Journal of the Korean Institute
of Industrial Engineers
Vol.22, No.1, March, 1996



# 순열 flowshop 스케줄링에서의 평균 flowtime 최소화를 위한 경험적 알고리듬

## A heuristic algorithm for mean flowtime minimization in permutation flowshop scheduling

우훈식*, 임동순**, 이완규***

Hoon-Shik Woo*, Dong-Soon Yim**, Wan-Kyu Lee***

**Abstract**

Based on a job insertion method, we developed a heuristic algorithm for the mean flowtime objective in a permutation flowshop environment. The simulation experiments are implemented to evaluate the effectiveness of the proposed algorithm against the existing heuristics. The experiments reveal the superiority of the proposed algorithm to other heuristics especially when the ratio of the number of jobs and number of machines is greater than or equal to two.

## 1. 서론

일반적으로 flowshop은 주어진 $n$개의 작업을 정해진 작업 순서에 의해 일련의 $m$개 기계에서 처리하는 생산 시스템을 말한다[3,18]. 컴퓨터및 생산 시스템에서 예를 찾아볼 수 있는 이러한 시스템에서 전체 가능한 스케줄의 수는 $n!$이며 이로 인하여 순열 flowshop이라 불린다. 순열 flowshop에서의 목표는 시스템의 특정 성능을 최적화하는 작업의 특정 순열 즉 최적 스케줄을 구하는 것이다.

순열 flowshop 스케줄링 문제는 지난 40여년간 스케줄링 분야에서 연구가 활발한 주제 중의 하나이다[15]. 잘 알려진 Johnson[12]의 2-기계 문제에 대한 makespan 최소화 알고리듬의 발표이후, 대부분의 연구는 $m$-기계

* 시스템공학연구소
** 한남대학교 산업공학과
*** 한국표준과학연구원

에서의 makespan을 최소화하는 연구에 치중되었다. 하지만, 기계 수가 2보다 큰 경우 Rinooy Kan[21]에 의해 makespan문제가 NP complete의 특성을 갖는다고 증명되었으며, 이로 인하여 작업의 수가 많은 경우 기존의 최적화 기법은 실용적이지 못하다. 따라서, 대안으로 경험적 방법론이 Campbell et al.[6], Dannenbring[7], Nawaz et al.[17], Ho & Chang[10] 등에 의해 제시되었으며, 현재까지는 Nawaz et al.의 NEH 알고리듬이 가장 우월한 결과를 제공한다고 알려져 있다[23,24].

평균 flowtime 최소화를 위한 스케줄링은, 2-기계 문제에 대한 Ignall & Schrage[11]의 branch & bound 방법을 효시로 볼 수 있다. 뒤를 이어 Bansal[4], Adiri & Pohoryles[1], Szwarc[22], 그리고 Ahmadi & Bagchi[2] 등의 연구가 진행되었다. 하지만, 평균 flowtime 최소화의 경우도 Garey et al.[8]에 의하여 2-기계 문제 조차 NP complete 임이 증명되었다. 따라서, 기존의 최적화 방법보다는 경험론적 방법에 기초한 연구가 주로 수행되었으며, 대표적인 연구는 Gupta[9], Miyazaki et al.[16], Rajendran & Chaudhuri[20], Rajendran[19]등을 들 수 있다. 최근에 Rajendran [19]은 Nawaz et al.의 NEH 알고리듬을 변형시켜 작업간의 경험적인 선호관계를 구성, 이를 부분 스케줄에 삽입함으로써 최종 스케줄을 생성하는 알고리듬을 개발하였으며, 기존의 알고리듬보다 우월하다고 발표하였다.

Makespan 과 평균 flowtime 최소화 기준에서 각각 최강의 결과를 생성한 NEH 알고리듬과 Rajendran의 알고리듬은 traveling salesman 문제 해법으로 사용되었던 Karg & Thompson의 도시 삽입 방법[13]을 변형한 것이다[19]. 본 연구에서는 경험론적 방법에 기초한 이러한 기존의 두 알고리듬을 일반화한 알고리듬을 제안하고, 제안된 알고리듬과 기존의 알고리듬들 간의 성능평가를 수행하였다. 즉, 평균 flowtime에 대하여, 기존의 연구에서 우수한 알고리듬이라고 알려진 CDS, NEH, 그리고 Rajendran의 방법과 비교하였다. 평균 flowtime은 기계 이용율의 평준화, 최종 작업물과 재공 재고의 회수율등과 같은 생산 활동에 있어서의 중요한 목표를 대변하는 성능 평가 수단이다. 또한, 일반적인 시각에서, 한쪽 성능 기준에 우월한 알고리듬이 다른 성능 기준에 대하여도 좋은 결과를 생성할 것으로 기대된다. 따라서 이들 알고리듬을 다른 기준에서 비교 분석하는 것 또한 의미가 있을 것이다.

## 2. 기존의 알고리듬 고찰

본 장에서는 makespan과 평균 flowtime 최소화를 위하여 고안된 기존의 알고리듬을 고찰한다. 고려된 flowshop은 주어진 $m$개의 기계가 $j=1,2,\cdots,m$으로 구성되며, 작업은 $J_i$, $i=1,2,\cdots,n$으로 구성된다. 또한 작업 $J_i$의 기계 $j$에서의 작업시간은 $p(i,j)$로 정의되었다.

고려된 문제는 Maccarthy & Liu[15]의 표기법을 따르면 $n/m/F,Perm/\overline{F}$의 형태를 따르는 문제이다. 주어진 작업 스케줄이 $J_{i_1}J_{i_2}\cdots J_{i_n}$이라 하고 기계 $j$에서의 작업 $J_i$의 시작시각을 $S(i,j)$, 종료시각을 $T(i,j)$라 하자. 그러면, 종료시각과 시작시각의 관계는 다음과 같다.

$$T(i,j)=S(i,j)+p(i,j) \qquad (1)$$

또한, 기계 $j$에서의 작업 $J_i$의 최단 시작시각은

$$S(i,j) = \max\{T(i,j-1), T(i-1,j)\} \quad (2)$$

이며, 이것은 작업 $J_i$는 기계 $j-1$에서 작업이 끝나기 전에는 기계 $j$에서 작업을 시작할 수 없으며, 또한 이전 작업 $J_{i-1}$이 기계 $j$에서 끝나야만 하는 것을 의미한다. 즉, 작업 $J_i$와 기계 $j$가 모두 준비되어야 하는 것이다. 주어진 관계 (1)과 (2)를 반복적으로 수행하여 다음의 계산 순서로 알고리듬이 진행된다.

$T(1,1), T(1,2), \cdots, T(1,m), T(2,1), \cdots, T(2,m), \cdots, T(n,1), \cdots, T(n,m)$.

시스템 성능 측정 도구인 평균 flowtime은 다음과 같이 정의된다.

$$\sum_{i=1}^{n} T(i,m)/n.$$

### CDS 알고리듬

Campbell, Dudek, & Smith[6]에 의해 고안된 CDS 알고리듬은 알고리듬 간의 성능 비교를 위한 표준으로 자주 사용되는 알고리듬이다. CDS 알고리듬은 $m$-기계 문제를 $(m-1)$개의 2-기계 문제로 분리하여 Johnson의 알고리듬을 적용함으로써 생성된 결과 즉, $(m-1)$개의 스케줄을 $m$-기계 문제에 다시 적용하여 그 중 가장 좋은 스케줄을 선택하는 것이다. Johnson의 알고리듬을 적용하기 위한 2-기계 문제에서의 작업 시간은 다음과 같이 정의된다.

기계 1: $\sum_{j=1}^{k} p(i,j)$

기계 2: $\sum_{j=m-k+1}^{m} p(i,j)$

단, $k=1,2,\cdots,m-1$.

### NEH 알고리듬

Nawaz et al.[17]에 의해 제안된 알고리듬은 makespan 최소화 기준에 대하여 최강으로 알려져 있다. 기본적인 개념은 전체 작업 시간을 내림차순으로 정렬하여 이를 작업의 삽입 순서로 정하여 그 순서에 따라서 기존의 부분 스케줄에 삽입하여 가장 좋은 스케줄을 찾아 나가는 방법이다. 알고리듬은 다음과 같다.

단계 1. 각각의 작업에 대하여 전체 작업 시간을 계산한 후 내림차순으로 정렬.

단계 2. K=2
정렬된 순서에서 처음 두 작업을 선택하여 최적의 결과를 생성하는 해 선택 (현재해로 설정).

단계 3. K=K+1
정렬된 순서에서 K번째 작업을 현재해에 대하여 K개의 slot에 삽입하여 K개의 후보해를 생성후, 최적의 결과를 생성하는 해 선택 (현재해로 설정).

단계 4. 만약 K = $n$이면, 정지.
그렇지 않으면, 단계 3으로 이동.

### Rajendran의 알고리듬

Rajendran[19]에 의해 제안된 알고리듬은 Nawaz et al.의 NEH 알고리듬을 변형시켜서 평균 flowtime 최소화 목표에 적합하도록 재구성한 것이다. NEH 알고리듬과 차별되는

Rajendran 알고리듬의 특징은 작업의 삽입 순서와 작업 삽입의 경우의 수에 있다. Rajendran 알고리듬의 작업 순서는 NEH 알고리듬에서의 총 작업시간에 따른 내림차순과는 달리 가중치를 적용한 총 작업시간의 오름차순으로 정하였으며, 또한 작업 삽입 경우의 수를 제한하여 알고리듬의 속도를 향상시켰다. Rajendran은 기존의 Gupta [9], Miyazaki [16], 그리고 Ho & Chang [10]의 알고리듬과 비교, 평균flowtime 목표에서 우월하다고 주장하였다. Rajendran의 알고리듬은 다음과 같다.

단계 1. 각각의 작업에 대하여 가중 작업 시간을 계산한 후 오름차순으로 정렬.

단, 가중 작업 시간 $T_i = \sum_{j=1}^{m}[(m-j+1)p(i,j)]$.

단계 2. K=1

정렬된 첫 작업을 선택하여 부분 스케줄 생성 (현재해로 설정).

단계 3. K=K+1

정렬된 K번째 작업을 현재해의 정해진 위치 $\rho$에 삽입하여 후보해를 생성한 후, 최적의 결과를 생성하는 해 선택 (현재해로 설정).

단, 삽입 위치 $\rho$, $\text{int}[K/2] \leq \rho \leq K$.

단계 4. 만약 K = $n$ 이면, 정지

그렇지 않으면, 단계 3으로 이동.

## 3. 제안 알고리듬

본 장에서는 경험론적 방법에 기초하여 기존의 NEH 알고리듬과 Rajendran의 알고리듬을 일반화한 알고리듬을 제안한다. 전 장에서 설명한 NEH 알고리듬과 Rajendran의 알고리듬은 traveling salesman 문제 해법으로 사용되었던 Karg & Thompson의 도시 삽입 방법[13]을 변형한 것이다[19]. 본 논문에서 제안된 알고리듬도 또한 Karg & Thompson의 알고리듬을 기초로 하고 있다.

전 장에서 살펴본 기존 알고리듬들의 제한은 부분 알고리듬에 삽입할 작업을 특정 기준에 의하여 미리 정한다는 것에 있다. NEH 알고리듬의 경우는 총 작업 시간의 내림차순으로 결정하였으며, Rajendran의 경우는 가중치를 고려한 총 작업 시간의 오름차순으로 결정하였다. 1-기계 문제등을 다룬 기존의 알고리듬에서 유래한 이유를 들어서 이러한 기준이 설정되었지만, 일반적인 기준이라고 생각하기는 어렵다. 본 연구에서 제안된 알고리듬의 기본적인 개념은 이러한 기존 알고리듬의 비일반성을 극복하는 것이다. 즉, 부분 스케줄에 삽입할 작업을 미리 순서화하지 않고, 부분 스케줄에서 제외된 잔류 작업 모두에 대하여 후보해를 생성하여 그 중 최적의 결과를 생성하는 작업을 선택하는 것이다. 이렇게 특정 기준에 따른 작업 선택을 배제함으로써, 좀더 유연하게 그리고 폭 넓게 문제를 고찰하는 장점이 있다. 또한 작업 선택에 대한 기준을 특정 작업이 포함되었을 때의 결과에 직접 의지함으로써, 문제의 범위를 좁혀가면서 가능한 해를 모두 비교하는 장점이 있다.

다음은 제안된 알고리듬을 단계별로 정리한 것이다.

[제안 알고리듬]

단계 1. K=1
각각 $n$개의 작업에 대하여 평균 flowtime을 계산하여 그 중 최소값을 갖는 하나의 작업을 선택하여 현재해 S에 기록. 선행작업이 없기 때문에 평균 flowtime은 각 기계에서의 작업 시간 합이다. 나머지 $(n-1)$ 작업은 잔류 집합 R에 저장.

단계 2. K=K+1

단계 3. 작업 j ∈ R를 현재해 S의 K개 slot에 삽입하여 K개의 후보해를 생성 후, 최소 평균 flowtime을 생성하는 해를 선택. 잔류 집합 R에서 작업 j 제거.

단계 4. 만약 잔류 집합 R ≠ $\phi$ 이면, 단계 3으로 이동.

단계 5. $(n-K+1)$의 후보해중 최소 평균 flowtime을 갖는 해를 선택하여 현재해 S에 기록. $(n-K)$개의 현재해를 제외한 작업은 잔류집합 R에 저장.

단계 6. 만약 K = $n$이면, 정지.
그렇지 않으면, 단계 2로 이동.

일반적으로 순열 flowshop 알고리듬의 복잡도는 작업수 $n$와 기계수 $m$에 따라 좌우된다. 전장에서 고찰한 CDS 알고리듬의 복잡도는 Johnson의 알고리듬을 이용하였으므로 기계수에 보다 더 좌우하며, 작업 삽입 방법에 기인한 NEH, Rajendran, 그리고 제안 알고리듬은 작업의 수에 보다 더 좌우한다. 따라서, 알고리듬의 복잡성을 비교하는 방법은 표 1과 같이 최종 스케줄을 얻기 위하여 알고리듬 실행 도중에 사용된 부분 스케줄의 수를 계산하는 것이 합리적일 것이다. 표 1과 같이 기계수에 따라 CDS 알고리듬은 비교될 스케줄의 수가 $(m-1)$로 결정되며, 작업수에 따라 NEH 알고리듬의 경우는 $O(n^2)$, Rajendran 알고리듬의 경우는 $O(N^2)$, 그리고 제안 알고리듬의 경우는 $O(N^3)$로 나타낼수 있다.

**표 1. 알고리듬간의 부분 스케줄 생성수 비교**

| 알고리듬 | 부분 스케줄 생성수 |
|---|---|
| CDS | $m-1$ |
| NEH | $n(n+1)/2-1$ |
| Rajendran | i) $n$이 짝수일 때: $(n^2+6n-8)/4$<br>ii) $n$이 홀수일 때:<br>$\{(n')^2+6n'-8\}/4-(n'/2+1)$<br>단, $n'$은 $n$보다 큰 짝수중 가장 작은 수 |
| 제안 알고리듬 | $n(n+1)(n+2)/6$ |

## 4. 예제

제안된 휴리스틱을 예를 들어 설명하기 위하여 Nawaz et al.[17]의 논문에서 사용된 4-작업 5-기계 문제를 평균 flowtime을 최소화할 수 있도록 풀어본다. 각각의 작업에 대한 작업 시간이 표 2에 나타나 있다.

**표 2. 4-작업 5-기계 작업시간**

| 작업 | 기계1 | 기계2 | 기계3 | 기계4 | 기계5 |
|---|---|---|---|---|---|
| 1 | 5 | 9 | 8 | 10 | 1 |
| 2 | 9 | 3 | 10 | 1 | 8 |
| 3 | 9 | 4 | 5 | 8 | 6 |
| 4 | 4 | 8 | 8 | 7 | 2 |

알고리듬의 단계 1은 초기화 작업이라고

할 수 있다. 전체 4개의 작업에 대하여 각각 평균 flowtime을 계산하였다. 선행작업이 없기 때문에 평균 flowtime $\overline{F}$은 전체 작업 시간과 같다. 각각의 평균 flowtime은 33, 31, 32, 29의 값을 갖으며, 작업 4가 최소값을 갖으므로 S={4}, 그리고 잔류 집합 R={1, 2, 3}이 된다. 단계 2에서는 K=2를 설정한다. 단계 3에서는 $j(=1) \in R$ 을 현재해 S={4}의 2개 slot에 삽입하여 2개의 후보해 {1,4}와 {4,1}을 생성한다. 최소값은 {4,1}에서 결정되었으며 ($\overline{F}$=34.5), 잔류 집합 R={2,3}가 되었다. 단계 4에서는 잔류 집합 $R \neq \phi$ 이므로 단계 3으로 이동한다. 다시 단계 3에서는 작업 2, 작업 3의 순서로 후보해를 평가하며 결과는 표 3과 같이 $\overline{F}${4,1}=34.5, $\overline{F}${4,2}=34, $\overline{F}${4,2}=35가 구해졌다. 단계 5에서는 전체 3(=4-2+1)의 후보해중 해 {4,2}이 $\overline{F}${4,2}=34로 최소값을 갖으므로 현재해 S로 선택되었다. 또한 잔류 집합 R은 현재해 S={4,2}을 제외한 작업 즉 R={1,3}가 되었다. 단계 6에서는 K(=2) $\neq n(=4)$이므로 단계 2로 이동한다. 단계 2에서는 K값을 3으로 증가시키며, 위의 과정이 차례로 K=$n$=4가 될 때까지 반복된다. 표 3에 알고리듬의 결과가 단계적으로 정리되었으며 최종적으로 $\overline{F}$=42.5를 갖는 최적해 S={4,3,1,2}를 생성하였다.

표 4는 현 예제에 대한 각 알고리듬의 결과를 평균 flowtime에 대하여 나타낸 것이다. 평균 flowtime에 대하여, 모든 경우의 수를 고려한 최적 스케줄은 {4,3,1,2}이며, 그때의 평균 flowtime은 42.5이다. 제안 알고리듬과 NEH 알고리듬이 최적해를 생성하였으며, CDS 알고리듬은 최적값으로부터 4.24%의 편차를 보였으며, Rajendran의 알고리듬은 1.18%의 편차를 보였다.

**표 3. 단계적 알고리듬의 진행 결과**

| 진행수 K | 현재해 S | 삽입작업 $j \in R$ | 평균 flowtime $\overline{F}$ |
|---|---|---|---|
| 1 | {1} | 1 | 33 |
| | {2} | 2 | 31 |
| | {3} | 3 | 32 |
| | {4} | 4 | 29* |
| 2 | {4,1} | 1 | 34.5 |
| | {4,2} | 2 | 34* |
| | {3,4} | 3 | 35 |
| 3 | {4,1,2} | 1 | 39* |
| | {4,2,3} | 3 | 39* |
| 4 | {4,3,1,2} | 3 | 42.5* |

**표 4. 평균 flowtime에 대한 각 알고리듬의 결과**

| 알고리듬 | 결과 스케줄 | 평균 flowtime (편차) |
|---|---|---|
| 최적해 | {4,3,1,2} | 42.5 (0%) |
| 제안 알고리듬 | {4,3,1,2} | 42.5 (0%) |
| CDS 알고리듬 | {2,3,4,1} | 44.3 (4.24%) |
| NEH 알고리듬 | {4,3,1,2} | 42.5 (0%) |
| Rajendran 알고리듬 | {4,2,3,1} | 43.0 (1.18%) |

## 5. 시뮬레이션을 이용한 실험

### 5-1. 실험계획

본 장에서는 시뮬레이션을 이용하여 제안된 알고리듬을 기존의 알고리듬과 비교하였다. 기존의 알고리듬과 제안된 알고리듬이 CRAY Y/MP 시스템에서 FORTRAN 90으로 코딩되었다. 실험은 두가지 단계로 구분된다. 첫번째 단계는 작업의 수가 적은 문제를 고려하는 것이다. 작업의 수가 적은 경우에는 제한된 시간에 전체 해 영역을 모두 검토할 수 있으므로 최적해를 구할 수 있다. 고려된

작업의 수는 5, 6, 7, 8, 9 이며 기계의 수는 5, 10, 15, 20이다. 두번째 단계는 작업의 수가 많은 문제를 고려하는 것이다. 고려된 작업의 수는 10, 20, 30, 40, 50, 60, 70, 80이며 기계의 수는 5, 10, 15, 20이다. 두 경우 모두에서, 고려된 작업과 기계에 대하여 각각 100회씩의 실험이 행해졌다. 따라서, 첫번째 단계에서는 총 문제 수가 10,000이고 두 번째 단계에서는 12,800이다. 작업 시간은 Nawaz et al.에서 사용된 uniform 분포 (1,99)를 가정하였다.

각 알고리듬의 성능을 평가하기 위하여, 단계 1에서는 최적의 값을 나타낸 빈도수와 최적값으로 부터의 평균 편차를 사용하였다. 최적값으로 부터의 평균 편차는 다음과 같이 정의된다:

$$\sum_{i=1}^{100}\left(\frac{\text{알고리듬결과}_i-\text{최적값}_i}{\text{최적값}_i}\right)/100$$

단계 2에서는 최적의 값을 구하기가 어렵기 때문에 알고리듬의 결과중에서 가장 우수한 결과를 기준으로 하였다. 따라서, 최우수 결과를 나타낸 빈도수와 최우수 결과로부터의 평균 편차를 성능 평가 기준으로 정하였다. 최우수 결과로부터의 평균 편차는 다음과 같이 정의된다:

$$\sum_{i=1}^{100}\left(\frac{\text{알고리듬결과}_i-\text{최우수결과}_i}{\text{최우수결과}_i}\right)/100$$

### 5-2. 실험결과

표 5는 작업의 수가 적은 문제를 다룬 단계 1의 시뮬레이션 결과이다. 제안된 알고리듬은 평균 편차가 0.844%이며 최소값은 0.322%, 최대값은 1.817%이다. CDS 알고리듬의 경우는 평균이 4.875%, 최소값은 1.997% 최대값은 10.181%이며, NEH 알고리듬은 평균이 1.059%, 최소값은 0.427% 최대값은 1.864%이다. 또한 Rajendran의 알고리듬은 평균이 1.171%, 최소값은 0.429% 최대값은 2.421%이다. 또한 최적값을 나타낸 빈도수는 평균으로 제안된 알고리듬이 37.35, CDS 알고리듬이 6.4, NEH 알고리듬이 31 그리고 Rajendran의 알고리듬이 33.55를 나타냈다.

표 6은 작업의 수가 많은 문제를 다룬 단계 2의 시뮬레이션 결과이다. 비교된 알고리듬의 상대적 평균 편차는 제안 알고리듬 0.320%, CDS 알고리듬 13.044%, NEH 알고리듬 1.646%, Rajendran 알고리듬 2.032%이며, 가장 좋은 해를 나타낸 빈도수는 제안된 알고리듬이 67.06, CDS 알고리듬이 0.03, NEH 알고리듬이 23.75 그리고 Rajendran의 알고리듬이 10.59를 나타냈다. 따라서 위의 두 성능 측정을 참조하면, 제안 알고리듬, NEH 알고리듬, Rajendran 알고리듬, 그리고 CDS 알고리듬의 순서로 알고리듬의 우수 정도를 정할 수 있다.

표 7은 작업수에 따른 경험적 알고리듬의 평균 CPU 소요시간을 나타낸다. CPU 소요시간이 기계수에 관계하기도 하지만, 그보다는 작업의 수에 더욱 민감하다. 평균적으로 단일 문제에 대하여 각 알고리듬에 소요된 CPU 시간은 제안 알고리듬 1.99795초, CDS 알고리듬 0.0135초, NEH 알고리듬 0.11303초, 그리고 Rajendran 알고리듬 0.06020초이다. 상대적인 관점에서 제안 알고리듬이 다른 알고리듬에 비해 많은 CPU 시간을 요구하지만, 기존의 최적화 기법과 비교하여 CPU 시간은 그렇게 부담이 되지 않는다. 참고로 109

**표 5. 최적해를 기준으로 한 경험적 알고리듬의 성능 평가 비교**

| 작업수 | 기계수 | 제안알고리듬 | | CDS | | NEH | | Rajendran | |
|---|---|---|---|---|---|---|---|---|---|
| | | 평균 편차(%) | 최적해 빈도수 | 평균 편차(%) | 최적해 빈도수 | 평균 편차(%) | 최적해 빈도수 | 평균 편차(%) | 최적해 빈도 |
| 5 | 5 | 0.572 | 59 | 5.910 | 6 | 0.694 | 52 | 0.862 | 56 |
| | 10 | 0.331 | 67 | 3.142 | 18 | 0.677 | 49 | 0.455 | 64 |
| | 15 | 0.402 | 63 | 2.663 | 20 | 0.539 | 63 | 0.558 | 61 |
| | 20 | 0.322 | 70 | 1.997 | 28 | 0.427 | 65 | 0.429 | 67 |
| 6 | 5 | 0.871 | 70 | 5.695 | 28 | 1.165 | 65 | 1.160 | 67 |
| | 10 | 0.569 | 50 | 4.191 | 7 | 0.761 | 45 | 0.862 | 43 |
| | 15 | 0.441 | 55 | 2.990 | 4 | 0.760 | 35 | 0.670 | 46 |
| | 20 | 0.554 | 42 | 2.581 | 8 | 0.599 | 40 | 0.720 | 40 |
| 7 | 5 | 1.023 | 34 | 7.731 | 0 | 1.359 | 23 | 1.385 | 30 |
| | 10 | 0.917 | 35 | 5.266 | 2 | 1.027 | 32 | 1.166 | 31 |
| | 15 | 0.841 | 30 | 3.658 | 2 | 1.145 | 23 | 1.146 | 23 |
| | 20 | 0.554 | 38 | 3.378 | 1 | 1.029 | 17 | 0.759 | 31 |
| 8 | 5 | 1.227 | 24 | 9.050 | 0 | 1.553 | 22 | 1.625 | 22 |
| | 10 | 1.113 | 17 | 6.210 | 0 | 1.402 | 16 | 1.563 | 15 |
| | 15 | 1.124 | 20 | 3.925 | 2 | 1.083 | 18 | 1.569 | 17 |
| | 20 | 0.898 | 23 | 3.853 | 2 | 0.919 | 18 | 1.190 | 19 |
| 9 | 5 | 1.817 | 11 | 10.181 | 0 | 1.854 | 11 | 2.421 | 13 |
| | 10 | 1.332 | 14 | 6.299 | 0 | 1.536 | 6 | 2.061 | 8 |
| | 15 | 1.016 | 11 | 4.964 | 0 | 1.381 | 13 | 1.532 | 6 |
| | 20 | 0.959 | 14 | 3.808 | 0 | 1.274 | 7 | 1.282 | 12 |

MIPS의 계산 능력을 갖는 SUN 670 시스템에서 제안 알고리듬의 실행 CPU 시간은 10에서 80까지의 작업수에 대하여 평균적으로 0.02, 0.32, 1.58, 4.63, 19.60, 38.32, 70.24, 147.33초를 나타냈다. 따라서 실제 생산 현장에서 워크 스테이션급의 UNIX 시스템이 많이 사용되므로, 제안 알고리듬의 실행 CPU 시간은 생산현장에서 충분히 수렴된다.

시뮬레이션 실험에서 특이한 점은 제안된 알고리듬이 문제의 크기가 커지면 커질수록 타 알고리듬에 비하여 우수한 성능을 나타낸다는 것이다. 특히, 작업수와 기계수의 비율이 크면 클수록 제안 알고리듬의 우수성이 뚜렷해진다. 이러한 관계가 그림 1과 그림 2에 나타나있다. 그림 1은 작업수와 기계수의 비를 축으로 최우수해 빈도수를 나타낸 것이고, 그림 2는 상대 평균 편차의 분포를 나타낸 것이다. 그림 1에서 보면 제안 알고리듬의 최우수해 정도가 타 알고리듬에 비하여 전반적으로 많지만, 특히 작업수와 기계수의 비가 2보다 큰 경우 뚜렷하게 그 차이를 알 수 있다. 또한 그림 2의 상대평균편차에 대해서도, 작업수와 기계수의 비가 2보다 큰 경우 제안 알고리듬이 타 알고리듬에 비하여

**표 6. 최우수해를 기준으로 한 경험적 알고리듬의 성능 평가 비교**

| 작업수 | 기계수 | 제안알고리듬 | | CDS | | NEH | | Rajendran | |
|---|---|---|---|---|---|---|---|---|---|
| | | 평균 편차(%) | 최우수해 빈도수 | 평균 편차(%) | 최우수해 빈도수 | 평균 편차(%) | 최우수해 빈도수 | 평균 편차(%) | 최우수해 빈도수 |
| 10 | 5 | 0.692 | 44 | 9.264 | 0 | 1.009 | 42 | 1.612 | 28 |
| | 10 | 0.671 | 46 | 5.749 | 0 | 0.926 | 36 | 1.310 | 26 |
| | 15 | 0.489 | 46 | 4.689 | 0 | 0.849 | 42 | 1.130 | 22 |
| | 20 | 0.578 | 45 | 3.936 | 1 | 0.807 | 44 | 0.993 | 23 |
| 20 | 5 | 0.488 | 60 | 5.080 | 0 | 1.849 | 17 | 1.946 | 23 |
| | 10 | 0.563 | 43 | 9.825 | 0 | 0.924 | 43 | 1.909 | 14 |
| | 15 | 0.770 | 41 | 7.526 | 0 | 0.730 | 42 | 1.861 | 17 |
| | 20 | 0.597 | 48 | 6.304 | 0 | 0.687 | 38 | 1.708 | 14 |
| 30 | 5 | 0.236 | 79 | 8.855 | 0 | 2.600 | 10 | 2.157 | 11 |
| | 10 | 0.389 | 62 | 1.839 | 0 | 1.101 | 30 | 2.252 | 8 |
| | 15 | 0.598 | 50 | 9.254 | 0 | 0.854 | 36 | 2.110 | 14 |
| | 20 | 0.593 | 45 | 8.155 | 0 | 0.645 | 40 | 1.709 | 15 |
| 40 | 5 | 0.144 | 82 | 0.608 | 0 | 3.382 | 5 | 1.802 | 13 |
| | 10 | 0.219 | 73 | 3.421 | 0 | 1.422 | 17 | 2.289 | 10 |
| | 15 | 0.317 | 60 | 0.674 | 0 | 0.837 | 38 | 2.331 | 2 |
| | 20 | 0.647 | 44 | 9.208 | 0 | 0.643 | 44 | 2.002 | 12 |
| 50 | 5 | 0.079 | 87 | 2.510 | 0 | 3.910 | 1 | 1.969 | 12 |
| | 10 | 0.138 | 83 | 4.544 | 0 | 1.594 | 15 | 2.461 | 2 |
| | 15 | 0.332 | 59 | 1.727 | 0 | 0.933 | 31 | 2.236 | 11 |
| | 20 | 0.260 | 62 | 9.915 | 0 | 0.749 | 32 | 2.062 | 6 |
| 60 | 5 | 0.058 | 90 | 3.332 | 0 | 4.388 | 0 | 1.773 | 10 |
| | 10 | 0.064 | 88 | 5.327 | 0 | 1.851 | 8 | 2.496 | 4 |
| | 15 | 0.272 | 69 | 2.264 | 0 | 1.019 | 26 | 2.467 | 5 |
| | 20 | 0.262 | 67 | 0.752 | 0 | 0.811 | 29 | 2.339 | 4 |
| 70 | 5 | 0.036 | 91 | 4.383 | 0 | 4.672 | 0 | 1.710 | 9 |
| | 10 | 0.037 | 92 | 6.427 | 0 | 2.145 | 5 | 2.529 | 3 |
| | 15 | 0.197 | 73 | 3.122 | 0 | 1.230 | 22 | 2.373 | 5 |
| | 20 | 0.191 | 70 | 1.212 | 0 | 0.823 | 26 | 2.462 | 4 |
| 80 | 5 | 0.009 | 96 | 5.529 | 0 | 5.075 | 0 | 1.748 | 4 |
| | 10 | 0.043 | 94 | 6.926 | 0 | 2.286 | 3 | 2.431 | 3 |
| | 15 | 0.047 | 90 | 3.383 | 0 | 1.210 | 8 | 2.545 | 2 |
| | 20 | 0.217 | 67 | 1.666 | 0 | 0.709 | 30 | 2.300 | 3 |

명백하게 우월하다. 실제 현장의 사례를 연구한 Bestwick & Hastings[5]와 Lahiri[14]에 따르면, 작업수와 기계수의 비는 각각 5.625배 (작업수 45, 기계수 8)와 7.333배 (작업수 44, 기계수 6) 정도이다. 따라서 제안된 알고리듬은 생산 현장의 실제적 스케줄링 해결에

**표 7. 알고리듬의 CPU 소요 시간(초) 비교**

| 작업수 | 제안 알고리듬 | CDS 알고리듬 | NEH 알고리듬 | Rajendran 알고리듬 |
|---|---|---|---|---|
| 10 | 0.00337 | 0.00085 | 0.00102 | 0.00072 |
| 20 | 0.03923 | 0.00258 | 0.00671 | 0.00405 |
| 30 | 0.17444 | 0.00525 | 0.02093 | 0.01190 |
| 40 | 0.51612 | 0.00886 | 0.04754 | 0.02621 |
| 50 | 1.19126 | 0.01343 | 0.08922 | 0.04828 |
| 60 | 2.40207 | 0.01895 | 0.15138 | 0.08086 |
| 70 | 4.35293 | 0.02535 | 0.23703 | 0.12548 |
| 80 | 7.30420 | 0.03272 | 0.35036 | 0.18406 |
| 평균 | 1.99795 | 0.01350 | 0.11303 | 0.06020 |

유용한 대안을 제공할 수 있을 것이다.

그림 1과 그림 2에서 특기할 만한 사항은 Rajendran의 알고리듬에 관한 것이다. 표 5와 표 6에서는 평균적인 의미에서 NEH 알고리듬이 Rajendran 알고리듬보다 우월하였지만, 작업수와 기계수의 관계를 고려하면 그림 1과 그림 2에서와 같이 Rajendran 알고리듬의 결과가 NEH 알고리듬의 결과보다 우월하다. 즉, 작업수와 기계수의 비가 6보다 큰 경우, Rajendran 알고리듬이 최우수해 빈도수와 상대평균편차 측정에서 NEH 알고리듬보다 우월하였다.

## 6. 결론

본 연구에서는 $n$-작업 $m$-기계 순열 flowshop 문제에서 평균 flowtime을 최소화하는 작업 삽입 방법에 기반한 경험적 알고리듬을 제안하였다. 제안된 알고리듬은 시뮬레이션을 이용하여 실험되었으며, 실험은 크게 두 가지 단계 즉 최적해를 구할 수 있는 적은 작업수의 문제와 최적해를 구하기 어려운 많은 작업수의 문제로 나뉘어 광범위하게 수행되었다. 또한, 기존의 연구에서 우수한 결

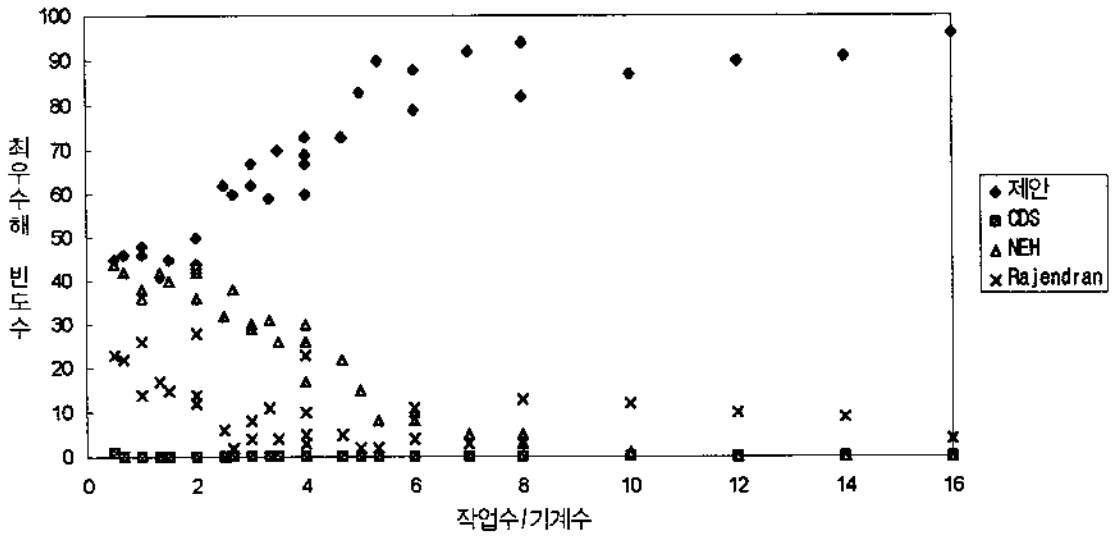


그림 1. 알고리듬간의 최우수해 빈도수 비교

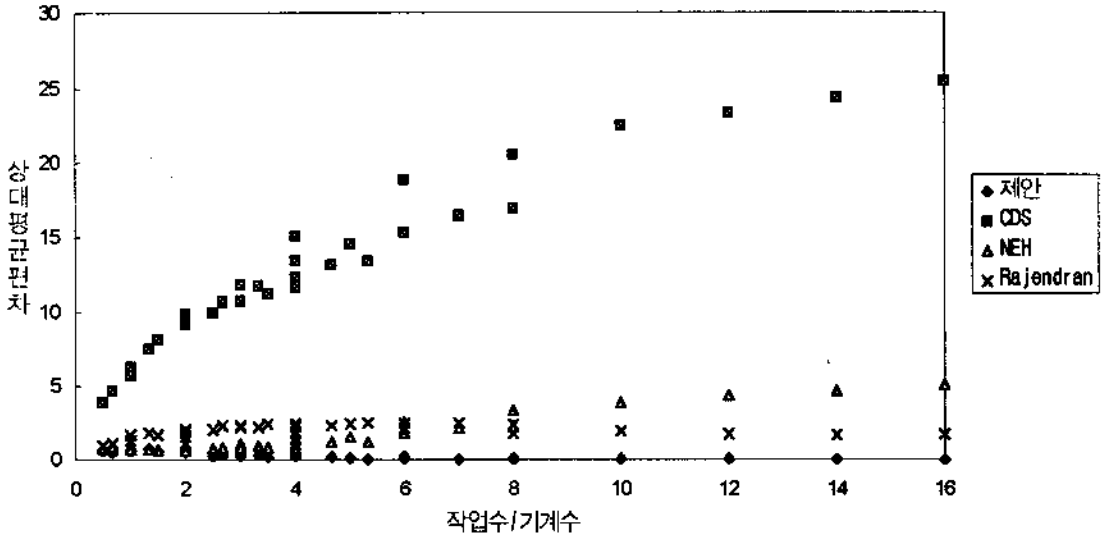


그림 2. 알고리듬간의 상대평균편차 비교

과를 생성한 CDS 알고리듬, NEH 알고리듬, 그리고 Rajendran의 알고리듬과 비교 분석되었다. 시뮬레이션 결과는 제안 알고리듬의 기존 알고리듬에 대한 우월성을 보여 주었으며, 특히 작업수와 기계수의 비율이 2보다 큰 경우 제안 알고리듬의 성능이 우수하였다.

## References

[1] Adiri, I. and Pohoryles, D., "Flowshop/no-idle or no wait scheduling to minimize the sum of completion time," Naval Research Logistics Quarterly, Vol. 29, pp. 395-504, 1982.

[2] Ahmadi, R. and Bagchi, U., "Improved lower bounds for minimizing the sum of flow times of n jobs over m machines in a flow shop", European Journal of Operational Research, Vol. 44, pp. 331-336, 1990.

[3] Baker, K. R., Introduction to sequencing and scheduling, Wiley, New York, 1974.

[4] Bansal, S., "Minimizing the sum of completion times of n-jobs over m machines in a flow shop," AIIE Transactions, Vol. 9, pp. 306-311, 1977.

[5] Bestwick, P. and Hastings, N., "A new bound for machine scheduling," Operations Research Quarterly, Vol. 27, pp. 479-487, 1976.

[6] Campbell, H., Dudek, R. and Smith, M., "A heuristic algorithm for the n-job, M-machine sequencing problem," Management Science, Vol. B 16, pp. 630-637, 1970.

[7] Dannenbring D., "An evaluation of flow-

shop sequencing heuristics," Management Science, Vol. 23, pp. 1174-1182, 1977.

[8] Garey, M., Johnson, D. and Sethis, R., "The complexity of flowshop and jobshop scheduling," Mathematics of Operations Research, Vol. 1, pp. 117-129, 1976.

[9] Gupta, J., "Heuristic algorithms for multistage flowshop scheduling problem," AIIE Transactions, Vol. 4, pp. 11-18, 1972.

[10] Ho, J. and Chang, Y., "A new heuristic for the n-job, m-machine flowshop problem," European Journal of Operational Research, Vol. 52, pp. 194-202, 1991.

[11] Ignall, E. and Schrage, L., "Application of the branch and bound technique to some flow shop scheduling problems," Operations Research, Vol. 13, pp. 400-412, 1965.

[12] Johnson, S., "Optimal two- and three-stage production schedules with set up times included," Naval Research Logistics Quarterly, Vol. 1, pp. 61-68, 1954.

[13] Karg, R. and Thompson, G., "A heuristic approach to solving travelling salesman problems," Management Science, Vol. 10, pp. 225-248, 1964.

[14] Lahiri, S., Rajendran, C. and Narendran, T., "Evaluation of heuristics for scheduling in a flowshop: a case study", Production Planning & Control, Vol. 4, pp. 153-158, 1993.

[15] Maccarthy, B. and Liu, J., "Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling," International Journal of Production Research, Vol. 31, pp. 59-79, 1993.

[16] Miyazaki, S., Nishiyama, N., and Hashimoto, F., "An adjacent pairwise approach to the mean flowtime scheduling problem," Journal of the Operational Research Society of Japan, Vol. 21, pp. 287-299, 1978.

[17] Nawaz, M. Enscore, E. and Ham, I., "A heuristic algorithm for the m-machine, n-job flowshop sequencing problem," OMEGA, Vol. 11, pp. 91-95, 1983.

[18] Pinedo, M., Scheduling: Theory, algorithms, and systems, Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[19] Rajendran, C., "Heuristic algorithm for scheduling in a flowshop to minimize total flowtime," International Journal of Production Economics, Vol. 29, pp. 65-73, 1993.

[20] Rajendran, C. and Chaudhuri, D., "An efficient heuristic approach to the scheduling of jobs in a flowshop," European Journal of Operational Research, Vol. 61, pp. 318-325, 1991.

[21] Rinnooy Kan, A., Machine scheduling problems, Nijhoff, The Hague, 1976.

[22] Szwarc, W., "The flowshop problem with mean completion time criterion," IIE Transactions, Vol. 15, pp. 172-176, 1983.

[23] Taillard, E., "Some efficient heuristic methods for the flow shop sequencing problem," European Journal of Operational Research, Vol. 47, pp. 65-74, 1990.

[24] Turner, S. and Booth, D., "Comparison of heuristics for flow shop sequencing," OMEGA, Vol. 15, pp. 75-78, 1987.

95년 11월 최초 접수, 96년 2월 최종 수정