Journal of the Society of Korea Industrial and Systems Engineering
Vol, 30, No. 2, pp.44－50, June 2007.

Rate-Modifying 활동이 있는 병렬기계의 Makespan 최소화를 위한 일정 계획

조항민 · 임승빈 · 정인재†

한양대학교 산업공학과

Parallel Machines Scheduling with Rate-Modifying Activities to Minimize Makespan

Hang-Min Cho · Seung-Bin Yim · In-Jae Jeong†

Department of Industrial Engineering, Hanyang University

This paper deals with the problem of scheduling jobs and rate-modifying activities on parallel machines. A rate-modifying activity is an activity that changes the production rate of equipment such as maintenance and readjustment. If a job is scheduled after the rate-modifying activity, then the processing time varies depending on the modifying rate of the activity. In this study, we extend the single machine problem to parallel machines problem and propose algorithms is to schedule the rate-modifying activities and jobs to minimize the makespan on parallel machines which is NP-hard. We propose a branch and bound algorithm with three lower bounds to solve medium size problems optimally. Also we develop three heuristics, Modified Longest Processing Time, Modified MULTIFIT and Modified COMBINE algorithms to solve large size problems. The test results show that branch and bound algorithm finds the optimal solution in a reasonable time for medium size problems (up to 15 jobs and 5 machines). For large size problem, Modified COMBINE and Modified MULTIFIT algorithms outperform Modified LPT algorithm in terms of solution quality.

Keywords : Makespan, Parallel Machine, Rate-modifying Activity

1. 서 론

본 논문은 병렬기계 안에서 작업 스케줄링과 rate-modifying 활동을 다루고 있다. Rate-modifying 활동은 기계의 수리(repair), 정비(maintenance), 조정(readjust) 등의 modifying 활동을 의미한다. Rate-modifying활동으로 인해 기계는 일정 시간 활동을 멈추게 되고 생산성의 변동이 일어날 수 있다. 다시 말해서 m대의 병렬기계 상황에서 각 job $j=(1,\cdots,n)$의 가공시간은 rate-modifying 이전에는 각 기계 $i=(1,\cdots,m)$에서 동일한 p_j가 되고 이후에는 $\alpha_{ij}p_j$가 된다. 여기서 α_{ij}는 job j가 기계 i마다 가지는 modifying rate이고 modifying 활동에는 시간 t_i가 소요된다. Rate-modifying 후에 할당되는 job j는 $\alpha_{ij}\geq 1$이면 가공시간이 증가하고, $\alpha_{ij}<1$이면 가공시간이 감소한다. 본 연구에서는 makespan을 최소화하는 m개의 병렬기계(P_m)에서 rate-modifying 활동(rm)과 job의 스케줄을 결정하는 문제 $P_m/rm/C_{\max}$을 다룬다[12].

이 문제의 특별한 경우인 $\alpha_{ij}=1,\ t_i=0,\ \forall i,j$은 $P_m//C_{\max}$의 문제와 동일하게 된다. 이 $m\geq 2$인 병렬기계 $P_m//C_{\max}$는 잘 알려진 *NP*-hard 문제이므로 본 연구

† 교신저자 ijeong@hanyang.ac.kr

의 $P_m/rm/C_{\max}$도 *NP*-hard 문제이다[12].

단일기계에서 수리, 정비, 조정 등을 고려한 문제는 Graves and Lee[5]가 기계의 정비와 setup time을 고려하여 total completion time, maximum lateness의 목적 함수에 대하여 최적해를 구하였다. 다음으로 Lee and Leon [9]는 rate-modifying 활동을 정의하고 이 활동이 있는 단일 기계에 대한 makespan, total completion time, total weight completion time, maximum lateness 등의 목적 함수에 대한 최적해를 구하였다. Lee and Lin[7]은 rate-modifying 활동이 있는 단일 기계에 가공시간이 resumable과 nonresumable한 경우에 대한 expected makespan, total expected completion time, maximum expected lateness, expected maximum lateness 등의 목적 함수로 갖는 문제를 연구하였다. 또한 Lee and Chen[8]은 정비 활동을 한번 해야 되는 병렬기계의 문제를 total weight completion time의 목적 함수로 하는 문제에 대하여 중간 크기까지 최적해를 구하였다. Lio, Shyur, and Lin[11]은 makespan을 목적함수로 기계가용성이 있는 2대의 병렬기계 문제의 특별한 경우에 대하여 최적해를 구하는 알고리듬을 제안하고 일반적인 2대의 병렬기계문제에 대하여 최적해를 구하는 알고리듬을 제안하고 일반적인 2대의 병렬기계 문제에 대해서는 알고리듬을 제안하였다.

Makespan을 목적함수로 하는 일반적인 병렬기계의 스케줄링은 Graham[4]가 longest processing time(LPT)으로, Coffman, Garey, and Johnson[2]은 MULTIFIT 알고리듬을 사용하여 문제를 풀었다. 이후에 Lee and Massey은 MULTIFIT와 LPT 알고리듬들을 적용한 COMBINE 알고리듬을 사용하여 문제를 풀었다[10]. 그리고 Ho and Wong은 2대의 병렬기계에서 binary search tree(lexicographic search)를 기반으로 한 알고리듬으로 최적해를 구하고 3대 이상에서는 휴리스틱을 제안하였다[6]. Mokotoff은 병렬기계문제의 최적해를 구하기 위해 exact cutting plane 알고리듬을 제안하였다[3].

본 연구에서는 Lee and Leon[9]가 다룬 단일기계 문제를 확장시켜 병렬기계에서 rate-modifying 활동의 유·무에 따라 생산성이 변화하는 makespan을 목적 함수로 하는 문제($P_m/rm/C_{\max}$)에 대하여 중간 크기 문제는 branch and bound 으로 최적해를 구하고 큰 크기 문제는 Modified LPT(MLPT), Modified MULTIFIT(MMF), Modified COMBINE(MCOM) 알고리듬들을 제안한다.

2. 문제 정의

이 문제는 병렬기계 $i=(1, 2, \cdots, n)$에서 job $j=(1, 2, j=(1, 2, \cdots, n)$와 modifying 시간 t_i을 가지는 rate-modifying 활동을 스케줄링 하는 것으로 구성되어 있다. 여기에서 각 job은 상호 연관이 없고 기계 i에서 시작을 하면 이 기계는 해당 job이 완료될 때까지 다른 작업을 할 수 없으며 setup time은 존재하지 않는다. 그리고 rate-modifying 활동은 기계 i에서 1회 이하 실행 가능하며, rate-modifying 활동이 진행 중인 기계는 modifying 시간 t_i동안 job을 가공할 수 없다. 여기에서 job j가 modifying 활동 이전에 가공되면 기계 마다 동일한 가공시간 p_j을 갖게 되며, 반면에 이후에 가공되면 기계 i에 따라 다른 modifying rate α_{ij}로 가공시간 $\alpha_{ij}p_j$을 가진다.

목적 함수는 makespan을 최소화하는 것이며 결정 사항은 rate-modifying 활동과 job의 sequence를 결정하는 것이다. 이러한 상황은 다음과 같이 표현된다.

m 전체 기계 수
n 전체 job 수
i 기계인덱스, $i\in\{1,2,\cdots,m\}$
j job 인덱스, $j\in\{1,2,\cdots,m\}$
h job sequence 인덱스
l rate-modifying 이전 job sequence 인덱스
p_j job j의 가공 시간
α_{ij} 기계 i에서 job j의 modifying rate, $\alpha_{ij}>0$
t_i 기계 i에서 rate-modifying 시간
k_i 기계 i에 할당된 job 수
C_i 기계 i에서 마지막 job의 가동이 끝나는 시간

$$x_{ijh}=\begin{cases}1, & \text{만약 job } j\text{가 기계 } i\text{에서 } h\text{번째 sequence로 가공되면}\\ 0, & \text{그렇지 않으면}\end{cases}$$

$$y_{il}=\begin{cases}1, & \text{만약 기계 } i\text{에서 } l\text{번째 job sequence 이후에 rate-modifying 활동}(rm)\text{이 진행}\\ & \text{즉, } (\cdots l\rightarrow rm\rightarrow l+1\cdots)\text{이면.}\\ 0 & \text{그렇지 않으면}\end{cases}$$

$$Minimize\ C_{\max} \quad \cdots (1)$$

$$s.t.\ C_{\max}=\max(C_1, C_2, C_3, \cdots, C_m) \quad \cdots (2)$$

$$C_i=\begin{cases}\sum_{j=1}^{n}\sum_{h=1}^{l}p_jx_{ijh}+y_{il}(t_i+\sum_{j=1}^{n}\sum_{h=l+1}^{k_i}\alpha_{ij}p_jx_{ijh}), & 0<l<k_i \quad \cdots (3)\\ \sum_{j=1}^{n}\sum_{h=1}^{k_i}p_jx_{ijh}, & l=k_i \quad \cdots (4)\\ t_i+\sum_{j=1}^{n}\sum_{h=1}^{k_i}\alpha_{ij}p_jx_{ijh}, & l=0 \quad \cdots (5)\end{cases}$$

$$\sum_{i=1}^{m}\sum_{h=1}^{k_i}x_{ijh}=1, \quad \forall j \quad \cdots (6)$$

$$\sum_{l=1}^{k_i} y_{il} = 1, \qquad \forall j \quad \cdots\cdots (7)$$

$$l \in \{0, 1, \cdots, k_i\} \quad \cdots\cdots (8)$$

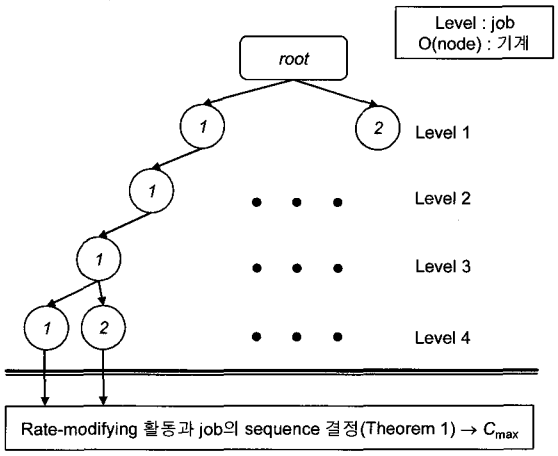
제약식 (3)은 rate-modifying 활동의 진행이 기계 i에 할당된 job sequence의 중간에 위치하면 rate-modifying 이전에 할당된 job들은 p_j을 가지며 그 다음 rate-modifying 활동이 t_i동안 진행되고 이후에 할당된 job들은 $\alpha_{ij} p_j$을 가진다. 그리고 제약식 (4)은 rate-modifying 활동이 기계 i에 할당된 마지막 job 이후에 위치하면 여기의 job들은 모두 rate-modifying 활동 이전에 할당됨으로 p_j을 가짐을 나타낸다. 다음으로 제약식 (5)은 rate-modifying 활동이 기계 i에서 job을 가공하기 전에 위치하면 여기의 job들은 모두 rate-modifying 활동 이후에 할당됨으로 먼저 rate-modifying 활동이 t_i동안 진행되고 그 다음 job들은 $\alpha_{ij} p_j$을 가짐을 나타낸다. 제약식 (6)은 job j가 하나의 기계에 하나의 sequence에만 할당됨을 나타내고 제약식 (7)은 기계 i에서 rate-modifying 활동은 1회 위치함을 나타낸다. 제약식 (8)은 rate-modifying 활동이 위치하는 sequence의 범위를 나타낸다.

3. 제안 알고리듬

3.1 branch and bound

본 문제의 해결 방법은 다음과 같다. 중간 크기 문제는 branch and bound(B&B) 알고리듬으로 최적해를 찾는다. B&B는 최적해를 구하는 열거법(enumeration method)의 일종으로 가능해를 열거해 가면서 최적해를 찾아 나가는 방법이다. 이때에 모든 가능해를 검색하기 위해서는 계산 시간이 소요되기 때문에 전체 문제를 상대적으로 쉬운 문제로 분할하는 분지(branch)를 하고 상한(upper bound : UB)와 하한(lower bound : LB)을 준다. 만약 임의의 노드(node)에서 구한 하한이 상한보다 안 좋은 값을 가지면 더 이상 분지하지 않아도 이 sub-tree의 가장 좋은 해는 상한보다 좋은 값을 갖지 못하므로 이후의 sub-tree를 소거(pruning) 하여 계산 시간을 줄일 수 있게 된다[1, 12].

B&B는 job 4개와 기계 2대의 예제로 보면 <그림 1>과 같이 분지하고 깊이 우선 탐색(Depth First Search)의 검색 방법으로 마지막 레벨까지 분지하여 job들이 기계에 모두 할당되면, 즉 기계에 따른 부분 문제로 분할되면 이때 각 기계의 C_i은 다음의 <Theorem 1>으로 rate-modifying 활동과 job의 sequence를 결정하여 $C_{\max}$을 구한다[1, 10].



<그림 1> Branch and bound 알고리듬의 tree 구조 예

<Theorem 1> Lee and Leon[9]. u_i는 기계 i에 할당된 job 집합이고 $v_i = \{j \in u_i : \alpha_{ij} < 1\}$이면 기계 i에서 rate-modifying 활동(rm)로 단축할 수 있는 시간(save time) $s_i = \sum_{j \in v_i} p_j (1 - \alpha_{ij})$이다. 그러면 최적의 sequence는

만약 $s_i \le t_i$이면 rm없이 u_i job 순서가 최적 스케줄, ... (9)

만약 $s_i > t_i$이면 $(u_i - v_i) \rightarrow (rm) \rightarrow (v_i)$이 최적 스케줄 ..... (10)

이다. ■

식 (9)는 rate-modifying 활동으로 단축할 수 있는 시간 s_i가 해당 기계의 modifying 시간 t_i보다 작거나 같은 경우로 rate-modifying 활동 없이 sequence를 가지는 것이고 식 (10)은 modifying 시간 t_i보다 큰 경우, 즉 효율이 발생하는 경우로 $u_i - v_i$을 먼저 할당하고 다음으로 rate-modifying 활동을 진행하며 마지막으로 v_i을 할당하는 것이다.

상한(UB)은 1개의 기계에 모든 job이 rate-modifying 활동 이전에 할당된 가능해를 초기(initial) 값으로 한다. 이것은 $initial\ UB = \sum_{j=1}^{n} p_j$이고 검색된 가능해 중에서 작은 해로 갱신시켜 나간다. 그리고 하한(LB)은 다음의 3가지를 사용하였다. 각 노드에서 현재까지 각 기계 i에 할당된 job들의 집합은 u_i'이고 $v_i' = \{j \in u_i' : \alpha_{ij} < 1\}$라면 기계 i에서 rate-modifying 활동으로 단축할 수 있는 시간(saving time), 즉 이 활동 이후에 할당된 job들에 따른 단축 시간은 $s_i' = \sum_{j \in v_i'} p_j (1 - \alpha_{ij})$이다. 그러면

$$LB1 = \max_i \sum_{j \in u_i} \min(p_j, \alpha_{ij} p_j) \quad \cdots\cdots (11)$$

$$LB2 = \max_i \begin{cases} \sum_{j \in u_i'} \min(p_j, \alpha_{ij} p_j), & s_i' \le t \quad \cdots\cdots (12) \\ \sum_{j \in (u_i' - v_i')} p_j + t_i + \sum_{j \in v_i'} \alpha_{ij} p_j, & s_i' > t \quad \cdots\cdots (13) \end{cases}$$

이다. 식 (11)의 하한 1($LB1$)은 현재 노드까지 할당된 job들이 가질 수 있는 최소 가공시간을 사용하는 것이다. 하한 2($LB2$)의 식 (12)은 rate-modifying 활동으로 인한 효과가 없는 것으로 job들이 가질 수 있는 최소 가공시간을 사용하는 것이고 식 (13)은 rate-modifying 활동으로 인한 효과가 있는 것으로 $u_i' - v_i'$을 먼저 할당하고 다음으로 rate-modifying 활동을 진행하며 마지막으로 v_i'을 할당하는 것이다. 그리고 $LB2$의 진행으로 만들어진 각 기계 i의 현재 시점 makespan을 c_i, 현재 노드까지 미 할당된 job들의 집합을 w라 하고 $\theta = \sum_{j \in w} \min(p_j, \min_i$ 라 정의하면

$$LB3 = LB2 + \max\left((1/m)\left(\theta - \sum_{i=1}^{m}(LB2 - c_i)\right), 0\right) \quad \cdots\cdots (14)$$

이다. 식 (14)의 하한 3($LB3$)은 현재 노드까지 할당된 job들로 $LB2$을 구하고 여기에 미 할당된 job들을 포함하는 것이다.

3.2 휴리스틱

큰 크기의 문제는 3개의 휴리스틱들로 가능해를 구한다. 이 휴리스틱들은 가공시간이 큰 순서로 정렬한 내림차순 리스트 J을 사용하여 알고리듬을 진행한다. 그리고 가공시간이 변화하는 상황이므로 다음의

$$\overline{\alpha_j} = (1/m) \sum_{i=1}^{m} \min(\alpha_{ij}, 1), \quad \forall j \quad \cdots\cdots (15)$$

$$p_j' = \overline{\alpha_j} p_j \quad \cdots\cdots (16)$$

을 사용하여 내림차순 리스트 $J = \{p_1' \ge p_2' \cdots \ge p_n'\}$을 만든다. 또한 이 휴리스틱들의 각 기계 i의 현재 시점 makespan c_i은 <Theorem 1>으로 구한다. 현재까지 각 기계 i에 할당된 job들의 집합을 u_i'라고 하고 $v_i' = \{j \in u_i' : \alpha_{ij} < 1\}$라 하면 기계 i에서 rate-modifying 활동으로 단축할 수 있는 시간은 $s_i' = \sum_{j \in u_i'} p_j(1 - p_j)$이다. 그러면,

$$c_t = \begin{cases} \sum_{j \in u_i'} p_j, & s_i' \le t \quad \cdots\cdots (17) \\ \sum_{j \in (u_i' - v_i')} p_j + t_j + \sum_{j \in v_i'} \alpha_{ij} p_j, & s_i' > t \quad \cdots\cdots (18) \end{cases}$$

이다. 먼저 Modified LPT(MLPT)은 LPT를 변형한 것이다. LPT은 job을 내림차순으로 정렬하여 list scheduling으로 기계에 할당하는 알고리듬이다[4]. 이 알고리듬에 식 (15), 식 (16)로 내림차순 리스트 J을 만들고 식 (17), 식 (18)의 c_i로 기계에 job을 할당 시에 가장 먼저 유휴가 발생하는 기계를 찾는다. 이 알고리듬은 다음의 단계를 따른다.

[MLPT]
단계 0. 리스트 J 생성, $u_i' = v_i' = \{\}, \forall i$.
단계 1. 각 기계의 현재 시점 makespan c_i을 구함.
단계 2. $j^* = \max_j$
$i^* = \max_i$ 라 하자.
$u_{i^*}' = u_{i^*}' + \{j^*\}$.
만약 $\alpha_{i^*j^*} < 1$이면 $v_{i^*}' = v_{i^*}' + \{j^*\}$이고,
$J = J - \{j^*\}$.
단계 3. 만약 $J = \{\}$이면 알고리듬 종료.
만약 $J \neq \{\}$이면 단계1로 돌아감. ■

다음으로 Modified MULTIFIT(MMF)은 n개의 job들을 m개의 bin들에 이분 탐색(bisection search)로 넣어가며 최소용량을 찾아가는 bin-packing 방법을 병렬기계에 적용한 MULTIFIT을 기본 구조로 한다[2, 11]. 이 방법은 job들을 내림차순 리스트 J로 정렬 후 인덱스가 낮은 bin에 먼저 넣어가는 First Fit(FF)을 적용한 First Fit Decreasing(FFD)을 사용하여 가능해를 빠르게 찾는다. 그리고 이분 탐색을 할 때 사용되는 하한(LB)과 상한(UB)의 중앙값인 bin의 용량, 즉 기계의 용량은 W이고, 이것은 $W = (LB + UB)/2$로 구한다. 여기서

$\mu = (1/m) \sum_{j=1}^{n} \min(p_j, \min_i \alpha_{ij} p_j)$,
$\lambda = \max_j (\min(p_j, \min_i \alpha_{ij} p_j))$라 정의하면 W의초기 LB과 UB은

$$initial\ LB = \max(\mu, \lambda) \quad \cdots\cdots (19)$$

$$initial\ UB = \max\left((2/m) \sum_{j=1}^{n} p_j, \max_j \right. \quad \cdots\cdots (20)$$

로 구한다[2].

임의의 iteration I에서 식 (15), 식 (16)로 만든 내림차

순 리스트 J의 n개의 job들이 W 상한 기준 내에 m대의 기계에 할당되면, 즉 가능해를 구하면 UB을 W의 값으로 변경하여 LB와 간격을 줄인다. 반대로, 할당되지 않고 m대의 기계를 초과하면 LB을 W의 값으로 변경하여 UB와 간격을 줄인다. 그리고 다음 iteration을 진행한다. 종료는 LB와 UB가 같아질 때 알고리듬을 빠져나온다. 여기서 FFD의 진행은 본 문제의 가공시간이 변동할 수 있기 때문에 job을 기계에 할당 해보고 식 (18), 식 (19)로 c_i을 계산하여 할당 가능한지 확인한다. 이 방법으로 job들을 W의 용량의 기계에 모두 할당 했을 때 필요한 기계 대수를 $FFD[J, W]$라 표현하면 다음의 단계를 따른다.

[MMF]
단계 0. 리스트 J 생성.
초기 LB과 UB을 구함
Iteration(I) = 1
단계 1. $W=(LB+UB)/2$.
단계 2. J을 사용하여 FFD를 적용하여 $FFD[J, W]$ 구함.
만약 $FFD[J, W] \le m$이면 $UB \leftarrow W$이고,
만약 $FFD[J, W] > m$이면 $LB \leftarrow W$이고.
단계 3. 만약 $LB = UB$이면 알고리듬을 종료,
만약 $LB \neq UB$이면 $I = I+1$, 단계 1로 돌아감. ■

마지막으로 Modified COMBINE(MCOM)은 MMF의 초기 상한(UB)을 좀 더 좋은 값으로 하여 가능해를 좋게 만드는 알고리듬이다. 이 방법은 초기 UB을 MLPT의 가능해로 사용한다. 그리고 그 외 단계는 MMF의 방법과 동일하다[10].

4. 실험 결과

Rate-modifying 활동은 병렬기계에서 발생할 수 있는 일반적인 상황을 고려해야 됨으로 실험을 다양한 조건에서 실행한다. 문제는 <표 1>과 같이 일양(Uniform) 분포에서 랜덤하게 발생시킴으로써 다양한 조건에서 문제를 실험하도록 한다. 또한 랜덤 발생 범위가 작을 때(small variance)와 클 때(large variance)로 상황으로 구분하여 실험 하도록 한다.

그리고 실험 문제의 크기는 <표 2>와 같이 job의 개수(n)와 기계의 대수(m)를 작은 크기에서부터 큰 크기의 문제까지 실험한다. 문제 크기에 따른 알고리듬의 사용은 job 15개와 기계 5대까지인 중간 크기까지는 branch and bound(B&B) 알고리듬으로 최적해를 구하고 휴리스틱들로 가능해를 구하였다. 그리고 큰 크기 문제는 최적해를 구할 시 많은 계산 시간이 소요되어 휴리스틱들로 빠른 시간 내에 가능해만 구하였다.

<표 1> 실험 데이터 생성

	small variance	large variance
가공시간 (p_j)	$U[5, 10]$	$U[5, 20]$
Modifying 시간 (t_i)	$U[0.5, 1]\times(1/n)\sum_{j=1}^{n} p_j$	$U[1, 1.5]\times(1/n)\sum_{j=1}^{n} p_j$
Modifying rate (α_{ij})	$U[0.5, 1.5]$	$U(0, 2]$

<표 2> 실험 문제 크기

기계(m)	Job(n)
2	5
3	10
4	15
5	15
10	50
50	500
100	1000

알고리듬의 구현 프로그래밍 언어는 C++을 사용하였고 실험한 컴퓨터는 PC Intel(R) 펜티엄 4, 3GHz, DDR 512RAM에서 실험을 하였다. 그리고 계산 시간(CPU Time)의 단위는 초(second)로 측정하였다. 실험은 <표 2>의 문제 크기 마다 <표 1>의 조합인 8가지 조건에서 각 20회씩 실험을 하였다. 따라서 1개의 문제 크기에 160회(20×8)를 실시하였다. <표 3>은 $\forall p_j$, t_j, α_{ij}에 대한 평균 실행 시간(seconds)에 관한 결과이다.

<표 3> $\forall p_j$, t_j, α_{ij}에 대한 평균 실행시간

		CPU Time (seconds)					
		B&B			MLPT	MMF	MCOM
기계	Job	$LB1$	$LB2$	$LB3$			
2	5	0	0	0	0	0	0
3	10	0.10	0.10	0.10	0	0	0
4	15	424.43	393.14	387.59	0	0	0
5	15	4815.16	4384.82	4382.91	0	0	0
10	50	-	-	-	0.00	0.00	0.00
50	500	-	-	-	0.12	0.16	0.25
100	1000	-	-	-	0.52	0.66	1.11

중간 크기 문제까지 최적해를 구하는 B&B는 기계 5

대와 job 15개 문제를 평균 4382초에 하한3($LB3$)가 최적해를 가장 빠른 시간에 찾는 좋은 한계(bound)이고 다음으로 $LB2$, $LB1$ 순으로 합리적인 시간 안에 최적해를 찾는 한계이다. 그리고 큰 크기의 문제에서는 3가지 휴리스틱이 모두 1초 내·외로 가능해를 찾았다. 여기서 Modified MULTIFIT(MMF)보다 Modified COMBINE (MCOM)이 초기 상한을 Modified LPT(MLPT)의 가능해를 사용하기 때문에 MLPT의 계산 시간만큼 더 소요된다.

다음은 해(solution)를 비교한 것으로 중간 크기 문제인 기계 5대와 job 15개까지의 문제는 B&B로 구한 최적해와 휴리스틱들로 구한 가능해 간의 % 편차를 비교하였다. 최적해는 Z^*이고 MLPT의 가능해는 Z^{MLPT}, MMF의 가능해는 Z^{MMF}, MCOM의 가능해는 Z^{MCOM}로 표현한다. 이 문제의 $\forall p_j, t_j, \alpha_{ij}$에 대한 해를 최적해 기준으로 비교한 결과는 <표 4>와 같다.

$$PD_{MLPT} = (Z^{MLPT} - Z^*)/Z^* \times 100\%$$
$$PD_{MMF} = (Z^{MMF} - Z^*)/Z^* \times 100\%$$
$$PD_{MCOM} = (Z^{MCOM} - Z^*)/Z^* \times 100\%$$

기계 5대와 job 15개의 문제에서 MLPT가 평균 28.12%, MMF가 평균 19.27%, MCOM이 평균 18.00%로 최적해와 차이가 있다. 그리고 <표 5>는 이 문제의 $\forall p_j$, t_j, α_{ij}에 대한 해를 유의 수준 $\alpha = 0.05$에서 T 검정 결과 p-값이다. 이 결과로 MCOM, MMF은 MLPT와 비교하여 차이가 있음을 알 수 있고 MCOM은 MMF와 비교하여 차이가 있다고 볼 수 없다. 따라서 MCOM과 MMF가 MLPT보다 더 좋은 가능해를 찾는 알고리듬들임을 알 수 있다.

<표 4> 중간 크기 문제의 해 % 편차 비교

기계	Job		PD_{MLPT}	PD_{MMF}	PD_{MCOM}
2	5	max	47.55%	47.18%	40.27%
		average	8.89%	4.44%	3.32%
		min	0.00%	0.00%	0.00%
3	10	max	84.68%	51.40%	51.40%
		average	20.03%	11.58%	10.33%
		min	0.00%	0.00%	0.00%
4	15	max	136.79%	73.16%	73.16%
		average	30.66%	18.54%	18.11%
		min	1.92%	0.00%	0.00%
5	15	max	104.92%	66.59%	66.59%
		average	28.12%	19.27%	18.00%
		min	0.00%	0.00%	0.00%

<표 5> 유의 수준 $\alpha = 0.05$에서 T 검정 결과 p-값

기계	Job	MLPT : MMF	MLPT : MCOM	MMF : MCOM
2	5	0.264	0.177	0.810
3	10	0.026	0.013	0.770
4	15	0.006	0.002	0.930
5	15	0.059	0.024	0.714

큰 크기 문제는 MLPT로 구한 가능해와 그 외 휴리스틱들로 구한 가능해 간의 % 편차를 비교하였다. 이 문제의 $\forall p_j$, t_j, α_{ij}에 대한 해를 MLPT의 가능해와 비교한 결과는 <표 6>와 같다.

$$PD_{MLPT-MMF} = (Z^{MLPT} - Z^{MMF})/Z^{MLPT} \times 100\%$$
$$PD_{MLPT-MCOM} = (Z^{MLPT} - Z^{MCOM})/Z^{MLPT} \times 100\%$$

<표 6> 큰 크기 문제의 해 % 편차 비교

기계	Job		$PD_{MLPT-MMF}$	$PD_{MLPT-MCOM}$
10	50	max	36.69%	38.84%
		average	10.78%	10.73%
		min	-4.01%	0.00%
50	500	max	37.77%	36.70%
		average	15.78%	16.00%
		min	0.00%	0.00%
100	1000	max	37.41%	36.88%
		average	16.76%	16.76%
		min	-0.10%	0.00%

<표 7> 유의 수준 $\alpha = 0.05$에서 T 검정 결과 p-값

기계	Job	MLPT : MMF	MLPT : MCOM	MMF : MCOM
10	50	< 0.0001	< 0.0001	0.979
50	500	< 0.0001	< 0.0001	0.992
100	1000	< 0.0001	< 0.0001	0.999

기계 500개와 기계 50대의 문제에서 MMF가 평균 15.98%, MCOM이 평균 16.00%로 MLPT의 가능해와 차이가 있다. 그리고 기계 100대와 job 1000개의 문제에서 MMF가 평균 16.75%, MCOM이 평균 16.76%로 MLPT의 가능해와 차이가 있다. 여기서 MCOM은 MLPT의 가능해를 한계(bound)로 갖는 좋은 가능해를 찾았다. 그리고 <표 7>은 이 문제의 $\forall p_j$, t_j, α_{ij}에 대한 해를 유의 수준 $\alpha = 0.05$에서 T 검정 결과 p-값이다. 여기에서 '< 0.0001'은 0.0001보다 작은 값이다. 이 결과로 MCOM, MMF은 MLPT와 비교하여 차이가 있음을 알 수 있고 MCOM은

MMF와 비교하여 차이가 있다고 볼 수 없다. 따라서 MCOM과 MMF가 MLPT 보다 더 좋은 가능해를 찾는 알고리듬들임을 알 수 있다.

5. 결 론

본 연구는 *NP*-hard 문제인 rate-modifying 활동이 있는 병렬기계의 makespan 최소화($P_m/rm/C_{\max}$)하는 알고리듬을 제안하였다.

Branch and bound(B&B) 알고리듬은 3개의 하한(LB)을 제안하여 기계 5대와 job 15개까지인 중간 크기의 문제에 대해 합리적인 시간 안에 최적해를 찾았다. 그리고 더 큰 크기의 문제에 대해 Modified LPT(MLPT), Modified MULTIFIT(MMF), Modified COMBINE(MCOM) 알고리듬등을 제안하여 짧은 시간 안에 좋은 가능해를 찾았다. 실험 결과로 B&B는 $LB3$가 가장 좋은 한계이며 다음으로 $LB2$, $LB1$ 순으로 최적해를 찾는 좋은 한계이다. 그리고 휴리스틱은 MCOM과 MMF가 MLPT보다 더 좋은 결과를 보였다. 여기서 MCOM은 MLPT의 가능해를 한계로 갖는 좋은 가능해를 찾았다.

추후 연구로는 B&B를 개선하여 좀 더 큰 문제를 해결할 수 있게 한다. 또한, 오차 한계(error bound)를 보증하는 해법을 찾아서 하한을 개선하여 효과적으로 해를 찾게 하며 최적해를 구하기 어려운 큰 크기 문제의 해 비교 시에 활용할 수 있도록 하는 것이다.

참고문헌

[1] Baker, K. R.; *Introduction to Sequencing and Scheduling*, 1943～, John Wiley & Sons Inc, 1974.

[2] Coffman Jr., E. G. and, Garey, M. R. and Johnson, D. S.; "An Application of Bin-Packing to Multiprocessor Scheduling," *SIAM Journal on Computing*, 7(1) : 1-17, 1978.

[3] Ethel Mokotoff; "An Exact Algorithm for the Identical Parallel Machine Scheduling Problem," *European Journal of Operation Research*, 152 : 758-769, 2004.

[4] Graham, R. L.; "Bounds on Multiprocessing Timing Anomalies," *SIAM Journal on Applied mathematics*, 17(2) : 416-429, 1969.

[5] Graves, G. H. and Lee, C.-Y.; "Scheduling Maintenance and Semi-resumable Jobs on a Single Machine," *Naval Research Logistics*, 46 : 845-863, 1999.

[6] Ho, J. C. and Wong J. S.; "Makespan Minimization for m Parallel Identical Processors," *Naval Research Logistics*, 42 : 935-948, 1995.

[7] Lee, C. Y. and Lin, C. S.; "Single-Machine Scheduling with maintenance and repair rate-modifying activities," *European Journal of Operation Research*, 135 : 493-513, 2001.

[8] Lee, C.-Y. and Chen, Z. L.; "Scheduling of Jobs and Maintenance Activities on Parallel Machines," *Naval Research Logistics*, 47 : 145-165, 2000.

[9] Lee, C.-Y. and Leon, V. J.; "Machine Scheduling with a Rate-Modifying Activity," *European Journal of Operation Research*, 128 : 119-128, 2001.

[10] Lee, C.-Y. and Massey, J. D.; "Multiprocessor scheduling: Combining LPT and MULTIFIT," *Discrete Applied mathematics*, 20 : 233-242, 1988.

[11] Liao, C. J. and Shyur, D. L., and Lin, C. H.; "Makespan Minimization for Two Parallel Machines with an Availability Constraint," *European Journal of Operation Research*, 160 : 445-456, 2005.

[12] Pinedo, M.; *Scheduling : Theory, Algorithms, and Systems*, 2nd Edition, Prentice-Hall, Englewood Cliffs, NJ, 2002.