Journal of the Korean Institute of Industrial Engineers
Vol. 34, No. 4, pp. 481-488, December 2008.

배치처리기계를 포함하는 두 단계 흐름생산라인의 일정계획

고시근† · 구평회 · 김병남

부경대학교 시스템경영공학과

Production Scheduling for a Two-machine Flow Shop with a Batch Processing Machine

Shie-Gheun Koh · Pyung-Hoi Koo · Byung-Nam Kim

Department of Systems Management and Engineering, Pukyong National University

This paper deals with a scheduling problem for two-machine flow shop, in which the preceding machine is a batch processing machine that can process a number of jobs simultaneously. To minimize makespan of the system, we present a mixed integer linear programming formulation for the problem, and using this formulation, it is shown that an optimal solution for small problem can be obtained by a commercial optimization software. However, since the problem is NP-hard and the size of a real problem is very large, we propose a number of heuristic algorithms including genetic algorithm to solve practical big-sized problems in a reasonable computational time. To verify performances of the algorithms, we compare them with lower bound for the problem. From the results of these computational experiments, some of the heuristic algorithms show very good performances for the problem.

Keywords: Scheduling, Batch Process, Flow Shop, Integer Programming, Heuristic

1. 서 론

본 연구는 S사의 MLCC(Multi-Layer Ceramic Capacitor) 제조라인에서 파생된 문제인 병목공정 위주의 생산일정계획을 다룬다. 여기서 MLCC란 전기를 일시적으로 저장할 수 있는 전자소자로서 제조프로세스는 <Figure 1>과 같은 순서로 진행되는데 개략적으로 설명하면 다음과 같다. 우선 원자재인 세라믹 파우더를 반죽하여 얇게 펴고 그 위에 전기회로를 인쇄한다. 전기회로가 인쇄된 얇은 세라믹 판들을 제품종류에 따라 몇 개 층으로 적층한 다음, 압착을 통해 하나의 판으로 결합시킨다. 아직까지 무른 상태인 이 세라믹 판들은 제품에 따라 적당한 크기로 절단된 후 열처리를 통해 단단하게 경화된다. 열처리가 끝나면 제품들의 뾰족한 모서리 부분을 매끄럽게 연마하고, 제품의 양쪽 끝부분에 전극을 설치하는 것으로 대략적인 제조 프로세스가 종료된다.

이 제조과정에서 가장 시간이 많이 소요되는 부분은 열처리 과정이다. <Figure 1>에서 가소 및 소성 공정이 열처리에 해당되는 과정으로서 이 두 공정에서의 소요시간이 전체 제조시간의 25% 수준에 이른다. 그런데 이 두 공정 중 소성공정에서는 열을 가하는 소성로 속을 컨베이어 벨트에 적재된 제품이 통과하는 방식으로 열처리가 이루어지므로 제품의 투입이 연속적일 뿐 아니라 제품 특성별로 기계가 거의 전용화되어 있어 생산일정계획 상의 문제는 거의 없다. 반면에 가소공정은 오븐과 같은 형태의 기계에 로트가 투입되면 20~40시간에 이르는 처리시간 동안 다른 일을 수행할 수 없어 많은 수의 가소기계가 운영되고 있다. 그러나 이것도 충분하지 않아 가소공정 앞에서 정체가 많이 발생하며, 따라서 라인 관리자들은 가소공정을 MLCC 생산라인의 병목자원으로 생각하고 있었다. 실제로 단위시간당 생산량과 공정별 표준시간을 사용한 부하분석을 통해 가소 공정이 이 제조라인의 병목공정임을 확인할 수 있었다.

여기서 굳이 TOC(Theory Of Constraints)에서 말하는 "제약

이 논문은 2006년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2006-521-D00574).
† 연락저자 : 고시근, 608-739 부산광역시 남구 용당동 산 100, 부경대학교 시스템경영공학과, E-mail : sgkoh@pknu.ac.kr
2008년 9월 1일 접수; 2008년 10월 5일 수정본 접수; 2008년 10월 13일 게재 확정.

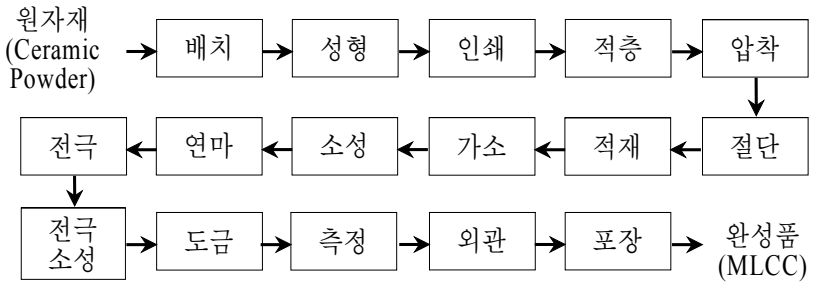

Figure 1. MLCC manufacturing process

자원의 능력이 곧 시스템 전체의 능력"이라는 원칙을 인용하지 않더라도, 전체 라인의 생산성을 제고하기 위해서는 가소공정의 효과적인 사용이 필요하고 또한 이를 위해 효율적인 생산일정계획이 필요함은 주지의 사실이다. 그런데 이 가소공정에 대한 생산일정계획을 작성하는 데는 다음과 같은 특수한 상황이 고려되어야 한다.

MLCC 제품은 매우 작고($0.3 \times 0.3 \times 0.6 \sim 3.2 \times 5.0 \times 5.7 mm^3$) 다양한 크기를 가지고 있어 제품의 크기에 따라 18,000개에서 840,000개까지의 로트를 구성하여 이 로트 단위로 생산이 이루어진다. 또한 이렇게 구성된 로트도 제품별로 부피가 모두 달라서 한 개의 로트가 가소 기계 한대를 점유하는 정도가 모두 다르다. 즉, 기계 한 대의 처리능력을 1로 보았을 때 제품의 종류에 따라 한 로트가 기계 한 대를 점유하는 비율은 0.07에서 0.92로 매우 다양하며, 따라서 가소 기계 한대가 여러 개의 로트를 한꺼번에 배치(Batch)로 처리할 수 있게 된다. 또한 MLCC 제품은 크기 외에도 다양한 전기적 특성에 따라 수백 가지 종류로 분류된다. 하지만 가소 기계의 입장에서는 가소 소요시간 및 가소 온도가 같다면 다른 종류의 제품이라 하더라도 동시에 처리할 수 있다. 즉, 제품의 종류가 다르더라도 소요시간 및 처리 온도에 따라 모든 로트들을 몇 개의 그룹으로 분류할 수 있고, 같은 그룹에 속하는 로트들은 기계의 용량이 허용하는 범위 안에서 하나의 배치로 묶어서 동시에 처리할 수 있다.

Dobson and Nambimadom(2001)이 BLSP(Batch Loading and Scheduling Problem)라고 지칭하였던 이러한 유형의 문제는 MLCC 생산라인의 가소공정 이외에도 1) 철강 및 세라믹 산업의 열처리 공정, 2) 집적회로(IC) 생산라인의 여러 가지 화학처리공정, 3) 철판이나 PCB를 절단하기 위한 NC가공공정 등 여러 산업분야에서 찾아볼 수 있다.

현재까지 발표된 BLSP 관련 연구결과들은 대부분 배치처리기계 한 공정으로 이루어진 생산시스템을 다루고 있다. 그러나 실제로는 대부분의 경우 이러한 배치처리공정은 여러 공정으로 이루어진 생산라인에 소속된 한 개의 공정이므로, 배치처리공정 하나만 효율화시키는 계획은 다른 공정의 효율을 오히려 감소시킬 수도 있어 전체적인 측면에서는 효율성을 보장할 수 없다. 특히 흐름생산 방식으로 운영되는 생산라인의 경우 라인의 균형화(Line Balancing)가 상당히 이루어져 있어 어느 한 공정만이 절대적인 병목이고 나머지 공정들은 생산능력에 많은 여유가 있다고 볼 수는 없기 때문에 한 공정만을 효율화하는 생산계획을 라인 전체에 적용하는 것은 상당한 위험성이 있다. 실제로 본 연구의 계기가 된 MLCC 생산라인에서는 가장 중요한 병목공정인 가소공정만을 대상으로 생산스케줄을 시행한 결과 가소공정의 효율은 높아졌으나 가소 이외의 공정에서 문제가 자주 발생하였다. 특히 <Figure 1>에서 "전극"공정의 경우 제품 종류별로 부하량의 편차가 커 부하가 많이 걸리는 제품이 일시적으로 몰리게 되면 제품의 흐름에 문제가 발생하게 되고, 따라서 전극공정 앞에 예상치 못했던 재고가 많이 발생하게 되었다.

본 연구에서는 이러한 문제를 해결하기 위해 라인의 제2병목공정을 생산계획시에 고려하고자 한다. 즉, 배치처리기계와 일반적인 처리기계의 두 병목공정(예를 들어 <Figure 1>에서 가소공정과 전극공정)으로 구성된 흐름라인에 대한 효과적인 생산일정계획을 도출하고자 하는 것이다. 단, 여기서 두 병목공정 사이의 공정들(예를 들어 <Figure 1>에서 소성 및 연마공정)은 능력이 충분하여 생산일정계획에 영향을 주지 않는다고 가정하고 모형에서 제외하였다.

본 논문의 구성은 다음과 같다. 우선 제 2절에서 기존의 관련연구결과에 대해 관찰한 다음, 제 3절에서는 대상문제에 대한 정수계획모형을 개발하고 상용 소프트웨어를 사용해 최적해를 구해본다. 제 4절에서는 최적해에 대한 하한(lower bound)를 개발하고, 제 5절에서 몇 가지의 단순 휴리스틱 방법론을 제안한 다음, 제 6절에서는 유전알고리즘을 사용한 접근방법을 제안한다. 제 7절에서는 수치실험을 통해 제안된 휴리스틱의 결과를 최적해의 하한과 비교함으로써 그 성능을 평가한 다음, 마지막으로 제 8절에서는 결론 및 추후 연구과제를 제시하기로 한다.

2. 기존 연구동향

BLSP는 그 동안 비교적 널리 연구되지 않았던 분야로서 Ikura and Gimple(1986)의 연구가 이 분야의 첫 번째 연구였다. 그들은 모든 작업이 같은 처리 시간을 가지는 경우에 대해 정해진 납기를 준수하는 작업일정계획을 작성하였다. Lee *et al.*(1992)

은 처리시간이 서로 다른 제품들을 하나의 배치로 묶을 수 있고 그 배치의 처리시간은 묶여진 작업들 중에서 가장 긴 처리시간과 같아지는 경우의 BLSP를 연구했다. 그 결과 단일기계 및 병렬기계 문제에 대해 지연 작업의 수를 최소화하는 효율적인 알고리즘을 제안했다. Chandru *et al*.(1993)은 단일 기계 BLSP의 최적해를 구하기 위한 branch-and-bound 알고리즘을 개발하였다. 그러나 이 알고리즘은 크기가 작은 문제에만 적용이 가능하므로 현실적인 크기의 문제를 해결하기 위한 2개의 휴리스틱 알고리즘을 제안하였다. Hochbaum and Landy (1997)는 작업들의 완료시간 합을 최소화하기 위한 효율적인 동적계획법(Dynamic Programming) 알고리즘을 제안하였다. Uzsoy(1995)는 공통된 작업시간을 가지는 작업그룹들이 존재하여 같은 그룹에 소속된 작업들만이 한 배치를 형성할 수 있고 각 작업들의 부피는 모두 동일한 경우의 BLSP를 연구하였다. 그 결과 단일 기계 및 병렬 기계에 대해 여러 가지의 수행척도를 대상으로 최적 및 근사해를 구하는 알고리즘을 제시하였다. Mehta and Uzsoy(1998)도 같은 그룹에 소속된 작업들만이 배치를 형성할 수 있는 경우의 BLSP를 다루었다. 그들은 총 지연시간을 최소화하기 위한 동적계획 알고리즘과 적정한 시간에 해답을 구할 수 있는 휴리스틱을 제시하였다. Lee and Uzsoy(1999)는 작업의 도착이 동적일 때 총 완료시간을 최소화하기 위한 단일기계 BLSP를 다루었다. 그들은 몇 가지 휴리스틱 알고리즘을 개발하고 그 성능을 수치실험으로 관찰하였다.

위에서 제시한 연구들은 모두 각 작업들의 부피가 같은 상황을 고려하였다. 작업들의 부피가 다른 경우의 BLSP에 대해서는 Dobson and Nambimadom(2001)이 연구하였다. 그들은 같은 그룹에 소속된 작업들은 모두 동일한 처리시간을 갖는다고 가정하고 작업별 완료시간의 가중합을 최소화하는 BLSP를 다루었다. 그들은 최적해에 대한 하한을 제시하기 위해 문제에 대한 정수계획법 수리모형을 개발하고 몇 개의 휴리스틱을 제시한 뒤 최적해의 하한과 비교하여 휴리스틱의 성능을 평가하였다. Uzsoy(1994)는 다른 그룹에 속한 서로 다른 부피의 작업이 하나의 배치로 같이 처리될 수 있는 단일기계 BLSP를 연구했다. 그는 배치의 처리시간이 배치 안에서 가장 긴 처리시간을 가진 작업의 처리시간과 같아진다는 가정 하에 총 완료시간을 최소로 하는 몇 개의 휴리스틱과 더불어 크기가 작은 예제의 경우에는 최적해를 구할 수 있는 branch-and-bound 알고리즘을 제안했다. Kempf *et al*.(1998)은 기계의 능력 뿐 아니라 보드 이용률을 고려한 단일기계 BLSP를 연구했다. 최근에는 Koh *et al*.(2004, 2005)이 가장 일반적인 상황(단일 및 병렬기계, 작업대상의 크기는 다양, 동일 작업그룹 안에서만 배치형성 가능)의 BLSP를 해결하기 위한 모형을 제시하고 매우 효과적인 해결 방법론을 제시하였다.

배치처리가 아닌 일반적인 두 기계로 이루어진 흐름생산라인에 대해 전체 작업완료시간(makespan)을 최소화하는 Johnson 규칙(Johnson, 1954)이 발표된 이후 두 단계 흐름생산라인에 대해서는 수많은 연구가 이루어졌다. 그러나 배치처리기계를 포함한 흐름생산라인에 대해서는 매우 적은 수의 연구가 수행되었다. 우선 Ahmadi *et al*.(1992)는 배치처리기계를 포함하는 두 기계 및 세 기계 흐름라인의 일정계획문제를 다루었으므로 본 연구와 매우 유사하다고 볼 수 있다. 그러나 그들의 연구에서는 작업의 크기가 모두 동일하여 배치로 묶을 수 있는 작업의 수가 고정된 상황을 다루었다. Chandra and Gupta(1997)는 특정한 사례 생산라인을 대상으로 배치처리기계를 고려한 생산계획방법론을 개발하고 효과를 평가하였다. 마지막으로 Bhatnagar *et al*.(1999)은 배치처리기계가 포함된 복잡한 생산라인에 대해 생산일정계획 및 제품조합 문제를 연구하였다.

본 연구에서는 배치처리기계 한 대와 일반적인 기계 한 대가 직렬로 연결된 흐름생산라인에서 작업들의 총 완료시간(makespan)을 최소화하기 위한 일정계획방법론을 찾고자 한다. 배치처리기계는 Koh *et al*.(2005)이 가정한 형태와 동일하다. 즉, 각 작업들의 크기는 다양(arbitrary job sizes)하고, 처리시간 등의 특성으로 인해 하나의 배치로 묶여질 수 있는 작업들에는 제한(incompatible job families)이 있다. 이러한 문제에 대한 정수계획모형을 개발한 다음, 다양한 휴리스틱 알고리즘의 제안을 통해 적절한 시간 내에 좋은 품질의 해를 구하는 방안을 모색한다.

3. 혼합정수계획모형

본 절에서는 앞서 제시한 문제에 대한 정수계획모형을 개발한다. 이를 위해 다음과 같은 용어를 정의한다.

n : 계획대상 작업수
m : 배치로 묶여질 수 있는 작업 패밀리의 수
i : 작업에 대한 인덱스, $i = 1, 2, \cdots, n$
j : 작업 패밀리에 대한 인덱스, $j = 1, 2, \cdots, m$
k : 배치 및 투입순서에 대한 인덱스, $k = 1, 2, \cdots, n$
f_j : 패밀리 j에 속하는 작업 인덱스 집합
r_i : 작업 i의 크기(기계의 크기를 1로 하여 normalize 한 값), $0 < r_i \leq 1$
p_j : 패밀리 j에 속하는 작업에 대한 배치처리기계의 처리시간
q_i : 일반기계의 작업 i 처리시간

모형에서 사용되는 변수들은 다음과 같다.

x_{ik} : 작업 i가 배치 k에 포함되면 1, 그렇지 않으면 0
y_{jk} : 배치 k가 패밀리 j인 작업들로 이루어지면 1, 그렇지 않으면 0
C_{1k} : 기계 1(즉, 배치처리기계)에서 배치 k를 마친 시간
C_{2k} : 기계 2(즉, 일반기계)에서 배치 k를 마친 시간

이 기호들을 사용해 본 연구의 대상문제를 혼합정수계획(MILP; Mixed Integer Linear Programming) 모형으로 표현하면

다음과 같다.

$$\text{Minimize } C_{2n} \tag{1}$$

subject to

$$\sum_{i=1}^{n} r_i x_{ik} \le 1, \quad k = 1, 2, \cdots, n, \tag{2}$$

$$\sum_{k=1}^{n} x_{ik} = 1, \quad i = 1, 2, \cdots, n, \tag{3}$$

$$\sum_{j=1}^{m} y_{jk} \le 1, \quad k = 1, 2, \cdots, n, \tag{4}$$

$$x_{ik} \le y_{jk}, \ i \in f_j, \quad k = 1, 2, \cdots, n, \tag{5}$$

$$C_{11} = \sum_{j=1}^{m} p_j y_{j1}, \tag{6}$$

$$C_{21} = C_{11} + \sum_{i=1}^{n} q_i x_{i1}, \tag{7}$$

$$C_{1k} = C_{1,k-1} + \sum_{j=1}^{m} p_j y_{jk}, \quad k = 2, 3, \cdots, n, \tag{8}$$

$$C_{2k} = \max\{C_{1k}, C_{2,k-1}\} + \sum_{i=1}^{n} q_i x_{ik}, \quad k = 2, 3, \cdots, n, \tag{9}$$

$$x_{ik}, \ y_{jk} = 0 \text{ or } 1, \ \ i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, m, \ \ k = 1, 2, \cdots, n. \tag{10}$$

모형에서 식 (1)은 이 문제가 총 완료시간 최소화문제임을 보여준다. 식 (2)는 각 배치에 포함된 작업들의 크기 합이 배치처리기계의 용량보다 크지 않도록 해주며, 식 (3)은 하나의 작업이 하나의 배치에만 포함되도록 한다. 식 (4)는 각 배치에 포함되는 작업들의 패밀리가 하나를 넘을 수 없음을 표현한다. 최적해가 몇 개의 배치로 이루어지는지 문제를 풀기 전에는 알 수가 없으므로 모형에서는 최대 배치수로 n을 가정한다. 따라서 모형의 해를 구하면 어떤 배치에는 두 개 이상의 작업(물론 하나의 패밀리에 포함된)이 포함되기도 하고 어떤 배치에는 작업이 할당되지 않을 수도 있다. 식 (4)가 부등식인 이유가 여기에 있다. 식 (5)에 의하면, 특정 작업 패밀리가 특정 배치에 포함되지 않을 경우(즉, $y_{jk} = 0$), 그 패밀리에 속한 작업도 그 배치에 포함될 수 없다(즉, $x_{ik} = 0$). 반대로 특정 작업이 특정 배치에 포함되면(즉, $x_{ik} = 1$), 그 작업이 속한 패밀리도 그 배치에 포함되어야 한다(즉, $y_{jk} = 1$). 식 (6)~식 (9)는 각 기계에서 각 배치의 완료시간을 계산해준다.

모형의 유효성을 검사하기 위해 다음과 같은 데이터를 갖는 예제를 만들어 최적해를 구해 보았다.

n = 13, m = 2, f_1 = {1, 2, 3, 4, 5, 6}, f_2 = {7, 8, 9, 10, 11, 12, 13}, p = (15, 20), r = (0.3, 0.1, 0.5, 0.3, 0.6, 0.1, 0.5, 0.2, 0.3, 0.4, 0.2, 0.6, 0.1), and q = (7, 5, 9, 8, 7, 5, 8, 7, 4, 7, 9, 6, 5).

이 데이터를 사용해 상용 최적화 소프트웨어인 LINGO를 실행하였더니 3분 17초의 실행시간 이후 최적해로 5개의 Batch가 {1, 3, 6}, {7, 9, 11}, {10, 12}, {2, 4, 5}, {8, 13}의 투입순서로 형성되었다. 이 경우 완료시간은 102로서 위의 데이터를 사용하여 쉽게 확인할 수 있다.

비록 위의 예제는 그리 길지 않은 시간에 최적해를 구할 수 있었지만 작업의 수가 15개만 넘어도 1시간 내에 최적해를 구할 수가 없었다. 그런데 현실적인 문제에서는 작업의 수가 수백 개인 경우가 대부분이므로 LINGO나 CPLEX같은 최적화 소프트웨어를 사용해 현실문제의 최적해를 구하는 것은 거의 불가능하다고 보는 것이 타당하다. 이론적으로도 이 유형의 문제는 NP-hard에 속한다. Garey and Johnson (1979)에 의하면 Bin packing 문제가 strongly NP-hard이다. 그런데 본 연구에서 다루는 문제의 일부분인 '배치처리기계 한대만 있고 작업 패밀리도 하나만 있는 경우'가 Bin packing 문제와 동일하므로 본 연구의 대상문제도 strongly NP-hard라고 할 수 있다. 따라서 본 연구에서는 모형의 최적해를 구하는 것 보다는 적절한 시간 안에 좋은 품질의 해를 찾아주는 휴리스틱 방법론을 개발하는 데 초점을 맞추기로 한다.

4. 최적해의 하한

본 절에서는 앞에서 제시하였던 모형의 최적해에 대한 하한(lower bound)을 제시한다. 하한을 찾는 목적은 앞으로 개발하는 휴리스틱 알고리즘의 성능을 평가하는 것이다. 현실적인 크기의 문제에 대해서는 최적해를 알 수가 없으므로 휴리스틱의 성능을 평가하기 위한 기준값을 제시하는 것이다.

하한을 계산하는 아이디어는 두 가지가 있다. 우선 배치처리기계를 작업 패밀리 중에서 가장 작은 처리시간을 갖는 패밀리 배치가 최초로 통과한다고 가정하면 그 시간은 $\min\{p_j\}$가 된다. 그 이후 모든 작업들이 연결되어 유휴시간 없이 두 번째 기계를 통과하면 그 소요시간은 $\sum_{i=1}^{n} q_i$이므로 makespan에 대한 첫 번째 하한은

$$LB_1 = \min\{p_j\} + \sum_{i=1}^{n} q_i \tag{11}$$

로 놓을 수 있다.

두 번째 하한은 배치처리기계를 최소한의 배치수로 통과한 다음 두 번째 기계를 최소의 시간으로 통과하는 경우에 대한 값이다. 각 작업 패밀리에 대해 최소의 배치수는 $\left\lceil \sum_{i \in f_j} r_i \right\rceil$라고 할 수 있다. 여기서 $\lceil x \rceil$는 x보다 작지 않은 최소의 정수를 의미한다. 그러면 배치처리기계에 대한 makespan 하한값은 $\sum_{j=1}^{m} p_j \left\lceil \sum_{i \in f_j} r_i \right\rceil$가 되므로 두 기계에 대한 makespan 하한값으로는

$$LB_2 = \sum_{j=1}^{m} p_j \left\lceil \sum_{i \in f_j} r_i \right\rceil + \min\{q_i\} \quad (12)$$

를 사용할 수 있다.

하한값은 클수록 좋으므로 본 연구에서는 이 두 개의 값 중에서 큰 값을 하한으로 사용한다. 즉,

$$LB = \max\{LB_1,\ LB_2\} \quad (13)$$

를 최적해의 하한으로 사용한다.

앞에서 보았던 예제에 대해 적용해 보면 $LB_1 = \min\{15, 20\} + 87 = 102$ 이고 $LB_2 = 15 \lceil 1.9 \rceil + 20 \lceil 2.3 \rceil + 4 = 94$이므로 LB는 102이다. 이 값은 최적해와 동일하므로 매우 훌륭한 하한이라고 생각된다. 하지만 이 경우가 그렇다고 해서 모든 문제에 적용될 수는 없을 것이다. 최악의 경우 앞의 예제에서 작업의 크기가 모두 0.5를 약간 넘는다고 하면 모든 작업들이 각각 배치를 형성해야 한다. 이러한 상황이 되면 이 문제는 일반적인 두 기계 흐름라인이 되어 Johnson의 규칙(Johnson, 1954)을 적용해 최적해를 구할 수 있게 되고 그 때의 makespan값은 234가 된다. 하지만 이것은 매우 드문 경우로서, 위의 예제에서 작업의 크기를 0과 1사이에서 임의로 생성해 실험해본 결과 식 (13)의 하한값은 최적해에 매우 가까운 값을 보여주었다.

5. 휴리스틱 알고리즘

본 연구의 일정계획문제는 두 개의 문제가 결합된 문제이다. 하나는 개별 작업들을 배치로 묶는 것이고, 또 하나는 그 배치들의 투입순서를 결정하는 것이다. 일단 배치가 형성되면 투입순서는 Johnson의 규칙(Johnson, 1954)에 따르는 것이 최적이라고 알려져 있으므로, 본 연구에서 제시하는 휴리스틱들은 배치형성방법에 따라 구분되며 배치투입순서는 모두 Johnson 규칙을 따른다.

첫 번째로 제시하는 휴리스틱은 Johnson *et al.*(1974)이 Bin Packing 문제에 대해 최악의 경우에도 뛰어난 성능(excellent worst case performance)을 보여준다고 하였던 First Fit Decreasing 휴리스틱을 사용한 것이다. Koh *et al.*(2005)은 배치처리기계의 완료시간을 최소화하기 위해 이 휴리스틱을 응용한 LFF(Largest job First Fit) 휴리스틱을 사용하였고 실험 결과 뛰어난 성능을 발휘함을 보였다. 따라서 본 연구에서는 배치처리기계의 효율을 높이기 위해 이 LFF 휴리스틱으로 배치를 형성한 다음 처리순서는 Johnson의 규칙을 적용하기로 한다. 이 휴리스틱은 LFF와 Johnson을 적용하므로 LFF-JS라고 부른다.

Algorithm LFF-JS

단계 1 : 모든 패밀리에 작업이 남아 있지 않으면 <단계 3>으로 간다. 남아 있는 패밀리가 있다면 그 패밀리의 작업들을 크기(즉, r_i)가 작아지는 순서로 sorting한다.

단계 2 : 패밀리 내에 남아 있는 작업이 없으면 <단계 1>로 간다. 작업이 남아 있다면 첫 번째 작업을 그 작업이 들어갈 수 있는 첫 번째 배치에 포함시키고 남은 작업 리스트에서 제거한다. 만일 기존의 배치에 들어갈 자리가 없다면 새로운 배치를 만들어 그 배치에 포함시킨다.

단계 3 : 모든 배치가 형성되었고 각 배치의 선행기계(즉, 배치처리기계) 소요시간이 정해졌으므로, 각 배치의 후속기계 소요시간을 그 배치에 포함된 작업들의 후속기계 작업시간 합으로 놓으면, 각 배치를 대상으로 Johnson의 규칙(Johnson, 1954)을 적용해 투입순서를 정할 수 있다.

위의 휴리스틱은 후속기계와의 관계를 고려하지 않고 가능한 한 적은 수의 배치를 형성하려는 규칙이다. 따라서 배치처리기계 하나만을 대상으로 하는 문제에서는 뛰어난 성능을 발휘하였지만 흐름라인에서의 성능은 장담할 수가 없다. 따라서 다음으로 제시하는 휴리스틱에서는 후속기계의 작업시간을 고려하여 배치를 형성한다. 즉, 후속기계의 작업시간이 긴 순서대로 배치를 형성(TFF; longest processing Time First Fit)하고 Johnson의 규칙에 따라 배치의 투입순서를 정하는 TFF-JS 휴리스틱이다.

Algorithm TFF-JS

단계 1 : 모든 패밀리에 작업이 남아 있지 않으면 <단계 3>으로 간다. 남아 있는 패밀리가 있다면 그 패밀리의 작업들을 후속기계의 작업시간(즉, q_i)가 작아지는 순서로 sorting한다.

단계 2 : LFF-JS와 동일

단계 3 : LFF-JS와 동일

앞에서 제시한 두 개의 휴리스틱은 작업의 크기 혹은 후속기계의 작업시간이라는 한 개의 기준으로 배치를 형성하였다. 다음에 제시하는 휴리스틱은 이 두 개의 기준을 모두 사용한다. 즉, 작업의 크기와 후속기계의 작업시간을 곱한 값을 기준으로 배치를 형성(PFF; largest Product value First Fit)한 다음 Johnson 규칙으로 배치의 투입순서를 정하는 PFF-JS 휴리스틱을 제안한다.

Algorithm PFF-JS

단계 1 : 모든 패밀리에 작업이 남아 있지 않으면 <단계 3>으로 간다. 남아 있는 패밀리가 있다면 그 패밀리의 작업들을 크기와 후속기계 작업시간의 곱(즉, $r_i q_i$)이 작아지는 순서로 sorting한다.

단계 2 : LFF-JS와 동일

단계 3 : LFF-JS와 동일

6. 유전 알고리즘

본 절에서는 유전 알고리즘을 응용한 방법론을 제안한다. Koh *et al*.(2004, 2005)이 사용하였던 방법과 유사하게 구간 (0, 1)의 random number를 유전자(gene)로 사용한다. 즉, 전체 작업수(n) 만큼의 random number로 하나의 염색체(chromosome)가 구성되고, 각 유전자의 크기에 따라 배치를 형성하게 된다. 하나의 염색체로부터 하나의 일정을 생성하는 구체적인 절차는 다음과 같다.

Algorithm DECODE

단계 1 : 모든 패밀리에 작업이 남아 있지 않으면 <단계 3>으로 간다. 남아 있는 패밀리가 있다면 그 패밀리의 작업들을 해당 유전자 값이 작아지는 순서로 sorting 한다.

단계 2 : LFF-JS와 동일

단계 3 : LFF-JS와 동일

각 염색체로부터 일정을 생성하고 그 일정의 makespan을 계산한 다음에는 그 염색체의 적합도 값(fitness value)을 계산하여야 한다. 그런데 본 연구의 문제는 makespan을 최소화하는 문제이므로 계산된 makespan 값을 적합도 값으로 바로 사용할 수 없다. 따라서 다음과 같은 간단한 변형을 통해 각 염색체의 적합도 값(F_i)을 계산한다. 단, 여기서 Z_i는 각 염색체로부터 계산된 makespan 값이고 $Z_{\max}$는 그 세대(generation)의 makespan 값 중 최대값을 의미한다.

$$F_i = Z_{\max} - Z_i \tag{14}$$

각 세대의 염색체 중 가장 우수한 두 개는 다음 세대로 바로 넘기고 나머지 염색체들은 교배(crossover) 및 돌연변이(mutation) 연산을 통해 생성하여 유전 알고리즘을 진행한다. 해의 발전이 100세대동안 일어나지 않으면 알고리즘을 종료하도록 하였다. 염색체의 수(population size), 교배확률, 돌연변이 확률 등의 모수들은 여러 번의 실험을 거쳐 적절한 값을 찾아 사용하였다.

7. 수치실험

본 절에서는 임의로 생성된 문제들을 통해 앞 절에서 제안하였던 휴리스틱의 성능을 평가한다. 문제의 복잡도가 작업의 수(n)와 작업 패밀리의 수(m)에 밀접하게 관련되어 있으므로 이 두 개의 값을 여러 수준으로 변화시키며 테스트를 실시한다. 사용되는 수준은 n의 경우 50, 100, 150, 200 등 4수준이고 m의 경우 2, 4, 6, 8, 10 등 5수준이다. 총 20개의 (n, m) 조합에 대해 각각 30개의 문제를 생성한다. 각 문제를 만들기 위해서 우선 n개의 r_i값을 U(0, 100)/100 을 통해 생성한다. 여기서 U(a, b)는 a보다 크고 b보다 작은 임의의 정수를 의미한다. 다음으로 p_j값은 $j = 1, 2, \cdots, m$에 대해 U(10j, 10j+10) 을 통해 생성한다. 각 작업 패밀리에 속한 작업의 수는 $j = 1, 2, \cdots, m$-1에 대해서는 $\lfloor n/m \rfloor$으로 동일하다고 하였다. 여기서 $\lfloor x \rfloor$는 x보다 크지 않은 최대 정수를 의미한다. m번째 작업 패밀리에 속한 작업의 수는 나머지 작업의 수이므로 $n-(m-1)\lfloor n/m \rfloor$가 된다.

마지막으로 q_i값을 생성하기 위해서는 약간의 고려가 필요하다. 두 기계의 부하에 큰 차이가 발생하면 본 연구의 기본 취지에서 벗어나는 결과가 되므로 두 기계의 부하가 어느 정도 균형화되어 있어야 한다는 것이다. 각 작업의 크기가 0과 1사이에서 임의로 생성된 수이므로 각 배치에는 평균적으로 2개

Table 1. Performance evaluation for heuristic algorithms

n		m = 2			m = 4			m = 6			m = 8			m = 10		
		L-J	P-J	GA	L-J	P-J	GA	L-J	P-J	GA	L-J	P-J	GA	L-J	P-J	GA
50	평균	1.07	1.08	1.07	1.08	1.09	1.07	1.09	1.09	1.09	1.09	1.10	1.09	1.08	1.08	1.08
	최소	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	최대	1.17	1.20	1.17	1.18	1.18	1.16	1.19	1.20	1.19	1.19	1.19	1.18	1.19	1.19	1.19
100	평균	1.04	1.05	1.05	1.07	1.08	1.07	1.08	1.09	1.08	1.07	1.08	1.07	1.09	1.09	1.09
	최소	1.00	1.00	1.00	1.00	1.00	1.00	1.02	1.02	1.02	1.01	1.03	1.01	1.01	1.01	1.01
	최대	1.10	1.10	1.10	1.15	1.16	1.15	1.15	1.16	1.15	1.14	1.15	1.14	1.15	1.15	1.15
150	평균	1.05	1.05	1.06	1.06	1.08	1.07	1.07	1.08	1.07	1.08	1.08	1.08	1.09	1.10	1.08
	최소	1.00	1.00	1.00	1.00	1.00	1.00	1.01	1.03	1.03	1.00	1.00	1.00	1.02	1.03	1.01
	최대	1.09	1.10	1.09	1.11	1.13	1.12	1.14	1.15	1.14	1.19	1.19	1.19	1.19	1.19	1.19
200	평균	1.04	1.05	1.05	1.05	1.06	1.06	1.07	1.07	1.07	1.07	1.08	1.07	1.07	1.08	1.07
	최소	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.02	1.04	1.01	1.00	1.00	1.00
	최대	1.10	1.11	1.11	1.10	1.11	1.10	1.11	1.11	1.11	1.13	1.14	1.13	1.14	1.14	1.14

정도의 작업이 포함될 것으로 기대할 수 있다. 따라서 각 작업 패밀리에 속하는 작업의 수를 n_j라고 하면 배치처리기계의 총 작업소요시간은 $\sum_{j=1}^{m} p_j n_j/2$정도로 기대할 수 있고, 이 값을 전체 작업의 수(n)로 나눈 $T=\sum_{j=1}^{m} p_j n_j/2n$가 각 작업의 후속기계 소요시간이라면 대략적인 부하균형화가 이루어진다. 이러한 배경에 의해 q_i값은 U(T/2, 3T/2)를 통해 생성한다.

예를 들어 n = 50이고 m = 4인 조합의 경우에는 다음과 같이 데이터를 생성한다.

(1) 구간 (0, 100)에서 50개의 정수를 임의로 생성한 다음 그 값들을 100으로 나누어 r_i값으로 한다.
(2) p_1 = U(10, 20), p_2 = U(20, 30), p_3 = U(30, 40), p_4 = U(40, 50).
(3) $n_1 = n_2 = n_3 = \lfloor 50/4 \rfloor = 12$, $n_4 = 50 - 3(12) = 14$.
(4) $T = (p_1 + p_2 + p_3)(0.12) + p_4(0.14)$와 같이 계산하고 구간 ($T$/2, 3$T$/2)에서 50개의 정수를 생성하여 q_i값으로 한다.

데이터 생성 및 휴리스틱 알고리즘은 C++ 언어로 프로그램하여 메모리 512MB인 1.5GHz 노트북 PC에서 실행한 결과 모든 문제에 대해 유전 알고리즘은 5초 이내, 다른 단순 휴리스틱들은 1초 이내에 답이 구해졌다. 따라서 처리시간을 비교하는 것은 별 의미가 없을 것으로 생각된다.

각 휴리스틱의 결과는 <Table 1>에 요약하였다. 단, TFF-JS 휴리스틱은 다른 휴리스틱에 비해서 성능이 많이 떨어져 생략하였다. 이 표의 값들은 휴리스틱의 결과를 해당문제의 최적해 하한값으로 나눈 결과값이다. 따라서 1보다 작은 값은 나올 수 없으며, 만약 1이 된다면 그 해는 최적해가 된다. 예를 들어 <Table 1>의 좌상단에 위치한 n = 50이고 m = 2인 조합의 경우를 보자. 이 셀의 9개 값은 n = 50, m = 2에 대해 임의로 30개의 문제를 생성하여, 각 문제에 대해 세 가지 휴리스틱으로 스케줄을 작성하고, 그 스케줄의 최종 작업완료시간과 해당 문제의 최적해 하한값을 비교한 결과이다. 우선 LFF-JS 규칙(표에서는 L-J로 표시됨)에 의해 생성된 스케줄은 30개 문제에서 평균적으로 최적해 하한값보다 1.07배 늦은 시간에 작업을 완료하였다. 또한 최소값이 1.0이라는 것은 30개 문제 중에서 하한값과 동일한 완료시간을 갖는 경우가 있었다(다시 말해 휴리스틱의 결과가 최적해였다)는 의미이며, 최대값 1.17은 30개 문제 중에서 휴리스틱에 의한 작업완료시간이 하한값의 1.17배가 된 경우도 있었다는 것이다.

동일한 30개 문제에 대해 PFF-JS 휴리스틱(표에서는 P-J로 표시됨)을 적용한 결과는 평균이 1.08, 최소값이 1.0, 최대값이 1.20으로 LFF-JS에 비해 성능이 근소하게 떨어진다는 것을 알 수가 있다. 유전 알고리즘(표에서는 GA로 표시됨)의 결과는 평균이 1.07, 최소값이 1.0, 최대값이 1.17로 LFF-JS와 동일한 결과를 보여주고 있다.

이 결과는 다른 (n, m) 조합에서도 거의 차이를 발견할 수 없다. 평균값을 기준으로 비교하면 PFF-JS는 LFF-JS에 비해 1% 내외로 성능이 떨어진다. 반면 유전 알고리즘은 LFF-JS와 우열을 따지기가 어려울 정도로 비슷한 결과를 보여준다. 따라서 처리시간과 프로그래밍의 난이도를 감안한다면, 배치처리기계의 효율성을 최대화하도록 배치를 형성한 다음에 Johnson 규칙을 적용해 순서를 정해주는 LFF-JS 휴리스틱이 본 연구의 대상시스템에 대해 가장 좋은 결과를 보여준다. 또한 평균값을 기준으로 볼 때 그 결과값도 (n, m)의 변화에 거의 무관하게 1.04～1.09사이에 존재하는데, 비교 기준이 최적해가 아니라 최적해 하한이라는 점을 고려한다면 이 값은 이 휴리스틱의 성능이 매우 우수하다는 것을 보여주고 있다.

8. 결 론

본 연구에서는 배치처리기계와 일반적인 기계의 두 단계로 이루어진 흐름생산라인의 생산일정계획을 다루었다. 배치처리기계는 여러 개의 작업을 한 번에 처리할 수 있는데, 각 작업들은 크기가 다르고 배치로 묶여질 수 있는 작업 클래스로 구분되어 있다. 이러한 문제에 대해 우선 정수계획법 모형을 개발하고 최적해의 하한값을 제시하였다. 그러나 이 문제는 NP-hard에 속하고 현실적인 문제에서는 계획대상 작업의 수가 매우 많으므로 적절한 시간 안에 좋은 해를 구할 수 있는 휴리스틱 알고리즘을 제안하였다. 수치실험을 통해 세 가지 단순 휴리스틱 중에서는 LFF-JS 휴리스틱이 가장 우수함을 볼 수 있었다. 또한 이 휴리스틱은 유전 알고리즘의 결과와도 매우 유사한 성능을 보여주고 있는데, 알고리즘의 복잡도나 프로그래밍의 난이도가 유전 알고리즘에 비해 매우 낮다는 점을 고려한다면, 본 연구에서 제시하였던 4가지 알고리즘 중에서 가장 우수한 알고리즘이라고 할 수 있을 것이다.

향후 연구방향으로는 다음과 같은 것들을 생각할 수 있다. 우선 본 연구에서는 배치처리기계가 선행기계였지만 배치처리기계가 후행기계인 경우에 대해 관찰할 필요가 있을 것이다. 또한 본 연구에서는 최종 완료시간(makespan)을 최소화하는 문제를 다루었지만 다른 평가기준을 사용하여 문제를 해결하는 것도 필요할 것으로 생각된다.

참고문헌

Ahmadi, J. H., Ahmadi, R. H., Dasu, S., and Tang, C. S. (1992), Batching and scheduling jobs on batch and discrete processors, *Operations Research*, **40**, 750-763.

Bhatnagar, R., Chandra, P., Loulou, R., and Qiu, J. (1999), Order release and product mix coordination in a complex PCB manufacturing line with batch processors, *The International Journal of Flexible Manufacturing Systems*, **11**, 327-351.

Chandra, P. and Gupta, S. (1997), Managing batch processors to

reduce lead time in a semiconductor packaging line, *International Journal of Production Research*, **35**, 611-633.

Chandru, V., Lee, C. Y., and Uzsoy, R. (1993), Minimizing total completion time on batch processing machines, *International Journal of Production Research*, **31**, 2097-2121.

Dobson, G. and Nambimadom, R. S. (2001), The batch loading and scheduling problem, *Operations Research*, **49**, 52-65.

Garey, M. R. and Johnson, D. S. (1979), *Computers and Intractability : A Guide to the Theory of NPCompleteness* (San Francisco : W. H. Freeman).

Hochbaum, D. S. and Landy, D. (1997), Scheduling semiconductor burn-in operations to minimize total flowtime, *Operations Research*, **45**, 874-885.

Ikura, Y. and Gimple, M. (1986), Scheduling algorithms for a single batch processing machine, *Operations Research Letters*, **5**, 61-65.

Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R., and Graham, R. L. (1974), Worst-case performance bounds for simple one-dimensional packing algorithms, *SIAM Journal on Computing*, **3**, 299-325.

Johnson, S. M. (1954), Optimal two and three stage production schedules with setup times included, *Naval Research Logistics Quartly*, **1**, 61-68.

Kempf, K. G., Uzsoy, R., and Wang, C. S. (1998), Scheduling a single batch processing machine with secondary resource constraints, *Journal of Manufacturing Systems*, **17**, 37-51.

Koh, S. G., Koo, P. H., Ha, J. W., and Lee, W. S. (2004), Scheduling parallel batch processing machines with arbitrary job sizes and incompatible job families, *International Journal of Production Research*, **42**, 4091-4107.

Koh, S. G., Koo, P. H., Kim, D. C. and Hur, W. S. (2005), Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families, *International Journal of Production Economics*, **98**, 81-96.

Lee, C. Y. and Uzsoy, R. (1999), Minimizing makespan on a single batch processing machine with dynamic job arrivals, *International Journal of Production Research*, **37**, 219-236.

Lee, C. Y., Uzsoy, R., and Martin-Vega, L. A. (1992), Efficient algorithms for scheduling semiconductor burn-in operations, *Operations Research*, **40**, 764-775.

Mehta, S. V. and Uzsoy, R. (1998), Minimizing total tardiness on a batch processing machine with incompatible job families, *IIE Transactions*, **30**, 165-178.

Uzsoy, R. (1994), Scheduling a single batch processing machine with non-identical job sizes, *International Journal of Production Research*, **32**, 1615-1635.

Uzsoy, R. (1995), Scheduling batch processing machines with incompatible job families, *International Journal of Production Research*, **33**, 2685-2708.