Journal of the Korean Institute of Industrial Engineers
Vol. 25, No. 2, pp. 254-265, June 1999.

이종 병렬기계를 가진 2단계 혼합흐름생산시스템의 일정계획

이지수 · 박순혁

금오공과대학교 산업공학과

Scheduling Heuristics for a Two-Stage Hybrid Flowshop with Nonidentical Parallel Machines

Ji-Soo Lee · Soon-Hyuk Park

We consider two stage hybrid flowshop scheduling problem when there are two non-identical parallel machines at the first stage, and only one machine at the second stage. Several well-known sequence-first allocate-second heuristics are considered first. We then propose an allocate-first sequence-second heuristic to find minimum makespan schedule. The effectiveness of the proposed heuristic algorithm in finding a minimum makespan schedule is empirically evaluated by comparing with easily computable lower bound. The proposed heuristic algorithm as well as the existing heuristics are evaluated by simulation in four cases which have different processing time distribution, and it is found that the proposed algorithm is more effective in every case.

1. 서 론

병렬처리기계(parallel machines)는 동일 작업을 처리할 수 있는 기계가 하나 이상 있으나 실제 작업을 처리할 때 하나의 기계에서만 그 작업을 처리할 수 있는 상황을 말한다. 이종 처리기계(non-identical machines)는 어느 작업장에서 동일 작업을 처리할 수 있는 기계가 하나 이상 존재하지만, 능력이나 용량이 서로 다른 기계들을 말한다. 병렬처리기계의 경우, 이종 처리기계의 사용이 제조산업과 서비스산업에서는 매우 일반적인 상황이다. 타자수(typist) 집단의 경우를 예로 든다면, 어떤 타자수라도 문서를 작성할 수 있지만 실제는 능력이 다른 여러 타자수들 중 한 명의 타자수에게만 작업이 할당된다. 다른 예로서, 동종 작업을 처리하는 작업장에 놓인 동일 기능의 기계가 모델에 따라, 또는 기계의 노후화 정도에 따라 효율성의 차이를 보이는 경우를 생각할 수 있다. 이처럼 병렬처리기계는 비록 그들이 일반적으로 비슷한 유형이라 할지라도 생산률이나 처리시간 등이 다를 수 있는데 이를 이종 병렬처리기계(nonidentical parallel machines)라 한다(Randhawa and Kuo, 1997). 현실 산업사회에서 이종의 병렬기계가 배치된 경우가 많으나 이에 대한 기존의 연구가 거의 이루어지지 않고 있다. Petrov (1968)는 문제의 복잡성으로 인해 대부분의 연구자들이 혼합흐름생산시스템의 문제를 무시하고 있다고 지적하고 있다.

이종 병렬처리기계에 대한 주요 연구는 Guinet (1991), Randhawa and Smith (1995), Narashimhan and Panwalker (1984) 등에 의해 수행되었다.

Guinet (1991)은 일련의 병렬처리기계 군으로 이루어진 직물생산 시스템의 일정계획 문제에 대한 연구에서 각 작업장을 독립된 문제로 보고 선형계획법을 사용하여 이종 병렬처리기계로 구성된 1개 작업장의 평균흐름시간(mean flow time)을 최소화하는 일정계획을 작성한 다음 이들을 다시 결합하는 방안을 제시하였다.

Randhawa and Smith (1995)는 이종 병렬처리기계로 구성된 단일 작업장 문제의 일정계획 효율에 영향을 미치는 요소(factor)를 찾기 위한 모의실험을 행하여, 시스템 부하도(system loading)와 작업준비시간이 일정계획 효율에 중요한 영향을 미치며, 작업배정규칙(assignment rules)은 비교적 영향을 미치지 않는다는 사실을 발견하였다.

Narashimhan and Panwalker (1984)는 두 번째 작업장에 이종 병렬처리기계가 배치된 2단계 혼합흐름생산시스템에서 총 기계 유휴시간(total machine idle time) 및 총 작업대기시간(total waiting time)을 최소화하는 CMD(cumulative minimum deviation)규칙을 제안하였다.

본 연구는 두 개의 작업장으로 이루어진 혼합흐름생산시스

템 문제 중에서 첫 번째 작업장이 두 대의 이종 병렬처리기계가 배치되고, 두 번째 작업장에 한 대의 기계가 배치된 시스템에 대한 일정계획 작성문제를 고려한다. 이러한 사업장의 대표적인 형태로 플라스틱 사출공장을 예로 들 수 있다. 플라스틱 사출공정은 여러 대의 성능이 다른 사출기에서 여러 종류의 사출품들이 만들어지고, 이 사출품들은 한 개의 도장라인에 투입되어 도장되게 된다. 본 논문에서는 첫 번째 작업장에 두 대의 이종 병렬기계가 배치된 2단계 흐름생산시스템의 일정계획문제 중 전체작업완료시간(makespan)을 최소화시키는 휴리스틱 알고리즘(heuristic algorithm)을 제안한다.

이러한 시스템의 일정계획을 작성하기 위해서는 고려해야 할 두 가지 규칙이 발생한다. 우선 작업장 1에서 성능이 다른 두 기계에 작업을 어떻게 할당할 것인지를 결정하는 할당규칙(assign rule)이 있고, 각 작업을 어떠한 순서로 처리할 지 결정하는 처리순서결정규칙(sequencing rule)이 있다. 두 작업장 문제에 관한 이전의 많은 연구에서 먼저 처리순서를 결정하고, 그 다음 할당규칙을 사용하는 방법(list heuristic)을 사용하였다. 그러나 두 기계에 할당된 작업들은 처리시간의 차이에 의해 두 번째 작업장에서 처음에 의도하였던 대로 작업을 처리할 수 없게 된다. 본 연구는 먼저 작업의 할당순서를 정하여 할당규칙을 사용하고, 다음으로 할당된 작업의 처리순서를 결정(allocate first - sequence second heuristic approach) 함으로써 부분적인 처리순서를 유지하여 시스템의 전체작업완료시간(makespan)을 최소화하고자 하였다.

본 논문의 구성을 요약하자면, 2절에서는 본 논문이 고려하고 있는 혼합흐름생산시스템에 대한 설명을 하고 있다. 3절은 선작업처리순서결정-후작업할당의 문제를 설명하고, 4절은 선작업할당-후작업처리순서결정 휴리스틱 알고리즘을 제안한다. 5절에서는 3절과 4절의 알고리즘들을 비교하여 4절에서 제안한 휴리스틱 알고리즘이 전체작업완료시간(makespan)에 대해 성능이 우수함을 보였다.

2. 시스템 정의 및 연구방법

2.1 시스템의 정의 및 사용기호

일정계획문제의 분류를 위한 Lawler *et al.* (1982)의 표기법에 따르면, 이 논문에서 다루는 문제는 F2 | M_1=2, M_2=1| Cmax 로 나타낼 수 있다 <그림 1>. Lawler 등은 일정계획문제를 $\alpha|\beta|\gamma$의 형태로 표기하고 있다. α = F2 는 2개의 작업장을 갖는 flowshop을 나타내며, $\beta = \{M_1, M_2\}$는 각 작업장에 존재하는 기계의 수를 의미한다. 여기서 작업장 1에는 동일한 작업을 처리할 수 있으나 성능이 다른 두 대의 이종 병렬기계가 존재하고 (M_1=2), 작업장 2는 기계가 한 대(M_2=1)인 시스템을 고려한다. γ = Cmax 최적화 척도로서 전체작업완료시간(makespan)을 최소화하는 문제를 다룬다.

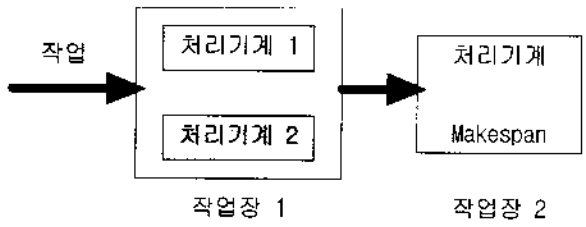

그림 1. F2 | M_1=2, M_2=1 | Cmax 시스템.

본 논문은 다음과 같은 가정을 하고 첫 번째 작업장에 이종 병렬처리기계가 배치된 2단계 혼합흐름생산시스템 문제를 다룬다.

- 작업장이 2개인 혼합흐름생산시스템으로 구성되어 있으며 각 작업은 작업장 1, 작업장 2의 순서로 처리된다.
- 각 작업장은 M_j(j=1, 2.)개의 기계들로 구성되어 있다 (M_1=2, M_2=1). 특히 작업장 1에 배치된 기계들 간에는 동일한 작업을 처리하는데 있어서 성능의 차이가 존재한다.
- 작업장 1에서 어느 한 작업은 작업장 1 내의 모든 기계에서 처리될 수 있으나 오직 한 기계에 할당되어 처리된다.
- 작업들은 각각에 대해 독립적이며, 작업들 간에 처리 중 순서 바꿈(preemption)은 허용되지 않는다.
- 작업장 1과 작업장 2 사이에는 작업장 1에서 처리된 작업들을 임시로 저장할 수 있는 저장공간(buffer)이 있으며, 용량은 무한하다.

이 문제를 해결하는데 사용되는 기호들은 다음과 같다.

n : 처리해야 할 전체 작업의 수
i : 처리할 작업을 나타내는 첨자 (i=1, 2, ..., n).
j : 작업장을 나타내는 첨자 (j=1,2).
M_j : 작업장 j에 배치된 기계의 수 (M_1=2, M_2=1).
m_{jk} : 작업장 j에 배치된 k번째 기계.
P_{i1k} : 작업장 1의 k 번째 기계에서 작업 i를 처리하는데 소요되는 시간.
P_{i2} : 작업장 2에서 작업 i를 처리하는데 소요되는 시간.
T_{1k} : 작업장 1의 k 번째 기계에서 모든 작업을 처리하는데 소요되는 시간.

$$T_{1k} = \sum_{i=1}^{n} P_{i1k}, \quad k = 1, 2.$$

N_{ik} : 작업장 1의 k 번째 기계에 할당된 작업의 수.

R_i : i 작업에 대한 기계 m_{11}과 기계 m_{12}의 처리시간의 비.

$$R_i = P_{i11} / P_{i12}, \qquad i = 1, 2, \cdots, n.$$

2.2 연구방법 및 모의실험 상황 생성

본 연구에 있어서 일정계획 문제는 두 가지 경우를 고려하고 있다. 즉, 먼저 작업처리순서를 결정하고 그 순서대로 작업을 할당하는 선작업처리순서결정-후작업할당(sequence first-allocate second) 방법과, 먼저 작업을 할당하고 그에 따른 처리순서를 결정하는 선작업할당-후작업처리순서결정(allocate first-sequence second) 방법이다. 지금까지 이 두 가지 경우 중 어떠한 방법이 더 나은지에 대한 연구결과는 없다. 단지 대부분의 연구에서 먼저 작업순서를 결정하고 결정된 순서에 따라 기계에 작업을 할당하는 선작업처리순서결정-후작업할당 방법을 사용하고 있다. 예외로 작업장 1과 작업장 2의 기계 대수가 동일한 경우 먼저 작업을 할당한 후 각 기계에서의 작업순서를 결정하는 선작업할당-후작업처리순서결정 방법을 Langston (1987)과 Sriskandarajah and Sethi (1989)가 채택하고 있다. 본 연구에서는 몇 가지 잘 알려진 선작업처리순서결정-후작업할당 방법에 의한 결과와 새롭게 제안하는 선작업할당-후작업처리순서결정 방법을 사용한 결과를 비교하고자 하였다. 선작업할당-후작업처리순서결정을 채택한 이유는 처리순서 결정 단계에서 아무리 최적으로 순서가 결정되었다 할지라도, 그 순서대로 마지막 작업장에서 처리된다고 보장할 수 없으며, 경우에 따라서는 의도와는 다르게 기존의 일반적인 방법보다 못한 결과를 초래하게 되기 때문이다<그림 2>. 그러나 할당을 먼저 한 후에 작업처리순서를 결정하면 전체적인 순서는 혼합된 형태로 나타나지만 부분적이나마 작업의 처리순서를 유지하게 된다<그림 3>. 일정계획 문제의 성능평가 기준에 영향을 미칠 수 있는 중요한 요인으로는 1) 작업처리순서결정방법, 2) 작업할당방법, 3) 생산시스템의 구성, 4) 작업장 1과 작업장 2의 작업처리시간 등 4 가지를 생각할 수 있다. 작업처리순서결정방법과 작업할당방법이 성능평가 기준에 영향을 미친다는 사실은 지금까지의 연구결과로 충분히 알 수 있다. 생산 시스템의 구성이란 작업장 1과 작업장 2의 기계 대수를 의미하며, 작업장 1과 작업장 2의 작업시간 비율에 따라 작업순서결정방법이 달라진다는 것은 Gupta and Tunc (1991)가 보여준 바 있다. 본 연구에서는 성능평가 기준에 영향을 미칠 수 있는 요인으로 작업처리순서 방법, 작업할당방법, 그리고 각 작업장의 처리시간을 고려하였으며, 성능평가 기준으로는 흐름생산시스템의 일정계획 연구에서 전통적으로 많이 사용되고 있는 전체작업완료시간(make- span)을 사용하였다.

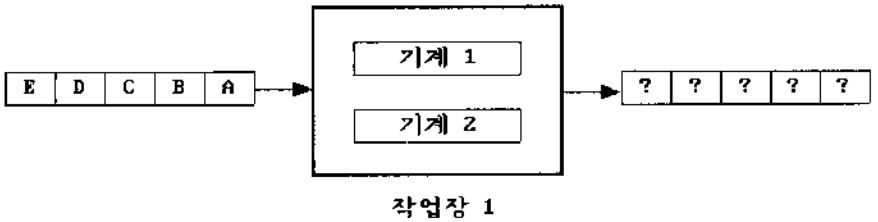

그림 2. 선작업처리순서결정-후작업할당의 경우.

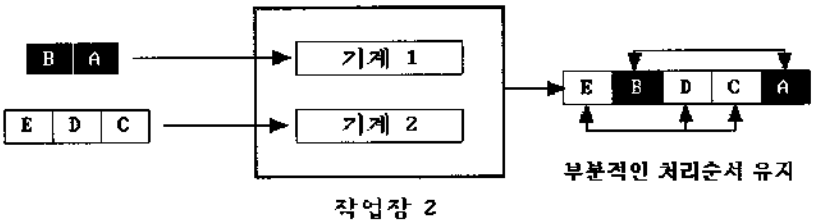

그림 3. 선작업할당-후작업처리순서결정의 경우.

이종 병렬기계 상황을 반영하기 위해, 본 논문에서 수행하는 모든 모의실험에서는 첫 번째 작업장(j = 1)의 서로 다른 기계(k = 1, 2)에서 동일한 작업(동일 i)을 처리하더라도 처리시간(P_{i11}, P_{i12})을 각각 독립적으로 발생시켰으며, 작업장의 처리시간이 성능평가 기준에 영향을 미치는 상태를 파악하기 위해 작업처리시간 (P_{i11}, P_{i12}, P_{i2})의 분포를 다양하게 바꾸어 <표 1>과 같이 4 가지 경우를 고려하였다. Case 2를 제외하고는 작업

표 1. 모의 실험에 사용한 처리시간 분포의 특성

	Line balancing 상태	P_{i1}과 P_{i2}의 모수 특성		P_{i1}	P_{i2}	P_{i2}
		평균	표준편차			
Case 1	non-dominant	동일	동일	U(1,39)	U(1,39)	U(1,39)
Case 2	dominant	동일	동일	U(1,39)	U(1,39)	U(1,19)
Case 3	non-dominant	다름	다름	U(1,39)	U(1,79)	U(1,59)
Case 4	non-dominant	다름	동일	U(1,39)	U(21,59)	U(1,59)

* U(a,b)는 uniform distribution in the range (a,b)를 의미]

장 1, 2의 작업처리에 있어서 균형(line balancing)이 이루어지도록 각 기계에서의 작업처리시간 분포를 결정하였다(Gupta and Tunc (1991)은 Case 2의 경우를 dominant data set이라고 칭하고, Case 1, 3, 4의 경우를 non-dominant data set이라고 칭하였음).

3. 선작업처리순서결정 - 후작업할당

3.1 List Heuristic

m개의 동종 처리기계가 배치된 환경에 대한 일반적인 형태(list scheduling 휴리스틱, Graham(1966))는 다음과 같다. 각각의 작업들은 지정된 순서에 따라 처리되며, 어떤 순서를 갖는 작업은 가장 먼저 사용 가능한 기계에 할당된다. 즉, 이전에 할당된 작업을 끝마친 기계에 할당한다. 이러한 규칙을 최우선 사용가능기계(first available machine : FAM) 할당규칙이라 한다. List scheduling 휴리스틱은 처리순서에 따라 상이한 결과를 나타내게 되는데 본 연구에서는 처리순서 결정 규칙으로 최소 처리시간(SPT) 우선규칙과 최대 처리시간(LPT) 우선규칙을 사용하였다.

고려되는 시스템이 두 개의 작업장으로 구성되어 있기 때문에 두 개의 규칙을 각각의 작업장에 적용하되, 첫 번째 작업장의 두 기계 중 각 작업에 대한 처리시간이 짧은 기계의 처리시간을 사용하여 SPT와 LPT를 적용하여 각각 SPT1, LPT1이라 하였고, 두 번째 작업장에서의 각 작업의 처리시간에 대해 SPT와 LPT를 적용하여 각각 SPT2, LPT2라 하였다.

이상의 4가지 우선순위규칙으로 처리순서가 결정된 작업들은 작업장 1의 기계들 중에서 최우선 사용가능기계(FAM)에 할당되며, 두 번째 작업장에서는 FCFS 규칙에 따라 작업을 처리하게 된다 <그림 4>.

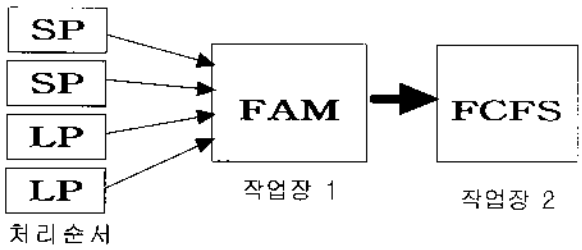

그림 4. List Heuristic에 의한 처리순서.

3.2 모의실험 및 평가

앞에서 설명한 방법대로 각 처리순서 결정 규칙에 대해 처리할 작업의 수를 최소 5개에서 최대 100개까지 20가지를 선택하고, 2절에서 설명한 4 가지 case의 처리시간 분포 조합 각각에 대해 각 작업수별로 20개씩의 문제를 생성시켜 총 1600회의 모의실험을 수행하였다.

본 절의 목적은 선택된 4종류의 list heuristic의 상대적 우열을 가리는 것이므로, 생성된 문제에 4종류의 heuristic을 각각 적용하여 나온 4가지 결과의 전체작업완료시간(makespan) 중 최소값(이하 '최소 전체작업완료시간'이라 함)을 어떤 heuristic 많이 발생시키는지 관찰하였고, 그 결과를 <표 2> ~ <표 5>에 정리하였다. 첫 번째 칸에는 처리작업의 수를, 나머지 칸에는 각각의 작업처리순서 결정 규칙을 사용하였을 때 4가지 규칙 중 최소 전체작업완료시간을 만들어 낸 횟수를 기록하였다. 각 열의 합이 20 이상이 되는 것은 하나 이상의 규칙이 최소 전체작업완료시간을 만들어 냈음을 의미한다.

<표 2> ~ <표 5>에서 알 수 있듯이 SPT1이 모든 처리시간 분포의 경우에 대해서 우수함을 알 수 있다. 다만 Case 2의 경우 LPT2가 SPT1과 비교적 비슷한 결과를 보여주고 있다. 그 이유는 Case 2의 데이터가 dominant data set이라는 작업의 처리시간 분포의 특성에서 찾을 수 있다. 작업장 2에서의 처리시간의 평균과 표준편차 모두가 작업장 1의 두 기계에 대한 처리시간의 평균과 표준편차의 절반이기 때문에, 작업장 2에는 작업장 1의 처리시간보다 작은 처리시간이 발생하게 되고, 결국 작업장 1로부터 작업장 2로 투입되는 시간 간격과 작업장 2에서의 처리시간이 비슷하게 되어 주어진 문제가 각 작업장에 한 대의 기계가 설치된 2단계 흐름생산시스템과 유사하게 되기 때문이라고 생각된다.

표 2. List heuristic을 Case 1에 적용한 결과

작업수	최소 전체작업완료시간 발생빈도			
	SPT1	SPT2	LPT1	LPT2
5	13	2	2	8
10	14	0	3	10
15	17	0	0	4
20	15	0	2	7
25	18	0	1	4
30	18	0	0	3
35	16	0	0	6
40	16	0	1	4
45	15	0	0	6
50	15	0	0	6
55	15	0	1	6
60	17	0	0	3
65	19	0	0	3
70	17	0	2	5
75	18	0	1	2
80	16	0	0	5
85	16	0	1	6
90	15	0	1	5
95	19	0	0	2
100	17	0	0	4
총계	326	2	15	99

표 3. List heuristic을 Case 2에 적용한 결과

작업수	최소 전체작업완료시간 발생빈도			
	SPT1	SPT2	LPT1	LPT2
5	14	3	0	5
10	9	1	0	10
15	9	0	1	10
20	11	0	0	10
25	8	0	0	13
30	9	0	0	1
35	12	0	0	10
40	9	0	0	11
45	12	0	0	10
50	16	0	1	3
55	12	0	1	8
60	8	0	0	12
65	10	0	0	10
70	8	0	1	11
75	11	0	1	8
80	7	0	0	13
85	8	0	0	12
90	14	0	0	6
95	10	0	0	11
100	14	0	0	7
총계	211	4	5	191

표 4. List heuristic을 Case 3에 적용한 결과

작업수	최소 전체작업완료시간 발생빈도			
	SPT1	SPT2	LPT1	LPT2
5	17	1	1	4
10	14	0	0	10
15	16	0	0	7
20	14	0	1	7
25	16	1	0	4
30	16	0	0	5
35	12	0	0	8
40	15	0	0	7
45	17	0	0	3
50	15	0	0	6
55	16	0	1	5
60	17	0	0	4
65	17	0	0	3
70	15	0	0	5
75	19	0	0	1
80	14	0	0	7
85	12	0	1	7
90	15	0	0	6
95	16	0	0	4
100	15	0	0	5
총계	308	2	4	108

표 5. List heuristic을 Case 4에 적용한 결과

작업수	최소 전체작업완료시간 발생빈도			
	SPT1	SPT2	LPT1	LPT2
5	16	5	3	6
10	19	0	2	3
15	17	0	1	3
20	19	0	0	2
25	17	0	0	7
30	18	0	0	5
35	19	0	0	2
40	15	0	2	3
45	15	0	1	6
50	15	0	1	5
55	18	0	1	5
60	18	0	0	2
65	17	0	0	5
70	16	0	1	4
75	16	0	0	7
80	16	0	0	5
85	17	0	0	3
90	16	0	3	2
95	18	0	0	4
100	16	0	0	4
총계	338	5	15	83

4. 선작업할당 - 후작업처리순서결정

4.1 제안 알고리즘

전체작업완료시간(makespan)을 최소화하기 위해 작업장 1에서는 가능한 한 모든 작업을 빨리 완료할 수 있도록 각 기계에 작업을 할당하여야 하며, 작업장 2에서는 기계의 유휴시간(idle time)을 최소화시켜야 한다. 작업장 1에서는 상대적으로 처리시간이 빠른 기계에 작업을 할당함과 동시에 작업장 1에서의 전체작업완료시간을 최소화하고, 작업장 2에서는 유휴시간을 최소화 되도록 작업의 처리순서를 정하여야 한다.

첫 번째 작업장에 두 대의 이종 병렬처리기계가 배치되고, 두 번째 작업장에 한 대의 기계가 배치된 혼합흐름생산시스템에서 전체작업완료시간을 최소화시키는 제안 알고리즘은 다

음과 같다.

Step 1. 작업장 1에서 최소 처리시간을 갖는 작업을 찾아내어 최소 처리시간으로 처리하는 기계에 할당한다. 또한, 할당된 작업은 할당된 기계에서 첫 번째로 처리되도록 한다.

Step 2. 작업장 1의 두 기계에 대해 나머지 작업에 대한 처리시간의 비(R_i)를 계산하여 오름차순으로 정렬한다. 여기서, R_i가 1에 가까우면 두 기계의 작업처리시간에는 큰 차이가 없다는 것을 나타낸다. R_i>1 이면 i 번째 작업을 처리하는데 있어서 m_{12}가 더 빠른 처리 시간을 갖는다는 것을 나타내며, R_i<1 이면 그 반대가 된다.

Step 3. 작업장 1의 기계 1에 할당 할 작업의 수를 결정한다. 작업장 1의 기계 1에 할당 할 작업의 수(N_{11})는

$$N_{11} = \left\lceil \left[1 - \frac{T_{11}}{T_{11}+T_{12}}\right] \times (n-1) \right\rceil$$

가 된다. 여기서, $\lceil x \rceil$ 는 x를 넘지 않는 최대 정수. 따라서 작업장 1의 기계 2에 할당되는 작업의 수는 $N_{12} = n - (N_{11} + 1)$이 된다. 즉, 두 기계에서 각각 모든 작업을 처리할 때 소요되는 시간비에 반비례 되도록 작업을 할당하게 된다. 할당은 Step 1에서 계산된 비율(R_i) 중 낮은 비율을 갖는 작업순서대로 계산된 수(N_{11}과 N_{12}) 만큼의 작업을 기계 1과 기계 2에 할당한다.

Step 4. 각 기계별로 할당된 작업들에 대해 SPT 우선순위규칙을 사용하여 작업처리순서를 결정하여 처리한다.

4.2 수치예제

위에서 설명된 제안 알고리즘과 전체 하한에 대한 예를 보이기 위해, 5개 작업에 대한 처리시간값과 R_i 값을 <표 6>에 정리하였다.

표 6. 작업의 처리시간 및 R_i

기계 \ 작업		$J1$	$J2$	$J3$	$J4$	$J5$
작업장 1	P_{i11}	3	9	35	8	23
	P_{i12}	11	30	32	38	16
	R_i	0.27	0.39	1.09	0.21	1.44
작업장 2	P_{i2}	38	20	16	18	14

Step 1. 작업장 1에서 최소 처리시간을 갖는 작업을 찾아내어 최소 처리시간으로 처리하는 기계에 할당한다. <표 6> 에서 작업장 1에서 최소 처리시간을 갖는 작업은 $J1$으로 m_{11}에서 3의 시간에 처리된다. 따라서 $J1$을 m_{11}에 할당한다.

Step 2. 나머지 작업을 R_i의 오름차순으로 정렬한다. <표 6> 에서 R_i의 오름차순으로 정렬된 결과는 { $J4, J2, J3, J5$ } 이 된다.

Step 3. 작업장 1의 m_{11}에 할당 할 작업의 수를 결정한다.

$$N_{11} = \left\lceil \left[1 - \frac{T_{11}}{T_{11}+T_{12}}\right] \times (n-1) \right\rceil$$

$$= \left\lceil \left[1 - \frac{78}{78+127}\right] \times 4 \right\rceil = \lceil 2.478 \rceil = 2$$

$$N_{12} = 4 - 2 = 2$$

따라서, 기계 m_{11}에는 작업 { $J4, J2$ }를 할당하고, 기계 m_{12}에는 작업 { $J3, J5$ }를 할당한다.

Step 4. 작업장 1의 각 기계에 할당된 작업을 최소 처리시간 우선순위에 따라 작업처리순서를 정한다.
따라서, 기계 m_{11}에서의 처리순서는 { $J1, J4, J2$ }, 기계 m_{12}에서는 { $J5, J3$ }이 된다.

표 7. 할당된 작업의 처리순서

	할당된 작업	$J1$	$J4$	$J2$
m_{11}	처리시간	3	8	9
	처리순서	1	2	3
	할당된 작업	$J3$	$J5$	
m_{12}	처리시간	32	16	
	처리순서	2	1	

아래의 <표 8>과 <그림 5>는 위의 예제에서 결정된 순서로 작업을 처리한 결과를 나타낸 것이다.
따라서, 본 예제의 전체작업완료시간(makespan)은 109 가 된다.

표 8. 제안 알고리즘의 처리결과 (시작시간 | 종료시간)

m_{11}	$J1$ (0 | 3)	$J4$ (3 | 11)	$J2$ (11 | 20)		
m_{12}	$J5$ (0 | 16)	$J3$ (16 | 48)			
S_2	$J1$ (3 | 41)	$J4$ (41 | 59)	$J5$ (59 | 73)	$J2$ (73 | 93)	$J3$ (93 | 109)

4.3 Global Lower Bound on Makespan(전체작업완료시간의 전체하한)

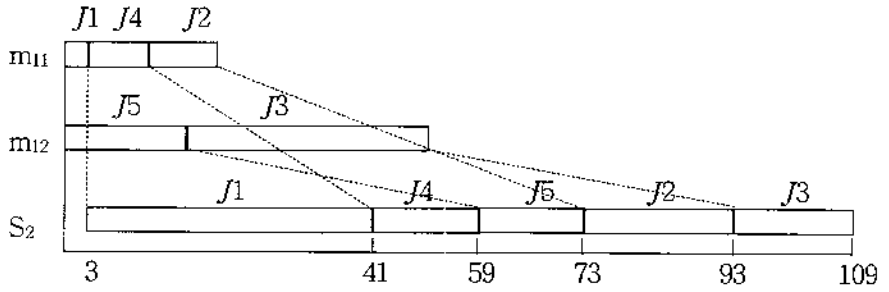

그림 5. 처리결과의 칸트 차트.

제안 알고리즘의 효율을 평가하기 위해, 최적 전체작업완료시간(makespan)값을 찾아야 한다. 그러나 최적 전체작업완료시간을 찾을 수 있는 알고리즘은 존재하지 않는다. Arthanary and Ramamurthy (1971)가 작업장 2에 한 대의 기계가 배치된 문제의 전체작업완료시간에 대한 하한을 제시하였으나, 이들의 하한은 매우 많은 계산을 필요로 한다. 따라서, 제안 휴리스틱 알고리즘의 효율을 평가하기 위하여 간단하고 쉽게 계산할 수 있는 전체하한(global lower bound)을 개발하여 최적 전체작업완료시간을 대신하여 사용하였다.

만약, 두 작업장에 각각 한 대의 기계가 배치되었다고 한다면 이 문제는 Johnson (1954)의 알고리즘을 사용하여 쉽게 계산될 수 있다. 따라서, 작업장 1에서 각 작업의 최소 처리시간을 작업장 1에 배치된 기계의 수로 나누고, 그 값을 Johnson의 알고리즘에 적용하여 계산된 전체작업완료시간을 하한으로 정하였다.

LB1: 모든 i에 대해 $\underset{k=1,2}{\text{Min}}\,(P_{i1k}) / 2$ 를 작업장 1에서의 i번째 작업처리시간으로 하여 Johnson의 알고리즘을 적용하여 계산된 시간.

두 번째 작업장에 한 대의 기계가 배치되었기 때문에 이상적인 경우 작업장 2의 모든 작업의 처리시간의 합과 작업장 1에서의 최소 처리시간을 더하면 가장 단순한 하한을 얻을 수 있다.

$$\text{LB2} = \underset{\text{for all } i,k}{\text{Min}}\,(P_{i1k}) + \sum_{i=1}^{n} P_{i2}$$

또 다른 하한은 작업장 1의 두 기계가 동일한 최소 시간으로 가동되도록 모든 작업을 처리한다. 이때 작업장 1에서 마지막 작업을 처리완료 하였다 하더라도, 작업장 2에서 이 마지막 작업을 처리해야 하므로 하한은 다음과 같이 정의 될 수 있다.

$$\text{LB3} = INT\left\{\frac{\sum_{i=1}^{n}\left[\underset{k=1,2}{\text{Min}}\,(P_{i1k})\right]}{2}\right\} + \underset{i=1,\cdots,2}{\text{Min}}\,(P_{i2})$$

여기서, $INT\{X\}$는 X보다 크거나 같은 최소 정수.

위에서 언급된 하한들을 이용하여, 작업장 1에 이종 병렬기계가 배치된 2단계 혼합흐름생산시스템에서의 전체하한(global lower bound)은 LB = Max(LB1, LB2, LB3)이며 최적 일정계획에 대한 전체작업완료시간(makespan)은 LB보다 작을 수 없다.

4.4 모의실험 및 결과

본 연구에서 제안된 알고리즘의 수행도 평가를 위해 4 가지 case의 처리시간 분포에 대해 처리할 작업의 수를 최소 5개에서 최대 100개까지 20가지 경우를 선택하여 각 20회씩 총 1600(4×20×20)회의 모의실험을 실시하였다.

제안된 알고리즘의 성능을 평가하기 위해, 주어진 문제에 제안 알고리즘을 적용하여 얻어진 전체작업완료시간(makespan)이 전체하한(global lower bound)과 얼마나 차이가 나는가를 나타내는 상대편차(relative deviation)를 사용하였다. 상대편차는 다음과 같이 정의된다.

$$\text{상대편차} = \left(\frac{\text{제안 알고리즘 결과} - \text{전체하한}}{\text{전체하한}}\right) \times 100$$

각 문제의 전체작업완료시간에 대한 전체하한으로부터 계산된 전체작업완료시간이 어느 정도 벗어나 있는지를 평가하기 위해, Gupta and Tunc (1991)가 분석했던 방법과 동일하게 다음과 같이 실험결과를 정리하였다.

$N1$: 전체하한과 일치하는 전체작업완료시간의 수
$N2$: 전체하한으로부터의 상대편차가 0%보다 크고 3%보다 작거나 같은 전체작업완료시간의 수
$N3$: 전체하한으로부터의 상대편차가 3%보다 크고 5%보다 작거나 같은 전체작업완료시간의 수
$N4$: 전체하한으로부터의 상대편차가 5%보다 큰 전체작업완료시간의 수
Min : 전체하한으로부터의 최소 상대편차
Avr : 전체하한으로부터의 평균상대편차
Max : 전체하한으로부터의 최대 상대편차

표 9. 제안 알고리즘을 Case 1에 적용한 결과

n	$N1$	$N2$	$N3$	$N4$	Min(%)	Avr(%)	Max(%)
5	14	4	0	2	0.00	1.71	14.29
10	17	3	0	0	0.00	0.13	0.99
15	20	0	0	0	0.00	0.00	0.00
20	20	0	0	0	0.00	0.00	0.00
25	19	1	0	0	0.00	0.02	0.40
30	20	0	0	0	0.00	0.00	0.00
35	20	0	0	0	0.00	0.00	0.00
40	20	0	0	0	0.00	0.00	0.00
45	20	0	0	0	0.00	0.00	0.00
50	20	0	0	0	0.00	0.00	0.00
55	20	0	0	0	0.00	0.00	0.00
60	20	0	0	0	0.00	0.00	0.00
65	20	0	0	0	0.00	0.00	0.00
70	20	0	0	0	0.00	0.00	0.00
75	20	0	0	0	0.00	0.00	0.00
80	20	0	0	0	0.00	0.00	0.00
85	20	0	0	0	0.00	0.00	0.00
90	20	0	0	0	0.00	0.00	0.00
95	20	0	0	0	0.00	0.00	0.00
100	20	0	0	0	0.00	0.00	0.00

표 10. 제안 알고리즘을 Case 2에 적용한 결과

n	$N1$	$N2$	$N3$	$N4$	Min(%)	Avr(%)	Max(%)
5	6	1	0	13	0.00	22.14	75.00
10	13	0	0	7	0.00	9.18	51.19
15	13	2	2	3	0.00	3.06	18.72
20	14	4	0	2	0.00	1.19	11.34
25	19	0	0	1	0.00	0.67	0.00
30	20	0	0	0	0.00	0.00	0.00
35	20	0	0	0	0.00	0.00	0.00
40	20	0	0	0	0.00	0.00	0.00
45	20	0	0	0	0.00	0.00	0.00
50	20	0	0	0	0.00	0.00	0.00
55	20	0	0	0	0.00	0.00	0.00
60	19	1	0	0	0.00	0.01	0.17
65	20	0	0	0	0.00	0.00	0.00
70	20	0	0	0	0.00	0.00	0.00
75	20	0	0	0	0.00	0.00	0.00
80	20	0	0	0	0.00	0.00	0.00
85	19	1	0	0	0.00	0.01	0.11
90	20	0	0	0	0.00	0.00	0.00
95	20	0	0	0	0.00	0.00	0.00
100	20	0	0	0	0.00	0.00	0.00

표 11. 제안 알고리즘을 Case 3에 적용한 결과

n	$N1$	$N2$	$N3$	$N4$	Min(%)	Avr(%)	Max(%)
5	17	0	0	3	0.00	2.18	27.66
10	19	1	0	0	0.00	0.02	0.39
15	19	1	0	0	0.00	0.01	0.29
20	19	1	0	0	0.00	0.03	0.55
25	20	0	0	0	0.00	0.00	0.00
30	18	2	0	0	0.00	0.03	0.34
35	19	1	0	0	0.00	0.00	0.09
40	20	0	0	0	0.00	0.00	0.00
45	20	0	0	0	0.00	0.00	0.00
50	20	0	0	0	0.00	0.00	0.00
55	20	0	0	0	0.00	0.00	0.00
60	20	0	0	0	0.00	0.00	0.00
65	20	0	0	0	0.00	0.00	0.00
70	20	0	0	0	0.00	0.00	0.00
75	20	0	0	0	0.00	0.00	0.00
80	20	0	0	0	0.00	0.00	0.00
85	20	0	0	0	0.00	0.00	0.00
90	20	0	0	0	0.00	0.00	0.00
95	20	0	0	0	0.00	0.00	0.00
100	20	0	0	0	0.00	0.00	0.00

표 12. 제안 알고리즘을 Case 4에 적용한 결과

n	$N1$	$N2$	$N3$	$N4$	Min(%)	Avr(%)	Max(%)
5	15	1	2	2	0.00	1.99	25.21
10	15	2	2	1	0.00	0.90	6.37
15	18	2	0	0	0.00	0.06	1.01
20	17	3	0	0	0.00	0.07	0.63
25	17	3	0	0	0.00	0.05	0.69
30	19	1	0	0	0.00	0.02	0.36
35	19	1	0	0	0.00	0.00	0.10
40	20	0	0	0	0.00	0.00	0.00
45	20	0	0	0	0.00	0.00	0.00
50	20	0	0	0	0.00	0.00	0.00
55	20	0	0	0	0.00	0.00	0.00
60	20	0	0	0	0.00	0.00	0.00
65	19	1	0	0	0.00	0.00	0.06
70	20	0	0	0	0.00	0.00	0.00
75	20	0	0	0	0.00	0.00	0.00
80	20	0	0	0	0.00	0.00	0.00
85	19	1	0	0	0.00	0.00	0.04
90	20	0	0	0	0.00	0.00	0.00
95	20	0	0	0	0.00	0.00	0.00
100	20	0	0	0	0.00	0.00	0.00

모의실험결과는 4 가지 Case로 나누어 <표 9> ~ <표 12>에 제시되었다. <표 9>, <표 11>, <표 12>를 살펴보면 계산된 전체작업완료시간이 전체하한과 일치하거나 거의 같다는 것을 알 수 있다. 특히 모든 경우에 있어서 작업의 수가 30 이상일 때 더 좋은 성능을 나타내고 있다. 따라서, 작업의 수가 큰 경우에 대해 제안 알고리즘이 유용하게 사용될 수 있으며, 아직 일반화되지 못한 이종 병렬기계에 대한 연구에서 대략적인 하한으로 사용될 수도 있을 것이다. <표 10>을 살펴보면 Case 2의 결과는 나머지 경우의 결과와 비교할 때 상대적으로 좋지 않은 결과를 보여준다.

또한, 처리시간분포조합에 따른 제안 알고리즘의 성능변화를 비교하기 위해 제안 알고리즘의 Case별 전체하한의 평균편차와 5% 이내 편차 발생 도수를 <그림 6> ~ <그림 7>에 정리하였다. Case 2의 경우 작업의 수가 적을 때 제안 알고리즘의 성능이 현저히 떨어짐을 알 수 있다. 그 이유는 3.2절에서 논하였듯이 Case 2의 처리시간분포가 dominant data set이기 때문이라고 판단된다. 실제 발생된 데이터를 조사한 결과, 작업장 1의 두 기계에 발생한 처리시간의 평균이 작업장 2에 발생된 처리

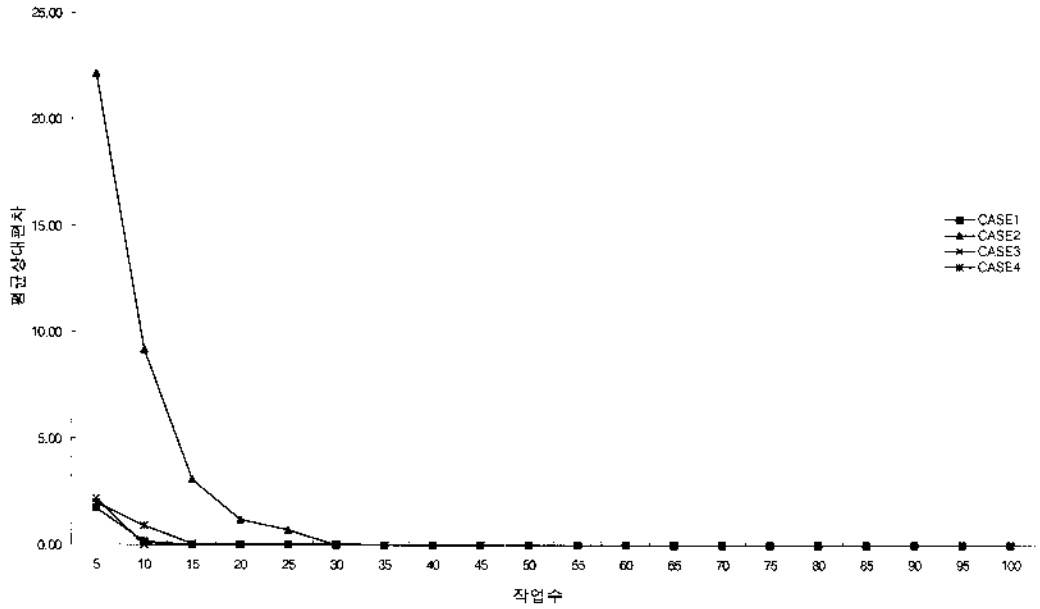

그림 6. Case별 제안 알고리즘과 전체하한과의 평균상대편차.

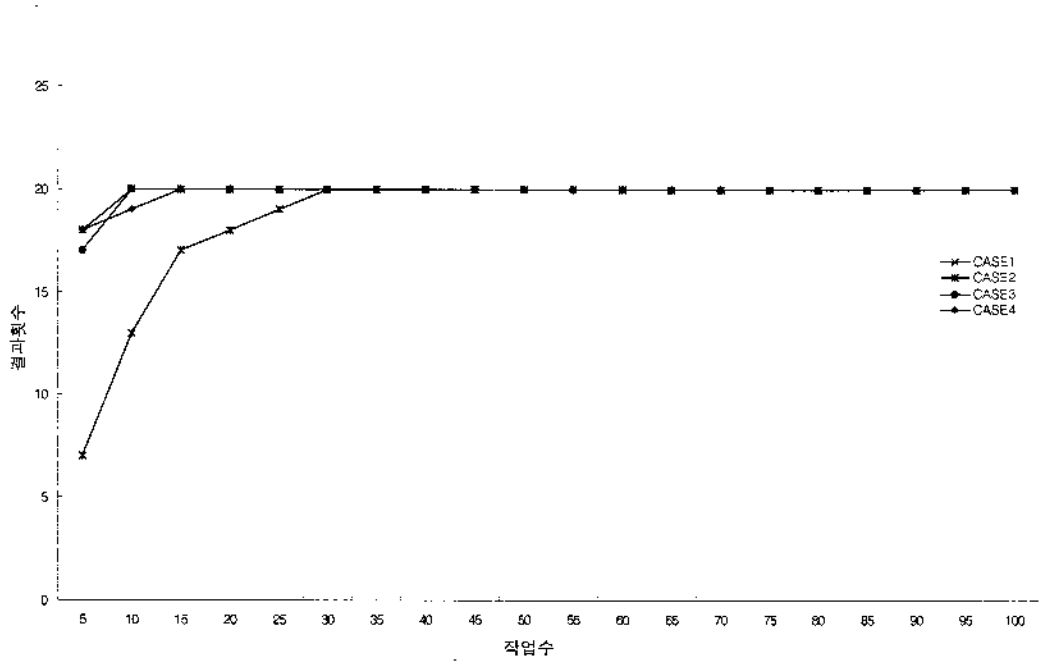

그림 7. Case별 제안 알고리즘이 전체하한과 5% 이내의 상대편차를 나타내는 결과의 수.

시간의 평균의 두 배가 넘는 경우 계산된 전체작업완료시간과 전체하한과의 차이가 발생한다는 것을 알 수가 있었다. 그러나 최적 전체작업완료시간은 계산된 전체하한보다 작을 수 없기 때문에, 실제 최적 전체작업완료시간을 고려한다면 더 나은 결과를 기대할 수 있다.

5. List Heuristic과 제안 알고리즘의 비교

이 절에서는 3절과 4절에서 설명한 휴리스틱 알고리즘들의 성능을 비교하였다. 3절의 연구 결과로, 선작업할당-후작업처리

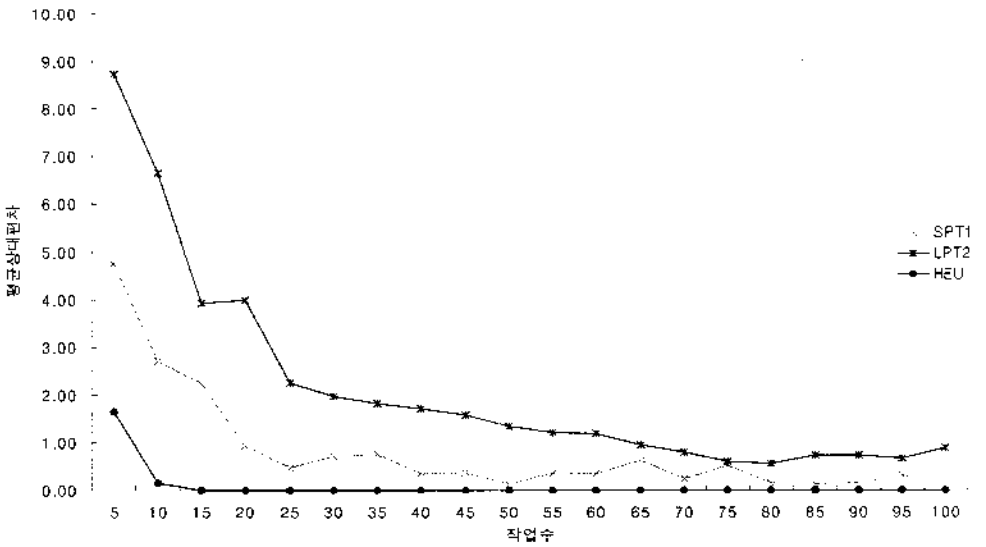

그림 8. Case 1에 SPT1, LPT2, HEU를 적용한 경우의 평균상대편차 비교.

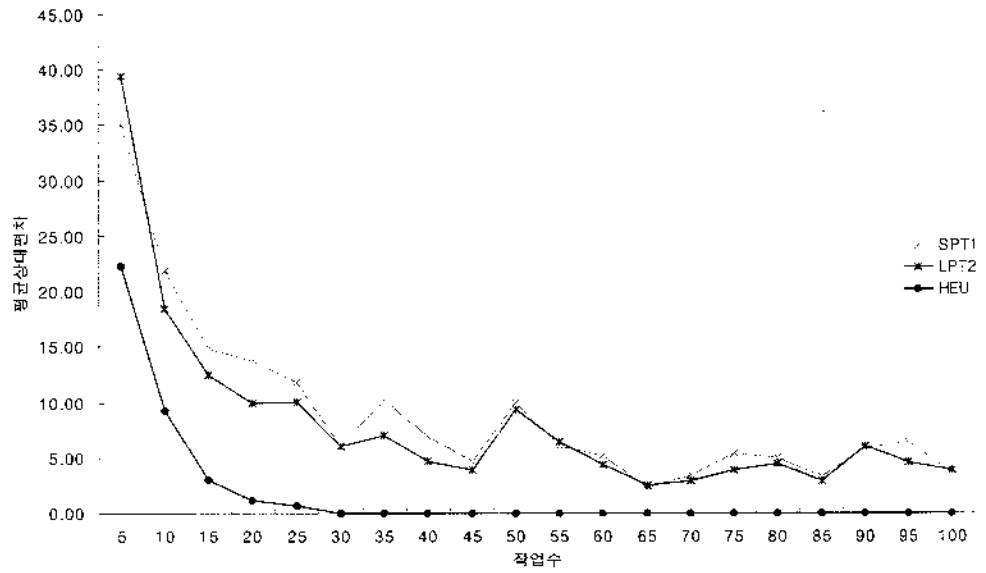

그림 9. Case 2에 SPT1, LPT2, HEU를 적용한 경우의 평균상대편차 비교.

순서결정규칙을 쓸 경우 SPT1과 LPT2가 우수한 성능을 보임을 알았으며, 4절의 연구결과로 새롭게 제안한 선작업할당-후작업처리순서결정 규칙을 쓰면 처리 작업수나 처리작업시간의 분포가 바뀌어도 대부분의 경우 구해진 해와 해의 하한과의 편차가 5% 이내에 존재하게 됨을 알았다.

그 동안 널리 사용되어온 선작업처리순서결정-후작업할당 규칙(list heuristic)에 비해 본 연구에서 제안한 선작업할당-후작업처리순서결정 규칙(제안 알고리즘)의 성능이 어떠한지 규명하기 위해 3절에서 사용된 SPT1과 LPT2 및 4절에서 제안한 알고리즘(HEU)을 사용해 4가지 case 별, 20가지 작업수별, 각 20

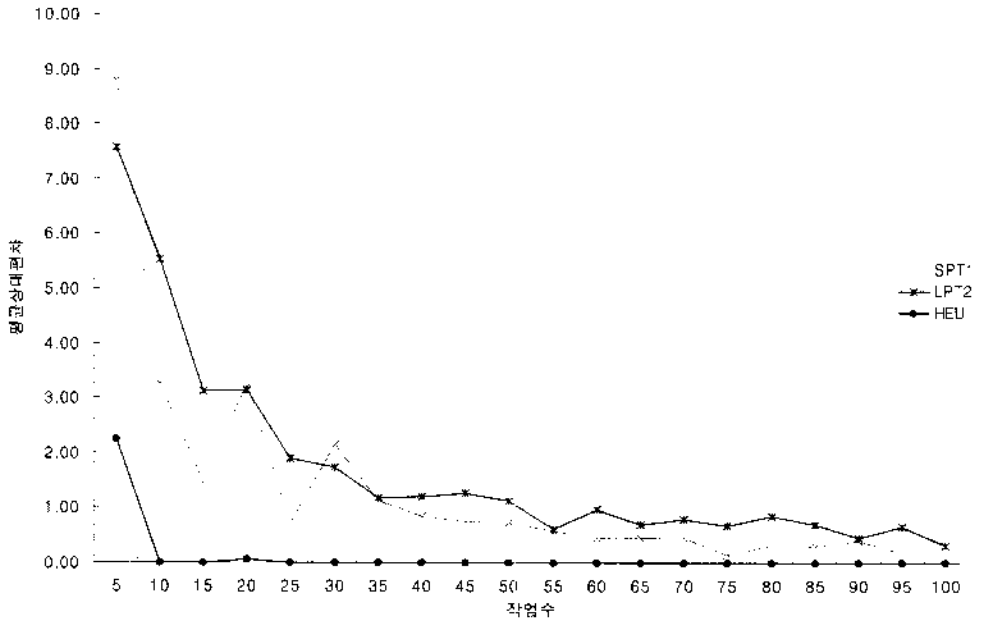

그림 10. Case 3에 SPT1, LPT2, HEU를 적용한 경우의 평균상대편차 비교.

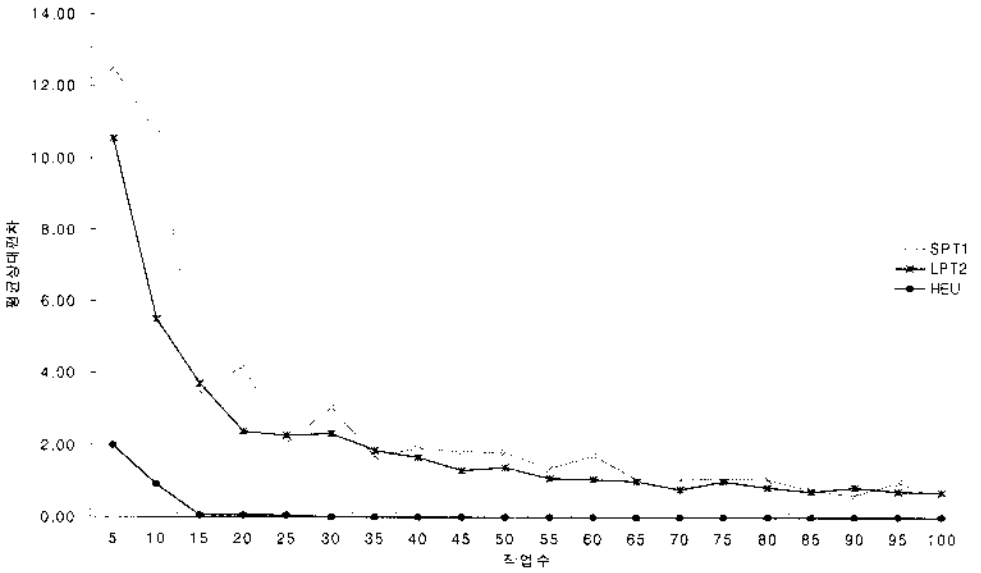

그림 11. Case 4에 SPT1, LPT2, HEU를 적용한 경우의 평균상대편차 비교.

회씩 총 1600회의 모의실험을 수행하였다.

각각의 규칙을 사용하여 얻은 해와 전체하한과의 평균상대편차를 비교한 결과가 <그림 8> ~ <그림 11>에 제시되었다. 표와 그림에서 알 수 있듯이 사용된 세 가지 방법들은 모든 경우에 대해 작업의 수가 적을 때 편차가 커짐을 알 수 있으며, 그 중 Case 2의 경우가 가장 큰 편차를 보이고 있다. 제안 알고리즘은 list heuristic의 두 방법과 비교했을 때 월등한 성능을 보이고 있다. 즉, 제안 알고리즘을 적용한 결과는 모든 경우에 있어서 작업의 수가 30 이상일 때 전체하한과의 차이를 보이지 않는 반면, 두 개의 list heuristic을 적용한 경우는 어느 정도의 차이를 보이고 있다. 이는 2.2 절에서 언급하였듯이, 선작업처리순서결정-후작업할당 규칙을 적용하는 경우에는 처리순서 결정단계에서 아무리 최적으로 순서가 결정되었다 할지라도 그 순서대로 마지막 작업장에서도 처리된다고 보장할 수 없는

반면, 선작업할당-후작업처리순서결정 규칙을 적용하면 전체적인 순서는 혼합된 순서로 나타나지만 부분적이나마 작업의 처리순서를 유지하게 되기 때문에 나타난 결과라고 판단된다 (<그림 2>, <그림 3> 참조).

지금까지의 결과를 놓고 본다면 제안 알고리즘은 SPT1과 LPT2 보다 전체작업완료시간(makespan)을 최소화시키는데 있어서 우수한 일정계획을 생성시킨다고 볼 수 있으며, 더 나아가서 본 논문에서 고려한 2단계 혼합 흐름생산시스템의 전체작업완료시간을 최소화시키기 위해서는 선작업처리순서결정-후작업할당 방법보다 선작업할당-후작업처리순서결정 방법을 사용하는 것이 타당함을 나타낸다.

6. 결 론

본 논문에서 고려된 시스템은 첫 번째 작업장에 두 대의 이종 병렬처리기계가 배치되고, 두 번째 작업장에 한 대의 처리기계가 배치된 혼합흐름생산시스템의 일정계획 작성 문제를 다루었다. 일정계획 작성 규칙의 성능을 평가하기 위한 척도로서 전체작업완료시간을 사용하였으며, 성능척도에 영향을 미치는 요인으로 작업처리순서결정방법, 작업할당방법, 그리고 작업처리시간을 고려하였다.

병렬처리기계와 관련된 기존의 대부분 연구처럼 먼저 작업처리순서를 결정하고 나중에 작업을 할당(sequence first-allocation second)하는 list heuristic 방법 4 종류를 사용해 본 결과 작업장 1의 두 기계 중 각 작업을 처리할 때 더 작은 처리시간을 대표시간으로 하여 SPT를 적용한 SPT1 처리순서결정규칙이 가장 우수한 것으로 나타났다.

기존의 list heuristic과는 다르게 먼저 작업을 작업장 1의 두 기계에 할당하고, 할당된 작업 내에서 처리순서를 결정(allocate first - sequencing second)하는 알고리즘을 제안하였다. 제안 알고리즘의 효율을 측정하기 위해 손쉽게 계산할 수 있는 전체하한(lower bound)을 개발하여 실험 결과와 비교한 결과, 작업의 수가 30 이상일 때 제안된 알고리즘으로 얻어진 makespan이 전체하한과 차이가 거의 없는 것으로 나타났다.

List heuristic 중 우수한 것으로 나타난 SPT1, LPT2와 제안 알고리즘의 성능을 비교한 결과, 모든 경우에 대해 SPT1, LPT2보다 제안 알고리즘의 성능이 우수한 것으로 나타났다.

이종 병렬처리기계가 배치된 혼합흐름생산시스템의 일정계획 작성 문제는 매우 일반적인 상황임에도 불구하고 문제해결의 어려움으로 인해 제대로 연구되고 있지 않았다. 본 논문은 이러한 혼합흐름생산시스템의 일정계획 수립 문제에 대한 해결 방안을 제시하고자 하였다. 본 논문에서는 첫 번째 작업장에 두 대의 이종 병렬처리기계가 배치된 경우를 연구 대상으로 하여 선작업할당-후작업처리순서결정 규칙의 우수성을 보였으나, 본 연구 결과를 첫 번째 작업장에 여러 대의 이종 병렬기계가 배치된 혼합흐름생산시스템의 경우로 일반화시키기 위한 연구를 현재 진행하고 있다.

참고문헌

Arthanary, T. S. and Ramaswamy, K.G. (1971), An extention of two machine sequencing problem, *Opsearch*, 10, 50-61.

Graham, R. L. (1966), Bounds for certain multiprocessing anomalies, *Bell System Technical Journal*, 45, 1563-1581.

Guinet, A. (1991), Textile production systems: a succession of nonidentical parallel processors systems, *Journal of the Operational Research Society*, 42(8), 655-671.

Gupta, J. N. D. and Tunc, E. A. (1991), Schedules for a two-stage hybrid flowshop with parallel machines at the second stage, *International Journal of Production Research*, 29(7), 1489-1502.

Johnson, S. (1954), Optimal two-and three-stage production schedules with set-up times included, *Naval Research Logistic Quarterly*, 1, 61-68.

Lanston, M. (1987), Interstage transportation planning in the deterministic flow-shop environment, *Operations Research*, 35, 556-564.

Lawer, E. L., Lenstra, J. K. and Rinnooy Kan, A. H. G. (1982), Recent developments in deterministic sequencing and scheduling: A survey, in: M.A.H. Dempster *et al.*(eds.), *Deterministic and Stochastic Scheduling*, Reidel, Boston, Ma., 35-73.

Narashimhan, S. L. and Panwalker, S. S. (1984), Scheduling in a two-stage manufacturing process, *International Journal of Production Research*, 22(4), 555-564.

Perrov, V. A. (1968), Flowline group production planning, *Business Publications*, London.

Randhawa, S. U. and Kuo, C. H. (1997), Evaluating scheduling heuristics for non-identical parallel processors, *International Journal of Production Research*, 35, 969-981.

Randhawa, S. U. and Smith, T. A. (1995), An experimental investigation of scheduling non-identical parallel processors with sequence-dependent set-up times and due dates, *International Journal of Production Research*, 33(1), 59-69.

Sriskadarajah, C. and Sethi, S. P. (1989), Scheduling Algorithms for flexible flow-shops: worst and average performance, *European Journal of Operational research*, 43, 143-160.