Journal of the Korean Institute of Industrial Engineers
Vol. 16, No. 2, Dec., 1990

FMS의 실시간 일정계획을 위한 수리적 의사결정에 관한 연구

김종한* · 박종헌** · 박진우** · 정성진**

A Mathematical Decision Making Model for Real-Time Scheduling of an FMS

Jong-Han Kim*, Jong-Hun Park**, Jin-Woo Park** and Sung-Jin Chung**

Abstract

This paper deals with the production scheduling problems of a dedicated Flexible Manufacturing System. In this work, a new mathematical formulation is proposed and two heuristic algorithms which can generate real-time schedules are suggested. Example problems to demonstrate the good performance and the validity of these two proposed algorithms are also included.

1. 서 론

FMS(Flexible Manufacturing System)는 NC (Numerical Control) 기계와 자동화된 MHS(Material Handling System) 등이 제어 컴퓨터에 의해 통합적으로 운영되는 시스템으로서 소량생산 시스템인 Job Shop의 유연성을 유지하면서 대량생산 시스템의 효율성을 달성하고자 하는 자동화된 생산 시스템이다. 이를 위해서는 시스템의 설계에서 운용계획에 이르기까지 면밀한 분석이 필요한 까닭에 현재 이에 관한 많은 논문이 발표된 바 있으며, 또한 많은 연구가 진행중이다. 특히 생산계획과 관련하여는 다양한 계층적 접근방법이 채택되고 있다[2, 11].

FMS의 일정계획(Scheduling) 문제는 동적인 Job Shop의 일정계획 문제와 유사하나 전통적인 Job Shop 또는 Flow Shop의 일정계획 문제와는 많이 다르다. FMS에서는 각 부품이 여러개의 대체 공

* 산업과학기술 연구소
** 서울대학교 산업공학과

정들을 가질 수 있고 전통적인 Job Shop과는 달리 공구 교환시간은 거의 무시할 수 있으며, 또한 전통적인 Flow Shop과는 달리 동시에 여러 종류의 부품들이 시스템에서 가공될 수 있다.

FMS에서의 일정계획 문제는 시스템의 상태와 상위 단계의 계획인 동시 가공품목의 수 및 종류에 의존하는데 FMS의 유연성과 동적인 특성때문에 FMS 문제의 수학적 분석방법은 (1) 적당한 Formulation이 어렵고 (2) Formulation이 되더라도 현실적으로 받아들일만한 시간에 풀리지 않으며 (3) 풀리더라도 모델에서 필요로 하는 자료의 수집과 유지에 소요되는 비용이 모델이 제공하는 이익보다 클 수 있다. 이러한 이유들 때문에 기존의 수학적 분석방법은 대부분 실용적 해를 제공하지 못하고 있다. 이러한 맥락에서 본 논문에서는 FMS의 실시간 일정계획에 대한 새로운 수학적 모형을 제시하고 이를 실시간에 풀 수 있는 2 종류의 Heuristic 알고리즘을 제시하고자 한다. 이하 제 2절에서는 기존의 연구현황에 대해 언급할 것이며, 제 3절에서는 새로운 수리적 모형, 제 4절에서는 Heuristic 일정계획 알고리즘, 제 5절에서는 결론 및 추후 연구과제가 제시될 것이다.

2. 연구의 현황

FMS의 일정계획을 위하여는 많은 연구결과가 발표되어 있으나 대략 다음 3개의 접근방법으로 대별할 수 있다.

(1) 수학적 계획법

(2) 실시간 기계별 작업선택 규칙(Dispatching Rule)에 의한 방법

(3) 기타

그리고 이것들의 목적함수는 한정된 시간에서의 생산량 최대화, 부품의 지연횟수 최소화, 기계들의 이용률 최대화 등을 들 수 있다. 그런데, FMS에서는 기계가 고가이므로 대개 기계이용률의 최대화가 보다 중요한 목적함수로 사용되며 이것은 제작기간의 최소화와 동일한 개념으로 사용된다. 이하 각각의 접근방법에 대해 좀 더 자세히 언급하여 보기로 한다.

(1) 수학적 계획법

Kimemia와 Gershwin[5]은 FMS를 여러개의 공구를 가진 공작기계 Node 및 공정경로 Arc로 표현되는 Network로 생각해서 생산량을 최대로 하는 생산비율 결정 모형을 제시하였고, Afentakis[11]는 Makespan을 최소화 하는 정수계획 모형을 제시하였다. 또한 Pourbabai[3]은 지연시간을 최소화 하는 정수계획 모형을 제시하였다. 단 이들 모형들은 이론바 NP-Hard 분류에 속하는 문제들로서 실시간내에 해를 제시할 수 없다는 문제점이 있다.

(2) 실시간 기계별 작업선택 규칙(Dispatching Rule)에 의한 방법

Stecke와 Solberg[9]는 Caterpillar사의 FMS에 대해서 여러 Loading 정책하에 16개의 Dispatching Rule들에 대한 시뮬레이션 연구결과 Loading 정책에 따라 다소의 차이는 있으나 SPT／TOT가 가장 많은 생산량을 도출함을 보였다.

Malstrom 등[4]은 기계선택 및 부품선택의 여러 Rule들에 대해 SLACK-WINQ(Least Remaining Slack Time－Least Work in Queue) Rule과 SPT-WINQ Rule이 부품을 가장 많이 생산하는 것을 보여주었다.

또한 Iwata 등[7, 8]은 여러 Dispatching Rule에 대해서 공작기계, 공구, 이송장치의 순서로 생산자원을 선택하는 것에 대해서 시뮬레이션한 결과 ESTA(Earliest Starting Time with Alternatives Considered)가 가장 높은 기계 가동률을 도출하고, EFTA(Earliest Finishing Time with Alternatives Considered) Rule이 가장 많은 부품을 생산함을 보였다.

(3) 기 타

Slomp 등[6]은 생산자원 선택에 대한 세가지

발견적 방법을 제시하고 이들의 개당 평균 생산시간에 대한 시뮬레이션 결과를 보여주었다.

Sarin 등[12]은 수리적으로 분석하기 어려운 FMS 일정계획의 동적인 상황에 AI 기법을 사용하는 것에 관하여 언급하였다.

Nakamura와 Shingu[10]는 부하균등화 측면에서 Makespan을 최소화 하는 발견적 방법을 제시하였으나, 동일 부품은 동일 공정경로를 따라 생산되어야 한다는 단점이 있다.

Kusiak[1]는 FMS의 조립작업까지 고려한 2단계 알고리즘, 즉 첫째 단계에서는 Johnson의 알고리즘을 이용해서 부품의 생산순서를 정하고, 둘째 단계에서는 각 작업의 우선순위, 작업의 가능 여부 등을 고려해서 작업순서를 정하는 방법을 제시하였다.

3. 수리적 모형의 개발

3-1. 문제의 설명 및 가정

본 절에서는 기계의 수 및 기계간의 이동시간 등 시스템에 대한 명세가 주어져 있으며, 생산될 부품의 수, 부품의 생산비율, 부품이 생산될 수 있는 여러 Routing, 각 작업이 수행될 수 있는 기계들에 대한 자료가 주어졌을 때 생산량 최대화를 위해 부품이 시스템에 Loading되는 순서, 부품이 생산될 Routing의 결정, 각 기계에서 수행될 작업들의 순서를 결정하는 문제에 대한 수리적 모형을 제시하고자 한다.

시스템에 대한 가정은 다음과 같다.

(1) 각 기계는 한번에 한 작업만을 수행할 수 있다.

(2) 준비시간은 작업순서에 무관하다.

(3) 기계 앞에는 일정량의 Buffer Storage가 있다.

(4) 각 작업의 작업시간 및 이동시간은 비확률적이다.

(5) 대체 Routing들이 허용된다.

(6) 각 부품의 첫 작업은 Loading이고, 마지막 작업은 Unloading이다.

(7) Preemption은 허용하지 않는다.

위의 가정은 기존의 연구에서와는 달리 대체 Routing 및 일정량의 Buffer Storage가 존재함을 가정하였다.

3-2. 수리적 모형

목적함수 : $\text{Max} \sum_i \sum_k \sum_t Cp_{ik}\, Op_{ik}(t)$

제약조건 :

1. $I_l(t) = I_l(t-1) + C_{l-1}(t-d) - a_{l-1,l}\, O_l(t)$

   $l=1, 2, \cdots, N,\ t=1, 2, \cdots, T$

2. $\sum_{i \in j} I_i(t) < b_j,\ j=1, 2, \cdots, M,\ t=1, 2, \cdots, T$

3. $\sum_{i \in j} O_i(t) < 1,\ t=1, 2, \cdots, T$

4. $\sum_k \sum_t Cp_{ik}\, Op_{ik}(t) > N_i,\ i=1, 2, \cdots, I$

5. $[1-O_l(t-1)]\, O_l(t)\, [\sum_{j=0}^{Pl-1} O_l(t+j)]$

   $= [1-O_l(t-1)] \times O_l(t) \times P_l$

   $t=1, 2, \cdots, T,\ l=1, 2, \cdots, N$

6. $O_l(t) = 0 \text{ or } 1$

   $l=1, 2, \cdots, N,\ t=1, 2, \cdots, T$

i : 부품 index, $i=1, 2, \cdots, I$

j : 기계 index, $j=1, 2, \cdots, M$

k : Routing index, $k=1, 2, \cdots, Q$

l : 작업 index, $l=1, 2, \cdots, N$

P_{ik} : 부품 i의 Routing k의 작업수

P_l : 작업 l의 작업시간

N_i : Batch당 생산될 부품의 양

$N = \sum\sum P_{ik}$: 총 작업수

$O_l(t)$: Indicator Function
시간 t−1에서 t 사이 작업 l의 수행 유무를 나타냄

$a_{l,l+1}$: 작업 l+1에서 작업 l이 끝난 부품의 시간단위당 처리능력

C_l : Operation l의 시간단위당 처리능력

$I_l(t)$: Time t−1에서 t 사이에 Operation l 앞의 재고량

b_j : 기계 j의 Buffer Storage 크기

$i \in j$: 기계 j에서 수행되는 부품의 집합

위의 모형에서 제약식 1은 재고의 순서, 2는 저장장소 크기, 3은 기계 용량, 4는 생산할 부품 수, 5는 Preemption 방지, 6은 Indicator Function의 제약조건에 관한 것이다. 그러나 위의 계산식은 Polynomial Time에 해를 제공하는 어떤 알고리즘도 알려져 있지 않은 NP-Hard 문제이므로 실제 문제에 대해 받아들일 만한 시간에 해를 제공할 수 없다. 이러한 맥락에서 다음 절에서는 이같은 모형을 기본으로 실시간내에 적절한 해를 제공하는 Heuristic 알고리즘을 제시하고자 한다.

4. Heuristic을 이용한 실제적 해법

4-1. 문제의 접근방법

생산될 부품이 시스템에 투입되는 순서, 부품이 생산되는 Routing 및 각 기계에서 수행될 순서를 결정하기 위한 Heuristic 알고리즘의 기본적인 생각은 다음과 같다.

(1) Batch에서 생산될 부품을 MPS(Minimal Part Set)으로 나누어서 문제의 크기를 줄인다. (2) Routing 선택은 부품 투입순서 결정 다음에 한다. (3) Makespan을 줄이는 방향으로 부품 투입순서를 결정한다. (4) Routing은 상호 보완적인 것과 각 기계의 Work Load Balancing을 고려해서 선택한다. 그리고 이 MPS에 대한 결과를 반복 시행한다.

먼저 몇개의 기호를 정의하면,

(1) R_{ik} : 부품 i의 k번째 Routing Vector

(2) N_i : Batch당 생산될 부품 i의 양

(3) $(n_1, n_2, \cdots, n_l)$ = Minimal Part Set, n과 N의 관계는 다음과 같다.

$G=(N_1, N_2, \cdots, N_l)$의 최대 공약수

$(N_1, N_2, \cdots, N_l)=(n_1, n_2, \cdots, n_l)\times G$

(4) R : Routing Matrix.(작업이 없는 것은 Zero로 둔다)

$$R=\begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_l \end{bmatrix} \quad \text{where } R_i=\begin{bmatrix} R_{i1} \\ R_{i2} \\ \vdots \\ R_{ia_i} \end{bmatrix}$$

(5) T : Processing Time Matrix

R Matrix Element를 가공시간으로 바꾸어서 n_i들로 곱한 것

$$T=\begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_l \end{bmatrix} \quad \text{where } R_i=\begin{bmatrix} T_{i1} \\ T_{i2} \\ \vdots \\ T_{ia_i} \end{bmatrix}$$

(6) P_{ikj} : 부품 i의 Routing k의 j번째 작업의 작업시간

(7) a_i : 부품 i의 대체 Routing의 수

(8) t_{ijk} : $n_i \times P_{ikj}$

(9) τ_{ik} : 부품 i의 k번째 Routing의 총 이동시간

(10) U : Machine Processing Time Matrix

T Matrix에서 Routing별로 Element의 순서를 바꾸어줌으로써 기계별 작업시간을 나타낸 것.

4-2. 알고리즘 I

— 부품 투입순서 결정

Step 1. 각 부품의 평균 작업시간을 가장 큰 평균 작업시간을 가진 부품에서 빼 그 차이를 vector로 나타냄

$$C_i=\frac{\Sigma\Sigma t_{ikj}}{a_i} \text{ for all i}$$

$C=\max C_i$

$A=(C-C_1, C-C_2, \cdots, C-C_l)$

Step 2. 각 부품의 평균 이동시간을 가장 긴 평균 이동시간의 부품에서 빼, 그 차이를 vector로 나

타냄.

$$S_i = \frac{\sum_k \tau_{ik}}{a_i} \text{ for all } i$$

$S = \max S_i$

$B = (S - S_1,\ S - S_2, \cdots,\ S - S_i)$

Step 3. 위에서 구한 vector A와 B에 가중치 a와 b를 곱해서 가장 작은 것부터 차례로 시스템에 투입함.

$W = a \times A + b \times B$

where a, b는 가중치

— Routing 조정

Step 4. 평균 작업시간과 이동시간의 합이 가장 큰 부품 i의 Routing은 대체공정 중에서 작업시간과 이동시간의 합이 가장 작은 것 선택(Tie가 발생시, 가공시간이 작은 것부터 선택)

Step 5. 다음에 투입될 부품의 Routing은 첫번째 부품의 Routing vector에서 두번째 부품의 Routing vector를 빼서, 비영인 Element는 1로, 영인 Element는 0으로 둔다.

이것에 임의의 가중치 vector C를 내적하여 이것이 가장 큰 것을 선택(Tie면 가공시간의 합이 작은 것부터 선택)

Step 6. 위의 과정을 MPS에 있는 모든 부품의 Routing이 결정될 때까지 반복함.

— Conflict 해결 및 해의 개선

Step 7. 각 기계는 FCFS의 규칙에 따라 수행(Tie면 Largest Remaining Processing Time Rule을 따름)

Step 8. 위의 결과에서 CT(Cycle Time)에 영향을 미친 모든 부품에 대해서 가공시간이 짧은 부품 순서로 Routing을 선택하면서 결과를 비교하여 해를 개선함(한번 선택된 Routing에 대해서 반복하여 선택하지 않음)

Step 9. 해가 개선되지 않으면, STOP

4-3. 알고리즘 II

알고리즘 I에서의 Step 4—Step 6을 Workload Balancing을 위해 다음과 같이 수행한다.

Step 4. 첫번째 투입되는 부품에 대해서 기계들의 평균 가공시간으로부터 차이의 합과 총 이동시간의 합이 가장 작은 것을 선택(Tie면 가공시간이 짧은 것 선택)

Step 5. 다음 투입되는 부품의 Routing은 첫번째 부품의 가공시간이 누적된 것에서 각 기계의 평균 가공시간으로부터 차이의 합과 총 이동시간의 합이 가장 작은 것을 선택.

Step 6. MPS에 있는 모든 부품에 대해서 Routing이 선택될 때까지 위의 과정을 반복함.

4-4. 예 제

예제 1

- 기계수 : M=6
- 부품 종류의 수 : L=4
- 부품 종류당 생산량 : (10, 10, 10, 10)
- Minimal Part Set : (1, 1, 1, 1)

$$R = \begin{bmatrix} \begin{matrix}1&2&3&4&5&6\\1&4&3&2&5&6\\1&3&2&5&4&6\end{matrix} & R_1 \\ \begin{matrix}1&5&2&4&3&6\\1&3&4&2&5&6\\1&4&2&5&3&6\end{matrix} & R_2 \\ \begin{matrix}1&3&2&5&4&6\\1&2&4&5&3&6\\1&4&3&5&2&6\end{matrix} & R_3 \\ \begin{matrix}1&3&2&4&5&6\\1&4&2&3&5&6\\1&4&5&3&2&6\end{matrix} & R_4 \end{bmatrix} \quad T = \begin{bmatrix} \begin{matrix}2&4&7&5&4&6\\2&4&7&5&4&6\\2&4&7&5&4&6\end{matrix} & T_1 \\ \begin{matrix}2&7&6&4&5&2\\2&7&6&4&5&2\\2&7&6&4&5&2\end{matrix} & T_2 \\ \begin{matrix}2&5&3&3&4&2\\2&5&3&3&4&2\\2&5&3&3&4&2\end{matrix} & T_3 \\ \begin{matrix}2&3&9&5&6&2\\2&3&9&5&6&2\\2&3&9&5&6&2\end{matrix} & T_4 \end{bmatrix}$$

$$D = \begin{bmatrix} \begin{matrix}20\\10\\10\end{matrix} & D_1 \\ \begin{matrix}10\\15\\10\end{matrix} & D_2 \\ \begin{matrix}10\\15\\10\end{matrix} & D_3 \\ \begin{matrix}15\\15\\10\end{matrix} & D_4 \end{bmatrix} \quad U = \begin{bmatrix} \begin{matrix}2&4&7&5&4&2\\2&5&7&4&4&2\\2&7&4&4&5&2\end{matrix} & U_1 \\ \begin{matrix}2&6&5&4&7&2\\2&4&7&6&5&2\\2&6&5&7&4&2\end{matrix} & U_2 \\ \begin{matrix}2&3&5&4&3&2\\2&5&4&3&3&2\\2&4&3&5&3&2\end{matrix} & U_3 \\ \begin{matrix}2&9&3&5&6&2\\2&9&5&3&6&2\\2&6&5&3&9&2\end{matrix} & U_4 \end{bmatrix}$$

기계간 이동 소요시간

	1	2	3	4	5	6
1	0	4	3	2	1	1
2	1	0	4	3	2	1
3	2	1	0	4	3	2
4	3	2	1	0	4	3
5	4	3	2	1	0	4
6	5	4	3	2	1	0

위의 문제를 Algorithm I을 적용해서 풀면

Step 1에서 A=(3, 1, 8, 0)

Step 2에서 B=(0, 1.7, 1.7, 0)

Step 3에서 a=1, b=2로 두면 W=(3, 4.4, 11.4, 0)

Releasing 순서는 4−1−2−3이다.

Step 4에서 부품 4의 세번째 Routing이 선택된다.

R_{43}=(1, 4, 5, 3, 2, 6)

Step 5에서 부품 2의 Routing별 계산결과는

(0, 1, 1, 1, 1, 0)

(0, 1, 1, 1, 1, 0)

(0, 0, 1, 1, 1, 0)

C=(1, 1, 2, 1, 1, 1)로 두고 곱한 결과 (5, 5, 4)이다.

여기서 총 소요시간이 짧은 첫번째 Routing Vector

R_{21}=(1, 5, 2, 4, 3, 6)이 선택된다.

Step 6에서 위의 과정을 반복한 결과

R_{12}=(1, 4, 3, 2, 5, 6)

R_{32}=(1, 2, 4, 5, 3, 6)이 선택된다.

Step 8−9에서 위의 계산결과 개선되지 않음.

이것을 Loading한 결과 MPS의 Cycle Time은 28이다.

알고리즘 I과 II를 적용시킨 결과와 예제가 있는 Nakamura 등의 알고리즘을 적용시킨 결과를 Gantt Chart로 표시하면 [그림 1]과 같다.

예제 2

- 기계수 : M=4
- 부품종류의 수 : L=5
- 부품 종류당 생산량 : (100, 100, 100, 100, 100)
- Minimal Part Set : (1, 1, 1, 1, 1)

$$R=\begin{bmatrix}1&2&2&4\\1&2&3&4\\1&3&2&4\\1&3&3&4\end{bmatrix}R_1\ \begin{bmatrix}1&2&2&3\\1&2&3&4\\1&3&2&4\\1&3&3&4\end{bmatrix}R_2\ \begin{bmatrix}1&2&2&4\\1&2&3&4\\1&3&2&4\\1&3&3&4\end{bmatrix}R_3\ \begin{bmatrix}1&2&4&0\\1&3&4&0\\1&2&4&0\\1&3&4&0\end{bmatrix}R_4\ R_5$$

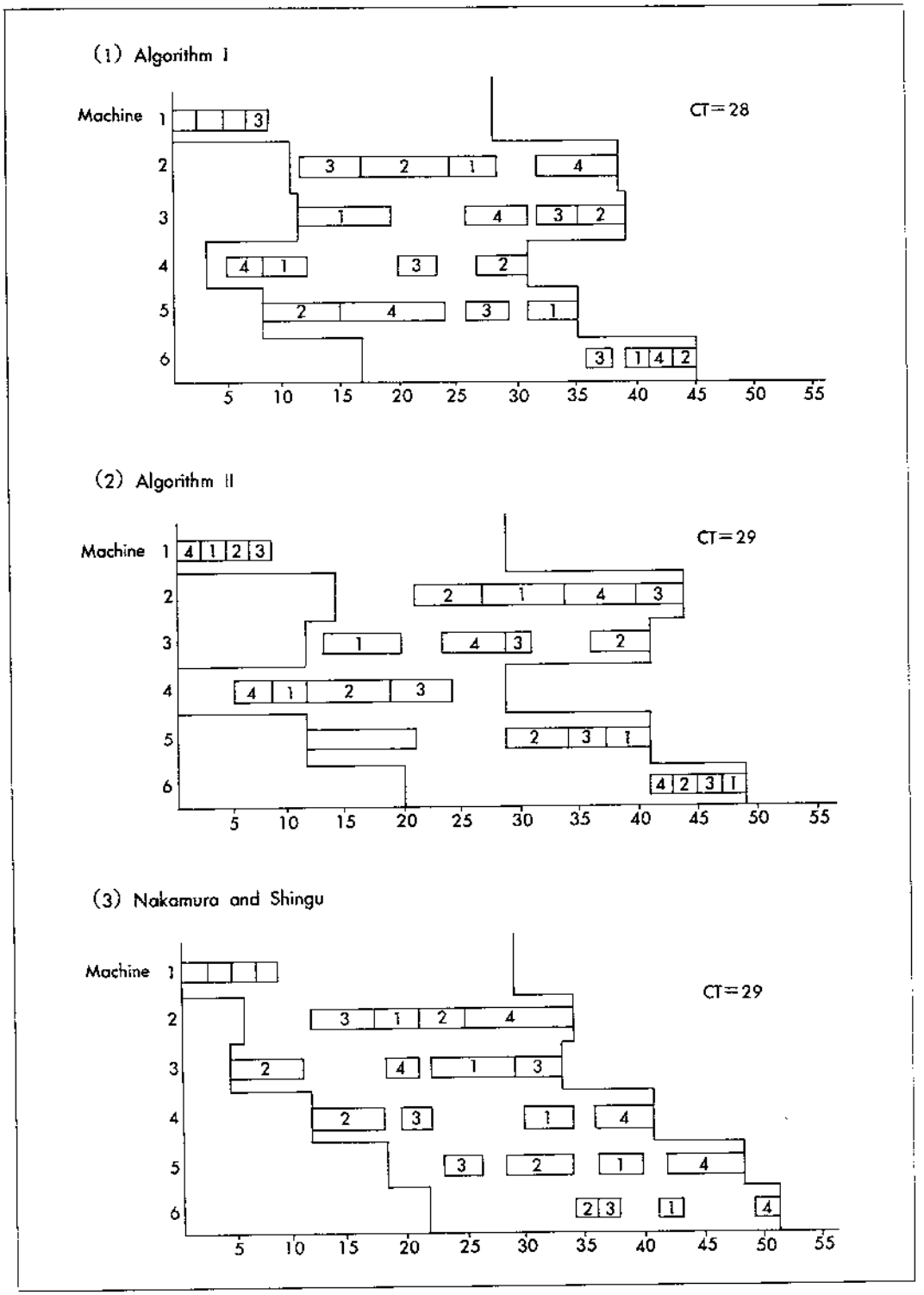
$$T=\begin{bmatrix}10&30&5&10\\10&30&10&10\\10&35&5&10\\10&35&10&10\end{bmatrix}T_1\ \begin{bmatrix}12&10&25&12\\12&10&30&12\\12&10&25&12\\12&10&30&12\end{bmatrix}T_2\ \begin{bmatrix}8&15&15&8\\8&15&20&8\\8&20&15&8\\8&20&20&8\end{bmatrix}T_3\ \begin{bmatrix}10&15&10&0\\10&25&10&0\end{bmatrix}T_4\ \begin{bmatrix}10&20&10&0\\10&25&10&0\end{bmatrix}T_5$$

$$D=\begin{bmatrix}4\\4\\8\\4\end{bmatrix}D_1\ \begin{bmatrix}4\\4\\8\\4\end{bmatrix}D_2\ \begin{bmatrix}4\\4\\8\\4\end{bmatrix}D_3\ \begin{bmatrix}4\\4\end{bmatrix}D_4\ \begin{bmatrix}4\\4\end{bmatrix}D_5$$

$$U=\begin{bmatrix}10&35&0&10\\10&30&10&10\\10&5&35&10\\10&0&45&10\end{bmatrix}U_1\ \begin{bmatrix}12&35&0&12\\12&10&30&12\\12&25&10&12\\12&0&40&12\end{bmatrix}U_2\ \begin{bmatrix}8&30&0&8\\8&15&20&8\\8&0&40&8\\8&0&40&8\end{bmatrix}U_3\ \begin{bmatrix}10&15&0&10\\10&0&25&10\end{bmatrix}U_4\ \begin{bmatrix}10&20&0&10\\10&0&25&10\end{bmatrix}U_5$$

기계간 이동 소요시간

	1	2	3	4
1	0	1	3	1
2	1	0	2	3
3	3	2	0	1
4	1	3	1	0



[그림 1] 알고리즘 I, II와 Nakamura 등의 알고리즘과의 비교 1.

Algorithm I을 적용하면

A=(1.5, 0, 9.5, 21.5, 19)

B=(0, 0, 0, 1, 1)

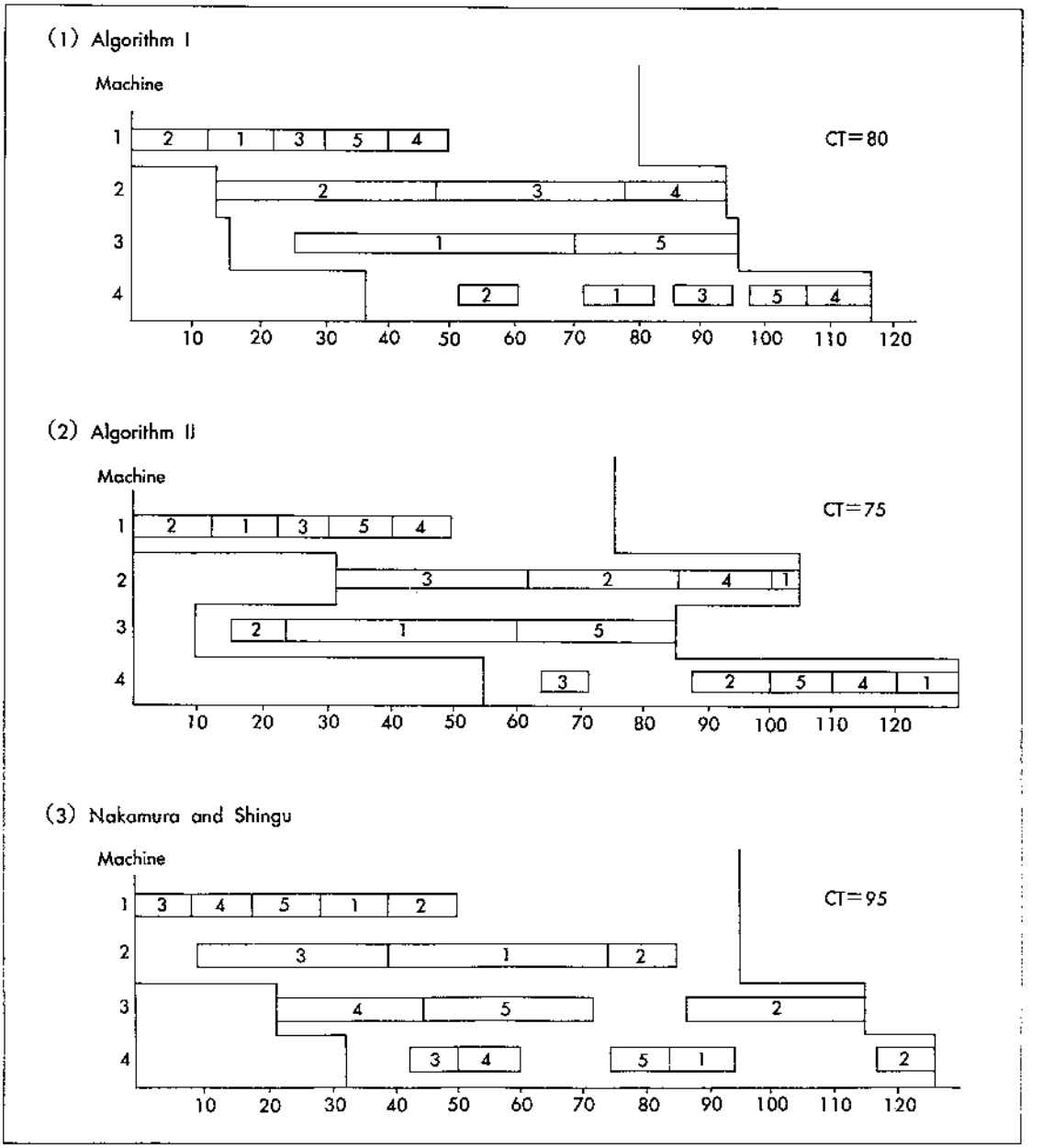
a=1, b=1, W=(1.5, 0, 9.5, 22.5, 20)

Releasing 순서는 2-1-3-5-4이다.

각 부품의 Routing은 C=(1, 2, 2, 1)에서

2(1, 2, 2, 4), 1(1, 3, 3, 4), 3(1, 2, 2, 4),

5(1, 3, 4), 4(1, 2, 4)이 선택되며



[그림 2] 알고리즘 I, II와 Nakamura 등의 알고리즘과의 비교 2.

Cycle Time은 80이다.

알고리즘 II를 적용시키면
각 부품의 Routing은
2(1, 3, 2, 4), 1(1, 3, 2, 4), 3(1, 2, 2, 4),
5(1, 2, 4), 4(1, 3, 4)이 선택되며
Cycle Time은 75이다.

Nakamura와 Shingu의 알고리즘을 적용하면
각 부품의 Routing은
2(1, 2, 3, 4), 1(1, 2, 2, 4), 3(1, 2, 2, 4),
5(1, 3, 4), 4(1, 3, 4)이며
Cycle Time은 95이다.

위의 결과를 Gantt Chart로 표시하면 [그림 2]와 같다.

5. 결론 및 앞으로 연구과제

본 연구에서 제시한 최적해를 제공하는 새로운 정수계획법 모형은 기존의 연구에서 고려하지 못한 Buffer Capacity 및 대체 공정을 고려하였으나 실제 문제를 받아들일 만한 시간에 해결하지는 못하였다 (4개 부품, 21개 작업에 대해 MPSX-MIP 370 package에서 CPU Time 2시간내에 해를 구하지 못하였다).

실시간 문제해결을 위해 제시된 두개의 Heuristic 알고리즘은 매우 간단해서 적용성이 높으며 동적인 상황에 대처하기가 쉽다. 그리고 기존의 알고리즘보다는 좋은 결과를 제공하였으나 최적해를 보장하지는 못한다.

따라서 최적해를 실제 시간에 구하는 방법에 관한 많은 연구가 있어야 할 것으로 생각되며, 특히 Linear Programming을 이용해서 해를 구하는 방법에 대한 많은 연구가 기대된다.

참고문헌

[1] A. Kusiak, "Scheduling Flexible Machining and Assembly Systems," Proc. of the Second ORSA/TIMS Conference on FMS, 1986.

[2] A.J. Van Looveren, L.F. Gelders and L.N. Van Wassenhove, "A Review of FMS Planning Models," Modelling and Design of Flexible Manufacturing Systems, Edited by A. Kusiak, 1986.

[3] B. Pourbabai, "A Production Planning and Scheduling Model for Flexible Manufacturing," Proc. of the Second ORSA/TIMS Conference on FMS, 1986.

[4] E.M. Malstrom and Richard H. Choi, "Evaluation of Traditional Work Scheduling Rules in a FMS with a Physical Simulator," J. of Manufacturing Systems, Vol. 7, No. 1, 1988.

[5] J. Kimemia and S.B. Gershwin, "Flow Optimization in FMS," Int. J. Prod. Res., Vol. 23, No. 1, 1985.

[6] J. Slomp, G.J.C. Gacman and W.M. Nawin, "Quasi On-Line Scheduling Procedures to FMS," Int. J. Prod. Res., Vol. 26, No. 4, 1988.

[7] K. Iwata, Y. Murotsu and F. Oba, "Problems for Job-Shop Type Machining Systems with Alternative Machine Tools," Annals of the CIRP, Vol. 29, No. 1, 1980.

[8] K. Iwata, K. Yasuda, A. Murotsu and F. Oba, "Production Scheduling of FMS," Annals of the CIRP, Vol. 31, No. 1, 1982.

[9] K.E. Stecke and J.J. Solberg, "Loading and Control Polices for a FMS," Int, J. Prod. Res., Vol. 19, No. 5, 1981.

[10] N. Nakamura and T. Shingu, "Scheduling of FMS," Toward the Factory of the Future, 1985.

[11] P. Afentakis, "Maximum Throughput in Flexible Manufacturing Systems," Proc. of the Second ORSA/TIMS Conference on FMS, 1986.

[12] S.C. Sarin and E.C. Dar-El, "Approaches to the Scheduling Problem in FMS," The 1984 Fall I.E. Conference Proceedings.