 2차정수계획법(mixed quadratic integer programming)을 이용했다. 하지만 문제에 대한 제한조건이 34개나 된다. 해를 구하기 위해 유전자 알고리듬(genetic algorithm)과 모의아닐링(simulated annealing)을 이용하였다. 유전자 알고리듬이나 모의 아닐링을 적용하기 위해서는 문제의 해를 알고리듬에 적합한 형태로 표현해야 한다. Tamaki et al.[9]의 방법에서는 정수계획법문제를 알고리듬 수행에 적합한 형태로 바꾸는 과정(coding)이 매우 복잡하다. 그리고 구해진 해를 다시 스케쥴로 바꾸는 과정도 매우 복잡하다.

이 문제는 앞에서 Table 3과 4에서 논의한 문제와 동일한 문제이며, 따라서 동일한 방식으로 풀어질 수 있다. 그리고 해를 표현하는 부가적인 과정(coding)이 필요 없으며, 구해진 트랜지션의 점화순서는 그대로 스케쥴이 된다.

이 문제를 본 논문에서 제시된 방법으로 푸는 데는 고려해야 할 사항이 있다. 즉 이 문제에서 공구를 기계에 장착하거나 탈착하는 과정은 3에서 12

Table 9 The results when the lot size of each job is 5 using h1.

w	Time	ITER	Depth	MS
1.0	0.62	30	106	543
0.7	1.26	177	106	498
0.5	1.64	208	108	469
0.4	72.99	1807	112	440

Table 10 The results when the lost size of each job is 10 using h1.

w	Time	ITER	Depth	MS
1.0	2.25	221	206	1011
0.9	2.16	236	206	1006
0.7	3.28	342	208	971
0.5	3.46	340	208	860

Table 11 The results when the lost size of each job is 10 using h 1+h 2.

w	dw	Time	ITER	Depth	MS
1.0	0.01	2.42	235	206	1004
0.9	0.01	2.10	227	206	995
0.7	0.01	2.85	298	208	949
0.5	0.01	4.76	399	210	854

Table 12 Job requirements of Example E1.

Job	Available machine	Lot size	Die type
J_1	M_1 / M_3 / M_4	3	1
J_2	M_1 / M_3 / M_4	4	1
J_3	M_1 / M_3 / M_4	5	1
J_4	M_2 / M_3	6	2
J_5	M_2 / M_3	3	2
J_6	M_2 / M_3	10	2
J_7	M_1 / M_2	7	3
J_8	M_1 / M_2	6	3
J_9	M_1 / M_2	12	3
J_{10}	M_1 / M_3	3	4

Table 13 The combination of machines and dies in Example E1.

Die type	The number of dies	Attaching time / Detaching time			
		M_1	M_2	M_3	M_4
1	2	6 / 4	8 / 6	4 / 9	5 / 3
2	1	3 / 3	9 / 7	1 / 2	2 / 6
3	3	4 / 10	2 / 3	8 / 6	3 / 9
4	1	1 / 5	10 / 4	5 / 1	4 / 6

Table 14 The processing times of machines according to attached die type.

Machine	Attached die type			
	1	2	3	4
1	30	×	30	28
2	×	50	55	×
3	40	60	×	67
4	60	×	×	×

Table 15 The results with heuristic h1+h2.

w	dw	Time	ITER	Depth	MS
1.0	0.05	0.89	255	87	1130
0.9	0.05	4.56	1290	92	1168
0.8	0.05	1.18	365	102	1019
0.7	0.05	1.54	444	104	959

정도의 시간단위를 요구한다. 그리고 기계에서 플라스틱을 성형하는 데는 25에서 60 정도의 시간단위를 요구한다. 앞에서 설명한 대로 Fig. 3에서와 같이 공구-기계의 결합을 모델링하는 페트리넷 모델에서 estimate cost는 현재의 마킹에서 다음 마킹까지 가는데 소요되는 최소비용만을 예측하는 것이다. 하지만 이 문제에서와 같이 작업의 수행시간과 공구의 장착/탈착시간이 크게 차이가 나는 경우 공구-기계의 결합을 모델링하는 페트리넷 모델에서 estimate cost는 작업의 수행을 모델링하는 페트리넷 모델에서 estimate cost에 흡수되어 탐색에 큰 영향을 미치지 못한다. 따라서 estimate cost를 이용하는 경험적 함수 h1이 큰 효과를 발휘하지 못하게 된다. 이러한 문제에 대처하기 위해 Fig. 3에서와 같은 공구-기계의 결합을 모델링하는 페트리넷 모델에서 플레이스의 estimate cost를 모두 0으로 놓는다. 그리고 Fig. 4와 5에서와 같이 job의 처리를 모델링하는 페트리넷 모델에서의 플레이스의 estimate cost만을 고려하기로 한다.

Table 15는 플라스틱 성형플랜트에 대해 스케쥴링한 결과를 보여 준다. Table 15에서 "Time", "ITER", "Depth", "MS"는 Table 7에서와 같은 의미로 사용된다. Tamaki et al.[9]의 결과에서 전체작업시간은 959이며, 계산시간은 116에서 269 초가 소요된다. 이에 비해 본 논문에서 제시한 방법으로 문제를 풀면 같은 전체작업시간 959를 구하는데 계산시간은 2초 이내로 소요된다. 따라서 본 논문에서 제시한 스케쥴링방법이 기존의 정수계획법에 기반한 방법보다 더 좋은 결과를 제공함을 알 수 있다.

5. 결 론

본 논문에서는 공구셋업시간을 고려한 FMS 스케쥴링방법을 제시하였다. 주어진 문제는 공구가 기계를 공유하고 job의 operation이 공구가 장착된 기계를 공유하는 방식으로 생각되어질 수 있으므로 각 공유의 상황에 따라 페트리넷 모델을 구성하였다. 그리고 이러한 페트리넷 모델을 기반으로 하여 최적의 또는 준최적의 스케쥴을 구하기 위해 경험적 탐색 알고리듬을 이용하였다. 본 논문에서 제시된 방법은 자원의 공유가 복수적으로 일어나는 상황에 쉽게 적용될 수 있다. 본 논문에서 제시한 경우는 각기 다른 공구가 기계에 장착되고 이렇게 장착된 상태에서 job의 operation이 수행되는 것을 모델링했다. 이와 유사한 상황의 스케쥴링 문제들은 같은 방법으로 모델링이 가능하고 해를 구할 수 있을 것이다.

탐색의 효율을 높이기 위해 페트리넷 모델 상에

서의 거리를 예측하는 경험적 함수와 기계 사이의 부하의 균형을 이용하는 경험적 함수를 제시하였으며 스케줄링을 통하여 그 유효성을 검증하였다. 기계들 사이의 부하균형을 이용하는 h2함수의 경우 dw를 해석적으로 구하는 방법이 제시되지 않았으며 앞으로의 연구에 포함될 것이다.

스케줄링 결과를 통해 알 수 있듯이 본 논문에서 제시된 방법과 기존의 다른 방법과의 비교를 통해서 제안된 방법의 유효성을 검증하였다. 플라스틱 성형공장에 대한 예의 경우 본 논문에서 제시된 방법이 기존의 방법보다 월등히 좋은 결과를 보였다. 이 예와 같이 실제 공장에 대한 적용도 앞으로의 연구에 포함될 것이다.

참고문헌

(1) Hillion, H. P. and Proth, J., 1989, "Performance Evaluation of Job-shop System using Timed Event Graph," *IEEE Trans. on Automatic Control*, Vol. 34, No. 1, pp. 3~9.

(2) Ramamoorthy, C. and Ho, G., 1980, "Performance Evaluation of Asynchronous Concurrent System Using Petri Nets," *IEEE Trans. on Software Engineering*, Vol. 6, No. 5, pp. 440~449.

(3) Ramchandani, C., 1974, "Analysis of Asynchronous Concurrent System Using Petri Nets" *Project MAC*, TR-120, M. I. T.

(4) Reddy, C. E., Chetty, O. V. K. and Chaudhuri, D., 1992 "A Petri Net Based Approach for Analyzing Tool Management Issues in FMS," *Int. J. Prod. Res.* Vol. 30, No. 6, pp. 1427~1446.

(5) Lee, D. Y. and DiCesare, F., 1994, "FMS Scheduling using Petri nets and Heuristic Search," *IEEE Trans. on Robotics and Automation*, Vol. 10, No. 2, pp. 123~132.

(6) Lee, D. Y. and DiCesare, F., 1994, "Integrated Scheduling of Flexible Manufacturing Systems Employing Automated Guided Vehicles," *IEEE Trans. on Industrial Electronics*, Vol. 41, No. 6, pp. 602~610.

(7) Lee, D. Y. and DiCesare, F., 1993, "Scheduling Flexible Manufacturing Systems with the Consideration of Setup Times," Proceedings of the 32nd International Conference on Decision and Control, pp. 3264~3269.

(8) Peterson, J. L., 1981, *Petri Net Theory and Modeling of Systems*, Prentice Hall.

(9) Tamaki, H., Hasegawa, Y., Kozasa J. and Araki, M., 1993, "Application of Search Methods to Scheduling Problem in Plastics Forming Plant : A Binary Representation Approach," Proceedings of the 32nd International Conference on Decision and Control, pp. 3845~3850.