대한기계학회논문집(A) 제 21 권 제 5 호, pp. 769~779, 1997

〈논 문〉

페트리네트를 이용한 유연생산시스템의 다중목표 스케쥴링

임성진* · 이두용**

(1996년 8월 20일 접수)

Multiple Objective Scheduling of Flexible Manufacturing Systems Using Petri Nets

Seong Jin Yim and Doo Yong Lee

Key Words: Flexible Manufacturing Systems(FMS : 유연생산시스템), Multiple Objective Scheduling(다중목표 스케쥴링), Petri Nets(페트리네트), Heuristic Search(경험적 탐색)

Abstract

This paper presents an approach to multiple objective scheduling of flexible manufacturing systems(FMS). The approach is an extension of the scheduling method that formulates scheduling problems using Petri nets, and applies heuristic search to find optimal or near-optimal schedules with a single objective. New evaluation functions are developed to optimize simultaneously the makespan and the total operating cost. A scheduling example is used to demonstrate the effectiveness of the proposed approach.

1. 서 론

Flexible Manufacturing Systems(FMS)는 여러 대의 수치제어 공작기계들(NC machines)과 로봇들, 그리고 기계들 사이에서 가공물 또는 부품을 운반하는 자동화된 자재운반시스템(automated material handling system)으로 이루어져 있다. 수치제어 공작기계는 하나 이상의 작업을 수행할 수 있다. 따라서 기존의 job-shop과는 달리 주어진 부품(job or part)이 하나 이상의 기계에서 가공될 수 있으며, 결과적으로 부품이 여러 개의 작업경로(routing)를 가지게 한다. 이러한 다양한 routing 속에서 각 job은 선택된 목적함수를 최적화시키고 생산을 완수하기 위해 요구되는 작업을 수행하는데 있어서 가장 적절한 기계에 할당되어야 하며, 이것이 생산스케쥴링 문제이다. 스케쥴링 문제는 기본적으로 조합최적화 문제로서 NP-hard problem으로 알려져 있다. 이러한 문제를 푸는 데는 문제의 크기에 따라 복잡도가 지수적으로 증가하게 된다.

스케쥴링 문제를 푸는 것은 주로 시뮬레이션(simulation)[1]이나 선형계획법(linear programming),[2] 대기이론(queuing theory)[3]을 이용하여 이루어졌다. 시뮬레이션은 현실적으로 정확한 해와 세세한 정보를 제공하지만 계산시간과 컴퓨터메모리를 많이 요구하며 적용대상이 변하는 경우 일반성이 떨어진다. 선형계획법, 특히 정수계획법(integer programming)은 합리적인 시간내에 최적해에 가까운 해를 구해 내지만 일단 해가 적합하지 않은 경우가 존재하여 이에 대해 고려해야 하고,

*한국과학기술원 기계공학과
**회원, 한국과학기술원 기계공학과

FMS환경에 나타나는 유연한 routing이나 공유된 자원, 변화하는 로트크기(lot size) 등을 모델링하기가 어렵다. 대기이론은 어느 정도 정확한 해를 빠른 시간 내에 구해 주지만 이론적인 가정이 실제 FMS와 맞지 않고 주어진 시스템의 성능을 평가하는 모델이기 때문에 주로 설계단계에서 이용된다.

페트리네트(Petri nets)는 비동기적이며, 동시발생적인 event에 의해 시스템의 상태가 변화하는 Discrete Event System(DES)를 모델링하는데 아주 적절한 도구이다. Petri nets는 FMS 환경에 고유한 유연한 routing, 자원공유, 공유된 자원간의 상호배제, 변화하는 로트크기와 같은 특징들을 모델링하는데 적절한 도구가 된다. 일단 FMS를 Petri nets를 이용하여 모델링하면 Petri nets의 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 transition이나 플레이스(place)에 시간을 부가한 timed Petri nets를 이용한다.

일단 FMS를 Petri net으로 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 노드를 확장할 때 확장하는 노드가 얼마나 유망한 노드인가를 예측하기 위해 평가함수(evaluation function)를 이용한다.

Shih and Sekiguchi[4]는 생산시스템을 Petri net을 이용하여 모델링하고 시뮬레이션을 하면서 충돌(conflict)이 일어날 때 beam search를 이용하였다. Kim and Woo[5]는 병원의 자동운반시스템을 Petrin net을 이용하여 모델링하고 시뮬레이션하였다. Lee and DiCesare[6, 7]는 FMS를 모델링하는데 있어서 Petri net를 이용하고 스케쥴을 구하는 데 있어서 경험적 탐색을 이용하였다. Sun et al.[8]은 Lee의 스케쥴링방법을 실제 FMS에 적용하였다. 이들의 연구에서는 전체작업 완료시간(makespan)을 최소화하는 것을 목표로 하는 단일목표 스케쥴링이 수행되었다. 그러나 실제 FMS 환경에서는 makespan을 최소화하는 것 외에도 전체작업비용(total operating cost)나 납기지연(tardiness), 재고비용(inventory cost)를 최소화하는 것, 기계들 사이의 부하균형을 맞추는 것과 같은 여러 개의, 때로는 상충하는 스케쥴링 목표들이 존재한다. 따라서 현실적으로는 다중목표들(multiple objectives)을 동시에 최적화하는 스케쥴링방법이 필요하게 된다.

다중목표 최적화에서는 일반적으로 모든 목표들을 동시에 최적화하는 것은 불가능하다고 알려져 있다. 각 목표들은 다른 목표들을 희생하지 않고서는 최적화되기 힘들다는 것이다. 이러한 경우 구해진 해를 Pareto optimal solution 또는 efficient solution이라고 한다. 이러한 해를 구하기 위해 제시된 방법으로는 일반화된 동적계획법(dynamic programming)[9]이나 목표계획법(goal programming),[10] Interactive approach[11] 등이 있다. 하지만 이들 방법들은 FMS에 특징적인 유연한 routing이나 자원공유, 변화하는 로트크기 등을 모델링하기 어려워서 현실적인 FMS 스케쥴링에 적용하기 어렵다.

본 논문에서는 FMS 스케쥴링 문제를 Petri net을 이용하여 모델링하고 다중목표들을 결합한 목적함수를 이용하는 경험적 탐색을 사용하여 다중목표 스케쥴링을 수행한다. 본 논문에서는 makespan과 total operating cost를 동시에 최적화하고자 한다. 이것은 탐색 알고리듬에서 makespan과 total operating cost를 동시에 고려한 평가함수를 이용함으로써 수행된다.

2. 스케쥴링 문제를 위한 Petri net 모델링

Petri net은 시스템 내에서 발생하는 event들이 비동기성과 동시성, 선행조건을 가지는 DES에서 event의 제어와 정보의 흐름을 모델링하는 도구이다. Petri nets를 이용하면 조건을 나타내는 place와 조건의 변화를 나타내는 transition, 그리고 place 안에 위치한 점으로서 place가 나타내는 조건의 유효성을 나타내는 토큰(token)을 이용하여 DES를 아주 적절히 묘사할 수 있다. Petri net의 마킹(marking)은 모델링된 시스템의 상태를 나타내며, 이러한 상태는 transition의 점화를 통해 변화하므로 transition의 firing sequence를 제어하면 원하는 상태와 시스템 거동을 얻을 수 있다.

Fig. 1은 한 대의 기계로 두 가지 작업을 처리하는 과정을 Petri net을 이용하여 모델링한 것을 보

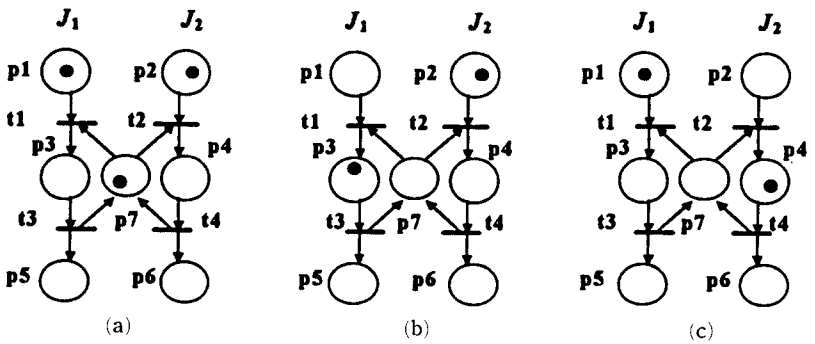

Fig. 1 Petri net modeling of shared resources

여 준다. Fig. 1에서 place p1, p3, p5는 각각 job J_1의 가공대기, 가공, 가공완료를 나타내며, transition t1, t3는 각각 job의 가공시작과 가공완료를 나타낸다. Place p2, p4, p6과 transition t2, t4는 job J_2의 경우로서 물리적인 의미는 job J_1의 경우와 같다. Place p7은 J_1과 J_2를 처리할 수 있는 기계의 이용가능성을 나타내며, 이 기계는 각 job에 공유된 자원(shared resource)이다. 각 place에 있는 토큰은 해당 place가 나타내는 조건이 유효하다는 것을 나타낸다. 즉 place p1과 p2에 있는 토큰은 J_1과 J_2가 가공 대기중이라는 것을 나타내며 place 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. 1에서와 같이 FMS의 스케쥴링은 FMS를 모델링한 Petri net 그래프에서 transition의 점화순서를 결정함으로써 수행된다. 그러나 스케쥴링에서는 가장 먼저 작업시간에 대한 정보가 필요하고 이외에 작업비용이나 기계의 부하 등과 같은 다른 정보들도 필요하다.

Timed Petri net(TPN)은 시스템의 조건을 의미하는 place나 사건을 의미하는 transition에 실제 시스템에 대응하는 시간을 부가해서 시간을 통한 성능해석이 가능하게 한 것이다. Timed Petri net에는 transition에 시간을 부가하는 timed-transition Petri net(TTPN)과 place에 시간을 부가하는 timed-place Petri net(TPPN)이 있다. TTPN의 경우 transition이 점화할 때 토큰이 사라지게 되고 결과적으로 특정시점에서 marking이 명확하지 않게 된다. 그리고 각 transition의 시간을 모두 추적해야 하는 번거로움이 생긴다. 이에 비해 TPPN은 특정시점에서 marking이 임의의 시점에서 확실하며 시간의 진행을 추적하기 위해서는 토큰을 가진 place만을 고려하면 된다. 따라서 본 논문에서는 TPPN을 이용하기로 한다.

Example 1 : FMS에서는 하나의 기계가 여러 가지 작업을 수행할 수 있기 때문에 부품의 routing이 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 첫 번째 작업을 수행하는 데에는 기계 M_2에서 2단위시간, 또는 기계 M_3에서 3단위시간이 소요된다. 그리고 작업들은 특정 기계를 공유하고 있어서 하나의 작업이 특정 기계를 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_2을 이용하는 경우 J_2의 1st operation은 기계 M_2을 이용하지 못하며, 기계 M_1를 이용해야 한다. Table 1에서 주어진 조건은 Petri net으로 모델링하면 Fig. 2와 같이 된다.

Fig. 2에서 place pmi는 기계 i의 이용가능성을

Table 1 Job requirements of example 1

Job	J_1	J_2
1st operation	$M_2(2)/M_3(3)$	$M_4(4)/M_2(3)$
2nd operation	$M_1(4)/M_3(6)$	$M_2(2)/M_3(4)$

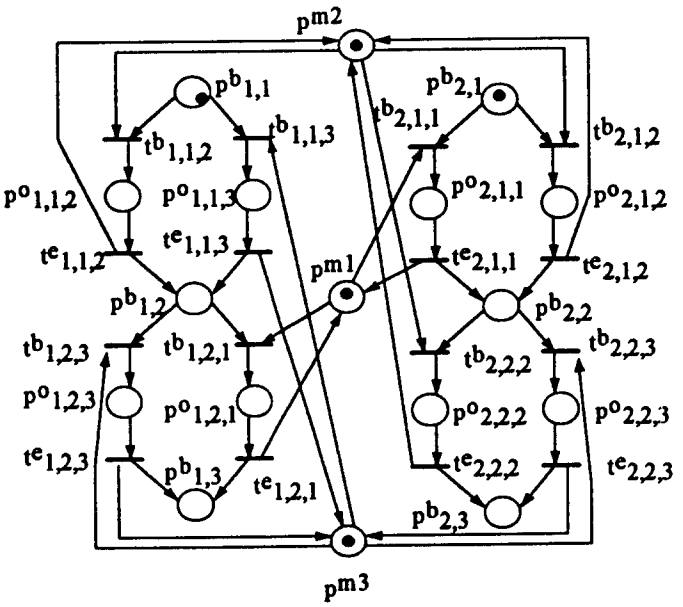

Fig. 2 Petri net mode for job requirements in example 1

나타낸다. place $p^b_{i,j}$는 job i의 j번째 buffer place를 나타낸다. 그리고 palce $p^b_{i,1}$에 있는 토큰의 개수는 job i의 로트크기를 나타낸다. place $p^o_{i,j,k}$는 job i의 j번째 operation이 기계 k에서 수행되는 상태를 나타낸다. Transition $t^b_{i,j,k}$와 $t^b_{i,j,k}$는 각각 job i의 j번째 operation이 기계 k에서 시작되는 것과 끝나는 것을 나타낸다.

Fig. 2와 같은 Petri net의 도달가능그래프는 Table 1에서 주어진 FMS가 가질 수 있는 모든 상태를 표현한다. 스케쥴은 transition의 점화순서로 주어지게 된다. 스케쥴을 구하는 과정은 모델링된 Petri net의 도달가능그래프에서 탐색을 통해 주어진 목적함수를 최적으로 하는 transition의 점화순서를 구하는 것이다. 이 논문에서 탐색과정은 경험적 탐색을 통해 이루어진다.

3. 경험적 탐색 알고리듬

FMS를 모델링한 Petri net의 도달가능그래프를 탐색하기 위해 다음과 같은 경험적 탐색 알고리듬 L1을 이용한다.[6, 7, 12]

Algorithm L1 :

(1) 초기 marking m_0를 리스트 OPEN에 넣는다.

(2) 만약 OPEN이 비어 있다면 실패로 끝낸다.

(3) OPEN에서 첫번째 marking m을 제거하고, m을 CLOSED에 넣는다.

(4) 만약 m이 목표 marking이면, 초기 marking에서 목표 marking으로의 최적경로를 출력하고 끝낸다.

(5) Marking m에서 점화가능한 모든 transition을 찾는다.

(6) 점화가능한 transition 각각을 점화시켜 생겨나는 marking m'를 구하고, 이 marking으로부터 m까지 포인터들을 지정한다.

(7) m의 모든 계승자 marking 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 marking의 f값의 순서로 OPEN에 있는 marking들을 재배열한다.

(9) 단계 2로 간다.

알고리듬 L1은 탐색을 수행하기 위해 세 가지 함수 $f(m)$, $g(m)$, $h(m)$를 이용한다. 함수 $f(m)$은 marking m을 통과하는 최적경로의 초기 marking에서 목표 marking까지의 비용의 예측치이다. 함수 $f(m)$은 두 부분으로 이루어진다. 즉 $f(m)=g(m)+h(m)$이다. 함수 $g(m)$은 지금까지 구해진 초기 marking에서 현재 marking m까지의 가장 낮은 비용이다. 함수 $h(m)$은 marking m을 통과하는 최적경로에서 marking m에서부터 목표 marking까지의 비용의 예측치이다. 현재의 marking m에서 목표 marking까지의 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 marking m을 이후 확장하게 된다.

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

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

알고리듬 L1은 TPPN의 도달가능그래프를 탐색한다. TPPN의 도달가능그래프의 아크는 점화하는 transition, 지연시간, 작업비용을 탐색을 위한 정보로 가지고 있다. 작업시간의 경우 transition의 입력 place의 토큰은 적어도 작업시간 동안, 즉 입력 place에 부가된 시간동안 지연되어야 한다. 하지만 시간지연은 모든 토큰에 적용되므로 지연시간은 place에 할당된 작업시간이 아니라 점화하는 transition의 입력 place들의 남아 있는 지연시간 중 최대값이 된다. 이에 비해 작업비용의 경우 단순히 점화가능하게 된 transition의 모든 입력 place에 할당된 작업비용의 합이다. 이것은 비용의 동시성의 개념이 없기 때문이다. 즉 동시성은 시간개념이지 비용개념은 아니다.

두 가지 목적함수를 동시에 최소화하기 위해 탐색알고리듬은 두 가지 목적함수를 하나의 목적함수로 결합한다. 본 논문에서 두 가지 목적함수를 결합하는 방법은 식 (2)와 같이 두 목적함수의 가중합(weighted summation)을 이용한다.

$$g = sc \cdot objective_1 + (1-sc) \cdot objective_2 \tag{2}$$

sc는 두 가지 목적함수 사이의 크기차이에서 오는 효과를 상쇄시키기 위한 것이다. 가령 *objective*1의 크기가 *objective*2의 크기에 비해 상당히 큰 경우 이 둘을 단순히 더하면 *objective*1만이 최소화되는 결과를 낳게 된다. 따라서 sc를 써서 이 목표들 사이의 크기의 차이를 상쇄해야 한다. 즉 sc를 이용하여 하나의 목표가 다른 목표들을 지배하지 못하게 하는 것이다. 하나의 목표가 다른 목표들을 지배하지 않는 범위에서 sc는 각 목표에 대한 중요도를 의미한다.

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

$$h1 = -w \cdot depth(m) \tag{3}$$

$$h2 = w \cdot At \cdot (E_1 - depth(m)) \tag{4}$$

$$h2' = w \cdot Ac \cdot (E_1 - depth(m)) \tag{5}$$

$$h3 = dw \cdot \sum(\textit{Differences of machine utilization}) \tag{6}$$

식 (3)에서 $h1$은 도달가능그래프에서 깊은 *marking*, 즉 *transition*이 많이 점화해서 얻어진 *marking*에 우선권을 주는 경험적 함수이다. *depth*(m)은 도달가능그래프에서 marking m의 탐색의 깊이, 즉 transition의 점화횟수를 나타낸다. 이 함수를 사용하는 경우 반드시 최적의 결과가 얻어지지는 않는다. 대신 w가 큰 경우 탐색이 깊이우선탐색과 같이 되어 탐색이 빨리 끝난다는 장점이 있다. 대신 해의 질이 나빠진다. 그리고 w가 작은 경우 탐색은 넓이우선탐색과 같이 되어 탐색이 오래 걸리게 되지만 해의 질은 좋아진다.

식 (4) (또는 식 (5))에서 $h2$(또는 $h2'$)에서 At(또는 Ac)는 미리 주어지는 모든 작업시간(또는 작업비용)의 합을 전체 transition의 개수로 나눈 평균치이다. E_1은 초기 marking에서 목표 marking까지의 예측깊이이다. 따라서 (E_1-*depth*(m))은 현재의 marking m에서 목표 marking까지 남아 있는 깊이, 즉 transition의 점화횟수이다. 여기에 각 transition이 점화하는데 소요되는 평균시간(또는 평균비용) At(또는 Ac)를 곱하면 현재의 marking에서 목표 marking까지의 작업시간(또는 작업비용)의 예측치가 된다.

$At \cdot (E_1\ depth(m))$은 모든 작업들이 각각 수행될 때의 makespan에 대한 예측치이다. 그러나 실제로 작업들이 동시에 수행되기 때문에 실제

makespan은 이 값보다 훨씬 줄어들게 된다. 이렇게 줄어드는 양을 예측하는 것이 w이다. $h2'$의 경우 작업비용에는 동시성이 없기 때문에 w는 단순히 total operating cost의 감소량을 예측하는 역할을 한다.

식 (6)에서 $h3$에서 각 기계의 누적가공시간들, 즉 기계들의 부하(workload)의 차이이다. 각 기계당 가공시간은 다음과 같이 구해진다. 먼저 점화하는 transition의 입력 place중에서 기계의 이용가능성을 나타내는 place가 있으면 해당 transition의 입력 place의 가공시간을 해당기계의 가공시간으로 더해서 구해진다. $h3$는 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 makespan을 줄이는 데 도움이 된다는 사실을 이용한 것이다. 다른 말로 하면 특정기계의 부하가 다른 기계에 비해 크게 증가하는 경우 이 기계에 병목현상이 생겨 makespan이 증가한다는 사실을 이용한 것이다. 이 함수는 작업시간에 대해서만 이용되면 다른 경험적 함수와 결합해서만 이용된다.

$h3$에서 dw는 $h3$가 다른 경험적 함수들과 결합할 때 크기를 조절한다. $h3$에서는 기계들의 작업부하 또는 이용시간들 사이의 차이를 더하게 되므로 실제 $h3$의 값은 매우 커지게 된다. 이 경우 $h3$가 다른 경험적 함수와 결합하게 되면 경험적 함수 자체가 너무 커지게 되어 목표까지의 실제 비용을 예측하지 못하게 된다. 따라서 dw를 이용하여 $h3$의 값을 다른 경험적 함수에 비해 상당히 작게 두게 된다. dw의 정확한 값을 해석적으로 구하는 방법을 아직까지 연구되지 않아 여기서는 제시하지는 않으며 실험을 통한 시행착오를 통해 결정하게 된다.

다중목표들이 하나의 g함수로 결합되는 방식처럼 목표까지의 거리를 예측하는 h함수도 하나의 h함수로 결합되어야 한다. 여기서, h함수는 g함수가 결합되는 것과 같은 방식으로 식 (7)과 같이 합을 이용하여 결합된다.

$$h = sc \cdot (h \text{ of } objective_1) + (1-sc) \cdot (h \text{ of } objective_2) \quad (7)$$

이렇게 결합되어 얻어진 평가함수 $f(m)$은 다음의 식 (8)과 같다.

$$f = sc \cdot objective_1 + (1-sc) \cdot objective_2 + sc \cdot (h \text{ of } objective_1) + (1-sc) \cdot (h \text{ of } objective_2) \quad (8)$$

4. 스케쥴링 결과

본 논문에서 스케쥴링 예로서 제시하는 FMS는 Table 2와 같다. Table 2는 각 job의 자원정보와 각 operation들의 선후관계, 그리고 작업시간과 작업비용에 대한 정보를 보여 주고 있다. 처리해야 할 job은 J_1, J_2, J_3, J_4, J_5이고, 각 job은 4개의 순서 지워진 작업(operation)을 거쳐서 완성된다.

Table 2 Job requirements of scheduling example

Job	J_1	J_2	J_3	J_4	J_5
1st Operation	M_1/M_3	M_1/M_2	$M_1/M_2/M_3$	M_2/M_3	M_1/M_3
time	7/4	8/12	10/15/8	9/5	10/15
cost	8/2	5/6	5/4/5	7/9	9/8
2nd Operation	M_2	M_{32}	M_2/M_3	M_1/M_3	M_2/M_3
time	3	4	2/6	6/2	7/14
cost	13	15	14/12	8/10	3/7
3rd Operation	M_1/M_3	M_1/M_2	M_1/M_2	M_2/M_3	M_1/M_2
time	3/6	7/14	2/4	7/12	5/8
cost	9/3	2/4	13/8	10/5	6/10
4th Operation	M_1/M_2	M_1/M_3	M_1/M_2	$M_1/M_2/M_3$	$M_1/M_2/M_3$
time	2/4	8/4	6/3	9/6/3	4/6/8
cost	10/9	6/10	7/5	10/7/9	3/8/4

각 작업(operation)은 여러 대의 기계에서 가공가능하며, 어떤 작업이 어떤 기계를 이용가능한지는 Table 2에 있다. Table 2에서 "time"과 "cost"는 각각 특정 기계에서 작업이 처리될 때 가공시간과 가공비용을 나타낸다. 각 job의 처리해야 할 부품의 개수, 즉 로트크기는 5이다. 따라서 탐색에 있어 초기상태는 각 기계를 나타내는 place에 토큰이 하나씩 있고, 각 job의 시작을 나타내는 place에 토큰이 5개씩 있는 경우이다.

문제를 푸는 과정에서 경험적 함수의 효과를 알아 보기 위해 위의 예에서 각 작업에 이미 할당된 작업시간과 작업비용을 다른 작업에 무작위적으로 할당하여 15개의 문제를 만든후 스케쥴링을 수행한다. 이것은 FMS에 대한 Petri net 모델은 그대로 두고 각 place에 할당된 시간만을 무작위적으로 교환하는 것이다. 이렇게 함으로써 가공시간이나 가공비용의 변화에 따른 탐색 알고리듬의 성능을 비교한다. 스케쥴링 결과에 대한 통계적인 해석을 통해 각 경험적 함수의 성능과 여러 가지 목표들이 감소하는 양상을 알아 본다.

탐색에서 알고리듬과 경험적 함수의 성능을 평가하는 데에는 알고리듬의 메인루틴의 iteration횟수와 탐색을 통해 얻어진 makespan, 그리고 total operating cost를 성능지표로 사용한다. 스케쥴링을 수행할 때마다 각 경험적 함수의 특정한 w에 대해 iteration 횟수와 makespan 또는 total operating cost를, 그리고 다중목표 스케쥴링의 경우 makespan과 total operating cost를 동시에 구하게 된다.

먼저 단일목표 스케쥴링이 수행된다. makespan과 total operating cost가 스케쥴링 목표가 되어 각각 최소화된다. 이 과정에서 앞에서 제시한 경험적 함수들의 효과를 알아 본다. 여기서 구해진 값들은 이후 다중목표 스케쥴링의 결과를 평가하는 기준으로 이용된다. 다음으로 다중목표 스케쥴링이 수행된다. 다중목표 스케쥴링에서는 makespan과 total operating cost가 동시에 최소화된다.

4.1 단일목표 스케쥴링

단일목표 스케쥴링의 경우 makespan과 total operating cost를 각각 최소화하게 된다. 여기서는 각 경험적 함수들의 특성들이 파악된다. 그리고 이러한 경험적 함수들이 결합되었을 때의 효과도 알아 본다. 먼저 makespan을 최소화한 결과를 살펴보면 다음과 같다. Table 3과 4에서 $h1$과 $h3$ 또는 $h2$와 $h3$를 결합할 때 $h3$의 dw는 0.1로 한다.

Table 3과 Table 4에서 "ITER", "Time", "MS"는 각각 알고리듬의 메인루프의 iteration 횟수, 계산시간(seconds), 결과되는 makespan을 나타낸다. Table 3을 보면 $h1$만을 이용하는 경우 w가 증가할수록 iteration이 줄어들며, 대신 makespan은 증가하는 것을 알 수 있다. 따라서 이러한 경험적 함수에서 큰 w를 사용하는 경우 수렴속도를 빠르게 한다는 것을 알 수 있다. 이 경우 탐색은 깊이우선탐색(depth-first search)이 된다. 이렇게 되면 해답의 질이 떨어지게 되어 makespan은 증가한다. 그리고 w가 작아질수록 iteration은 증가하여 더 많은 marking을 탐색한다는 사실을 알 수 있다. 이 경우 탐색은 넓이우선탐색(breadth-first search)이 된다. 더 많은 marking을 탐색한만큼 해답의 질이 좋아지게 되어 makespan은 감소한다. 하지만 w를 너무 낮추는 경우 iteration이 크게 증가되어 해를 구하는데 걸리는 시간이 크게 증가하게 된다. 예를 들어

Table 3 Scheduiling results of minimizing the makespan using h1

Using h1				Using h1+h3			
w	ITER	Time	MS	w	ITER	Time	MS
5	214.73	1.40	212.33	5	204.27	0.93	188.60
4	242.20	1.87	211.33	4	211.73	0.87	188.07
3	288.27	2.20	208.00	3	225.73	0.93	187.87
2	416.00	4.40	203.07	2	352.23	2.47	185.67
1.5	996.53	43.73	186.87	1.5	927.69	12.85	180.31

Table 3에서 $h1$만을 이용하는 경우 w를 2에서 1.5로 하면, 계산시간은 4.40 seconds에서 43.73 seconds로 크게 증가한다. Table 3에서 $h1$과 $h3$를 함께 이용하는 경우 $h1$만을 이용하는 경우에 비해 같은 iteration을 수행하면서, 또는 같은 계산시간을 소요하면서 makespan은 훨씬 좋아지게 된다. 따라서 기계들의 가공시간의 차이를 이용한 경험적 함수가 탐색에 있어 유용함을 알 수 있다.

$h2$를 이용한 Table 4의 결과는 $h1$을 이용한 Table 3과 유사한 결과를 보여 준다. 즉 같은 계산시간을 소요하면서 같은 정도의 해를 구해 낸다. 이것은 $h1$과 $h2$ 모두가 탐색에 있어서 marking의 깊이($depth(m)$)를 경험적 함수로 이용하기 때문이다.

다음으로 total operating cost를 최소화한 경우를 살펴 본다.

Table 5에서 "ITER", "Time", "TOC"는 각각 알고리즘의 메인루프의 iteration 횟수, 계산시간(seconds), 결과되는 total operating cost를 나타낸다. Table 5에서 $h1$과 $h2$를 이용하여 total operating cost를 최소화한 결과를 보여 준다. 같은 경험적 함수를 이용하였으므로, 앞에서 makespan을 최소화한 결과와 유사한 결과를 보여준다.

단일목표를 최소화한 결과를 보면 w를 줄여도 해가 크게 개선되지 않는 것을 알 수 있다. 이것은 경험적 함수가 현재 marking에서 목표 marking까지의 거리를 정확히 예측하지 못하고 있다는 사실을 의미한다. Fig. 3을 보면 다음과 같은 사실을 알 수 있다. h함수는 현재의 상태와 목표상태까지의 거리의 예측값이고, 실제 탐색과정에서 h의 함

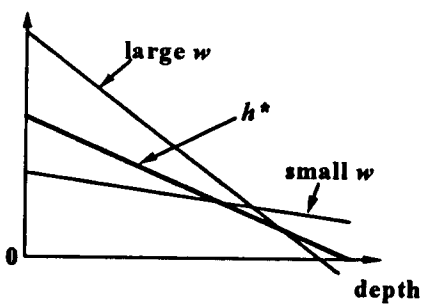

Fig. 3 The variation of h function according to w.

Table 4 Scheduiling results of minimizing the makespan using h2 and h3

Using h2				Using h2+h3			
w	ITER	Time	MS	w	ITER	Time	MS
1.0	243.40	1.87	210.60	1.0	219.08	0.93	188.20
0.8	288.27	2.27	208.00	0.8	236.60	0.93	187.73
0.6	304.73	2.60	205.13	0.6	364.93	3.53	186.00
0.5	470.00	6.53	201.47	0.6	827.87	19.13	184.20
0.4	1318.50	64.57	193.50	0.4	2392.00	228.08	177.75

Table 5 Scheduiling results of minimizing the total operating cost using h1 and h2′

Using h1				Using h2′			
w	ITER	Time	TOC	w	ITER	Time	TOC
10.0	231.27	0.67	692.12	1.5	337.20	1.20	668.20
6.0	339.33	1.20	669.13	1.4	353.53	1.40	667.80
5.0	448.00	2.33	661.00	1.2	764.69	9.15	648.38
4.5	764.31	9.31	648.77	1.2	764.69	9.15	648.38
4.0	2347.82	87.09	625.82	1.1	887.67	10.00	636.42

수값도 선형이 아니다. 그런데 위에서 이용된 경험적 함수는 모두 선형이다. h함수가 선형이므로 작은 w를 주는 경우 탐색의 초기에는 실제 h값보다 작게 되고, 탐색의 후반부에서는 실제 거리 h^*값보다 크게 된다. w값을 크게 주는 경우 탐색의 초기에는 실제 h값보다 크게 되고, 탐색의 후반부에서는 실제 h값보다 작게 된다. 따라서 탐색을 효율적으로 인도하지 못하게 되어 w값의 변화로는 최적의 해를 얻을 수 없다.

$h1$이나 $h2$와 함께 이용되는 $h3$는 이러한 경향을 상쇄시킨다. 즉, 각 기계의 이용시간에 대한 정보를 이용함으로써 선형인 h함수를 실제 h^*함수에 근사시킬 수 있는 것이다. 위에서의 결과를 이러한 사실을 보여 준다.

다음으로는 여기서 얻어진 결과들을 이용하여 makespan과 total operating cost를 동시에 최적화하는 다중목표 스케쥴링에 대해 알아 본다.

4.2 다중목표 스케쥴링

다중목표 스케쥴링에서는 makespan과 total operating cost를 동시에 줄이기 위해 둘을 결합한 g함수를 이용한다. 작업비용과 작업시간의 크기가 다르므로 sc를 이용하여 크기차이를 상쇄시킨다. sc의 값은 앞에서 단일목표 스케쥴링에서 구한 값을 이용하여 구한다. 예를 들어 단일목표 스케쥴링에서 최소의 makespan 값은 177.5이고, 최소의 total operating cost의 값은 625.87이다. 따라서 175.75/625.87=0.284≒0.3이 구해진다. 이 값은 makespan의 크기는 total operating cost의 크기의 약 30%라는 것을 의미한다. 따라서, 식 (8)에서 *objective*1이 *makespan*이고 *objective*2가 total operating cost라고 하면, $(1-sc)$의 값은 0.3이 되어 sc의 값은 0.7이 된다. Table 6과 8에서 $h1$과 $h3$를 결합할 때 $h3$의 dw는 0.1로 한다.

Tables 6, 7, 8에서 "ITER", "Time", "MS", "TOC"는 Table 4와 Table 5에서와 같다. 결과를 보면 w가 작아질수록 makespan과 total operating cost가 동시에 감소한다는 것을 알 수 있다. 물론 해답의 질은 단일목표 스케쥴링에서의 결과보다 좋지 않으며 iteration도 크게 증가한다. $h1$이나 $h2$를 독자적으로 이용하는 경우에는 makespan보다는 total operating cost의 감소 폭이 더 크며, $h1$이나 $h2$를 $h3$와 함께 이용하는 경우에는 total operating cost보다 makespan의 감소 폭이 더 크다.

이제까지의 결과에서 경험적 함수를 사용할 때 함수의 w와 결과되는 iteration, 그리고 해답의 질 사이에는 일정한 관계가 존재함을 알 수 있다. 가

Table 6 Scheduiling results of minimizing the makespan and the total operating cost using h1 and h3

Using h1					Using h1+h3				
w	ITER	Time	MS	TOC	w	ITER	Time	MS	TOC
6.0	666.00	10.80	213.80	731.53	6.0	700.13	8.53	205.87	745.40
5.5	1736.27	27.03	212.93	725.07	5.5	1495.43	28.93	204.00	734.93
5.0	4373.23	234.13	207.08	711.08	5.0	3887.00	150.09	199.45	715.80

Table 7 Scheduling results of minimizing the makespan and the total operating cost using h2, h2′

Using h_2+h_2'					
w of h2	w of h2′	ITER	Time	MS	TOC
1.0	1.5	294.93	0.67	221.73	728.13
0.8	1.4	350.47	0.90	220.93	731.40
0.6	1.3	423.60	1.20	221.67	731.40
0.4	1.2	917.13	6.05	211.00	733.00
0.35	1.1	3873.29	179.25	210.07	718.79

Table 8 Scheduling results of minimizing the makespan and the total opearting cost using h2, h2′ and h3

Using h2+h3					
of h2	w f h2′	ITER	Time	MS	TOC
1.0	1.5	268.33	1.13	216.40	748.20
0.8	1.4	309.87	1.47	212.13	747.53
0.6	1.3	389.27	2.07	209.20	749.60
0.4	1.2	740.80	7.27	204.00	738.73
0.35	1.1	3215.50	205.00	203.50	733.57

령 w를 크게 하는 경우 작은 iteration, 즉 빠른 시간내에 답을 구해 내지만 좋은 답을 구하지는 못한다. 그리고 w를 작게 하는 경우 시간은 매우 오래 걸리게 된다. 예를 들어 Table 7과 8에서 알 수 있듯이 w를 특정 수치 이상으로 낮추면 계산시간은 급격히 증가하게 된다. 대신 상대적으로 좋은 해를 구할 수 있다.

다중목표 스케쥴링에서 sc는 하나의 목표가 다른 목표들을 지배하지 않는 범위에서 각 목표에 대한 가중치를 의미한다. 하나의 목표가 다른 목표를 지배하지 않는 범위에서 sc값의 변화는 각 목표들에 대한 가중치의 변화를 의미하며 높은 가중치를 받은 목표는 다른 목표들보다 더 우선적으로 최적화 될 것이다. 실제 스케쥴링을 수행할 때 각 목표에 대한 가중치는 상황에 따라 변화하며 일정하지 않다. 예를 들어 실제 상황에서 비용을 줄이는 것보다 생산물량을 빨리 만드는 것이 더 중요한 경우 식 (8)에서 *objective*1이 makespan이므로 sc의 값이 total operating cost를 지배하지 않는 범위에서 더 커져야 한다. 따라서 sc를 결정하는 일은 각기 다른 상황에 처해 있는 설계자의 주관적인 일이다. 다중목표 스케쥴링에서는 모든 목표에 대해 해가 최적이 되는 것은 불가능하지만 특정의 sc에 대한 최적해는 존재한다.

5. 결 론

본 논문에서는 Petri net와 경험적 탐색을 이용하여 실제 FMS에 존재하는 여러 가지 목표를 동시에 최적화하는 방법을 제시하였다. 이것은 경험적 탐색에서 다중목표를 하나의 g함수에 결합함으로써 수행되었다. 탐색의 효율을 높이기 위해 탐색의 깊이에 따른 makespan 또는 total operating cost의 예측치와 기계별 가공부하의 차이를 이용하는 경험적 함수를 제시하였다. 이 함수들은 단일목표를 줄이는 스케쥴링, 특히 makespan을 줄이는데 효과가 있음을 보았다. 이러한 결과를 기반으로 각 경험적 함수를 다중목표 스케쥴링에 적용하였다.

본 논문에서 제시한 경험적 함수는 선형이므로 탐색의 효율을 크게 높이지 못하며, 최적해를 보장하지 못한다. 실제 목표 marking까지의 거리를 정확히 예측할 수 있는 경험적 함수를 개발하는 것이 앞으로의 과제이다. 그리고 본 논문에서 제시한 방법에서는 두 가지 목표들을 결합할 때 합을 이용하는 방법을 제시했다. 이외에 다른 방법은 벡터를 이용하여 여러 가지 목표를 결합하는 것으로 각 목표들을 벡터로 보고, 벡터의 크기를 g함수로 이용하는 것이다. 이와 더불어 FMS에 대한 Petri net 모델링을 확장하는 것도 앞으로의 과제이다.

참고문헌

(1) Chang, Yie-Long and Sullivan, R. S., 1986, "Using SLAM to Design the Material Handling System of a Flexible Manufacturing System," *International Journal of Production Research*, Vol. 24, pp. 15~26.

(2) Chen, Y. U. and Askin, R. G., 1990, "A Multiobjective Evaluation of Flexible Manufacturing System Loading Heuristics," *International Journal of Production Research*, Vol. 28, pp. 895~911.

(3) Stecke, K. E. and Solberg, J. J., 1985, "The Optimality of Unbalancing Both Workloads and

Machine Group Sizes in Closed Queuing Networks of Multi-server Queues," *Operations Research*, Vol. 33, No. 4, pp. 882~910.

(4) Shih, H. M. and Sekiguchi, T., 1991, "A Timed Petri Net and Beam-search Based On-line FMS Scheduling System with Routing Flexibility," *Proceedings of the* 1991 *IEEE International Conference on Robotics and Automation*, pp. 2548~2553.

(5) Kim, J. W. and Woo, J. W., 1994, "A Study on the Design and Real-time Operation of an Automatic Material Handling System," *Journal of KSME*, Vol. 34, No. 2, pp. 101~111.

(6) 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.

(7) 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.

(8) Sun, T. H., Cheng, C. W. and Fu, L. C., 1994, "A Petri Net Based Approach to Modeling and Scheduling for an FMS and a Case Study," *IEEE Trans. on Industrial Electronics*, Vol. 41, No. 6, pp. 593~601.

(9) Carraway, R. L., Morin, T. L. and Moskowitz, H., 1990, "Generalized Dynamic Programming for Multiattribute Optimization," *European Journal of Operations Research*, Vol. 44, pp. 95~104.

(10) Lee, S. M. and Jung, H. J., 1989, "A Multiobjective Production Planning Model in a Flexible Manufacturing Environment," *International Journal of Production Research*, Vol. 27, No. 11, pp. 1981~1992.

(11) Geoffrion, A. M., Dyer, J. S. and Feinberg, A., 1972, "An Interactive Approach for Multicriteria Optimization with an Approach to the Operation of a Academic Department," *Management Science*, Vol. 19, pp. 357~368.

(12) Nilsson, N., 1980, *Principles of Artificial Intelligence*, Palo Alto, CA : Tigoa.