검사공정의 작업배분을 위한 휴리스틱 알고리즘 개발

이 석 환* · 박 승 헌*
*인하대학교 산업공학과

Heuristic algorithm to assign job in inspection process

Seog-Hwan Lee* · Seung-Hun Park*
*Dept. of Industrial Engineering, Inha University

Abstract

In this paper, we developed a heuristic algorithm to assign job to workers in parallel line inspection process without sequence. Objective of assigning job in inspection process is only to assign job to workers evenly. But this objective needs much time and effort since there are many cases in assigning job and cases increase geometrically if the number of job and worker increases. In order to solve this problem, we proposed heuristic algorithm to assign job to workers evenly. Experiments of assigning job are performed to evaluate performance of this heuristic algorithm. The result shows that heuristic algorithm can find the optimal solution to assign job to workers evenly in many type of cases. Especially, in case there are more than two optimal solutions, this heuristic algorithm can find the optimal solution with 98% accuracy.

Keywords : Parallel line, Heuristic algorithm, Smooth index

1. 서 론

병렬라인 시스템은 대량생산이 가능한 이점이 있다. 이러한 시스템에서는 병렬라인의 검사공정을 하나로 통합하여 그 효율성을 꾀하는 특징이 있다. 이러한 방식은 주로 전기와 전자분야에 도입되고 있으며, PC 및 HDD 생산 등에 사용된다. 즉 이 시스템은 병렬로 늘어선 생산 라인(공정)에서, 효율성을 위해 최종 검사공정은 하나로 통합된 형태이다. 이 최종 검사 공정에는 복수의 작업자가 있으며, 각 작업자가 부품의 종류를 가리지 않고 모든 종류의 검사를 수행한다(<그림 1> 참조). 이때 각 작업자들에게 어떻게 부품의 검사 작업들을 배분 할 것인가 하는 문제가 발생된다. 작업들이 순서 의존적인 경우에는 라인효율이 최대로 되게 각 작업자에게 작업을 배분하는 전형적인 라인밸런싱 문제가 된다. 라인밸런싱에 관한 연구는 과거부터 수많은 연구가 있었다[2], [9], [11], [12]. 그러나 위에서 언급한 검사공정과 같이 작업들 간에 순서가 없는 특별한 경우는 그다지 다루어지지 않은 것 같다.

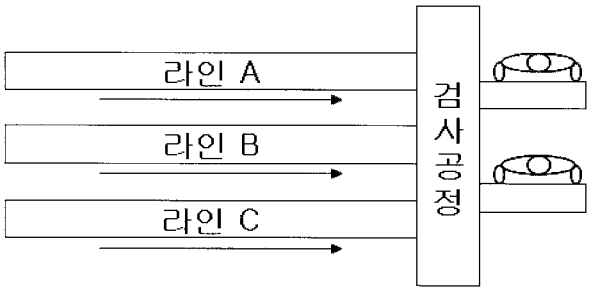
본 연구는 병렬라인 시스템의 작업순서가 없는 검사공정을 대상으로 복수의 작업자에게 작업(시간)을 배분하는 휴리스틱 알고리즘을 개발한다. 작업들이 순서의 존적인 경우, 목적함수는 각 작업자의 유휴시간의 합이 최소가(즉, 라인효율이 최대)되도록 한다. 그러나 작업간에 순서가 없는 경우에는 먼저 작업이 끝나는 작업자가 다른 작업자를 기다릴 필요가 없기 때문에 유휴시간의 합은 의미가 없다. 이러한 상황에서는 오히려 각 작업자가 얼마나 공평하게 작업시간을 배분 받는가가 목적함수로서 더욱 중요한 의미를 갖는다. 따라서 본 연구는 각 작업자간의 작업시간 합의 편차가 최소가 되도록 각 작업자에게 작업들을 배분하는 문제를 다룬다.

† 이 논문은 인하대학교 교내 연구비 지원에 의해 연구되었음.
† 교신저자: 이석환, 경기도 부천시 소사구 괴안동 185-34, 유한대학 금형설계과
M · P: 019-208-9250, E-mail: seoghwan@inha.ac.kr
2008년 7월 접수; 2008년 8월 수정본 접수; 2008년 8월 게재확정

2. 문제의 정의 및 휴리스틱 알고리즘

2.1 문제의 정의

본 연구에서는 병렬라인 시스템에서 작업순서가 없는 검사공정을 대상으로 복수의 검사 작업자에게 작업을 공평하게 배분하는 문제를 다룬다. 검사공정은 여러 라인에서 생산한 부품을 검사하는 공정으로 일반적인 형태는 <그림 1>과 같다. 라인 A, B, C는 각각 여러 가지 부품을 생산하여 검사공정으로 보내며, 하루 동안 검사공정으로 도착하는 부품의 종류와 수량은 정해져 있다. 검사공정으로 도착한 부품은 여러 명의 작업자에 의해 성능검사를 거친다. 각 작업자는 검사시간이 다른 여러 가지의 부품을 검사하기 때문에 각 작업자의 작업시간 합은 다르다. 여기에서 각 작업자의 작업시간 합의 차가 커지면 공평한 작업배분이 되지 않는다. 만일 어떤 작업자가 공평하지 않은 작업배분으로 인하여 다른 작업자보다 더 많은 시간의 작업을 하게 된다면 그 작업자는 육체적 피로의 증가와 정신적 스트레스로 인하여 검사작업의 질을 떨어뜨릴 수 있으며 검사 시스템의 효율도 낮아진다. 따라서 공평한 작업배분은 검사작업의 질을 향상시키고 검사 시스템의 효율을 높이기 위해서 중요하다.



<그림 1> 검사공정

검사공정의 작업이 순서의존적일 경우, 작업배분은 라인밸런싱 문제가 되며 목적함수는 라인 효율이 된다.

그러나 본 연구는 작업순서가 없는 문제로서 위에서 언급한 대로 목적함수는 작업자 간의 공평한 작업배분이 된다. 따라서 본 연구에서는 라인 효율을 목적함수로써 고려하지 않는다. 작업을 공평하게 배분한다는 것은 작업자 간 작업시간 합의 편차를 최소로 하는 것이다. 작업시간 합 편차의 최소는 평활지수(smooth index)로 측정이 가능하다. 평활지수는 수치들의 편차가 높고 낮음을 나타내는 척도로써 수치들의 편차가 적어지면 평활지수는 낮아진다. 따라서 각 작업자의 작업시간 합에 대해 평활지수가 낮을수록 작업은 공평하게 배분되었다고 볼 수 있다. 평활지수가 0인 경우는 최적배분의 해가 된다.

다음 식 (1)은 평활지수를 나타낸다.

$$\sqrt{\sum_{i=1}^{n}(Max[T_i]-T_i)^2}$$

작업자 : $i=1,2,\ldots,n$
n: 작업자의 수
T_i: 각 작업자의 작업시간 합 ····· (1)

한편 검사작업의 종류와 작업자의 수가 늘어나면 작업자에게 검사작업을 배분하기 위한 경우의 수는 다음 식 (2)와 같이 기하급수적으로 늘어난다.

$$\begin{aligned} &{}_kC_n + {}_{(k-n)}C_1 + {}_{(k-n-1)}C_1 + \ldots + {}_1C_1 \\ &\qquad\qquad \cdots \\ &{}_kC_n + {}_{(k-n)}C_1 + {}_{(k-n)} \end{aligned}$$

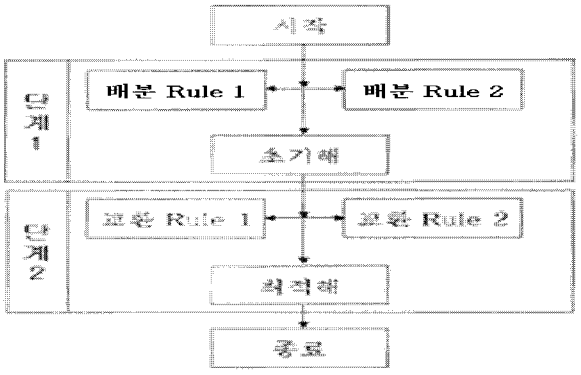
k 작업의 수 ····· (2)
n 작업자의 수

따라서 최적배분의 해를 발견하기 위해서는 많은 양의 계산이 필요하므로 휴리스틱 해법의 필요성이 있다.

본 연구에서는 작업자에게 작업의 양을 공평하게 배분하기 위한 휴리스틱 알고리즘을 제안하며, 그 때의 목적함수는 평활지수 최소이다.

2.2 휴리스틱 알고리즘

여기에서는 위에서 언급한 휴리스틱 알고리즘의 전개과정을 설명한다. 작업배분과정은 다음과 같다. <그림 2>와 같이 1단계에서는 작업을 배분하며 2단계에서는 배분된 작업을 교환한다. 다음으로 알고리즘에서 작업 간 결합의 기준으로 사용하는 이론적 배분시간 T^*는 식 (3)과 같다.



<그림 2> 작업배분과정

이론적 배분시간 $T^* = \frac{\sum_{i=1}^{k} J_i}{N}$ ·····(3)

총 작업시간 $\sum_{i=1}^{k} J_i$

요소작업시간 $J_i,\ i=1,2,\ldots,k$

작업자 $N_i,\ i=1,2,\ldots,n$

1단계는 작업배분의 초기해를 구하는 과정으로 최적 배분의 해를 구하는 시간을 단축하는 것이 목적이다. 1단계에서 사용한 방법은 가장 큰 작업과 가장 작은 작업을 결합하는 것이다. 이 방법은 한 작업자에게 배분한 총 작업시간을 이론적 배분시간에 근접하게 할 수 있다. 1단계는 배분 Rule 1과 배분 Rule 2로 구성한다.

1단계의 적용 과정은 다음과 같다. 배분 Rule 1, 2를 적용하기 전 먼저 한 명의 작업자에게 가장 작업시간이 긴 작업 $\max(J_i)$을 배분한다. 다음으로 배분 Rule 1과 배분 Rule 2를 적용한다. 배분 Rule 1과 배분 Rule 2에서 처리하는 내용은 <그림 3>과 같다.

배분 Rule 1 :	$k-1$개의 작업 중 작은 순서대로 하나씩 넣어 T^*를 넘지 않으면 가장 작은 작업을 배분한다. (단, T^*와 동일한 작업이 있으면 그 작업을 배분한다)
배분 Rule 2 :	$k-1$개의 작업 중 작은 순서대로 하나씩 넣어 T^*를 넘는 작업이 있으면, 첫 번째로 T^*를 넘는 작업의 초과시간과 T^*를 넘기 직전 작업의 미달시간의 차이를 계산하여 차이가 적은 작업을 배분한다.

<그림 3> 작업배분 1단계

2단계는 1단계에서 구한 초기해로 최적해를 구하는 과정이다. 2단계에서는 최적해를 발견하기 위해 작업교환을 사용한다. 작업교환은 교환 Rule 1과 교환 Rule 2로 구성한다. 교환 Rule 1은 두 작업자의 작업을 교환하는 것으로 각 작업자의 총 작업시간을 균등하게 하는 것을 목적으로 한다. 2단계의 적용 과정은 다음과 같다. 교환 Rule 1, 2를 적용하기 전 먼저 작업시간의 합이 가장 많은 작업자(T_{max})와 가장 적은 작업자(T_{min})를 선택하여 쌍 비교를 한다. 다음으로 교환 Rule 1과 교환 Rule 2를 적용한다. 교환 Rule 1과 교환 Rule 2에서 처리하는 내용은 <그림 4>과 같다.

교환 Rule 1 :	T_{min}의 작업 중에서 가장 작은 작업을 선택한다. 그 작업보다 T_{max}중 더 큰 작업이 있으면, 그 중 가장 작은 작업과 위에서 선택한 작업을 교환하고 더 큰 작업이 없으면 rule 2를 적용한다.
교환 Rule 2 :	T_{max} 중 가장 작은 작업을, T_{min}에 보낸다.

<그림 4> 작업교환 2단계

지금까지의 알고리즘을 순서대로 정리하면 다음과 같다.

1) 단계 1 (작업배분)

(1) 총 작업시간을 작업자의 수로 나누어 일인 작업자의 이론적 평균시간 T^*를 구한다.
(2) 작업을 작업시간의 내림차순으로 정렬한다.
(3) 한 명의 작업자에게 가장 작업시간이 긴 작업 $\max(J_i)$을 배분한다.
(4) $k-1$개의 작업 중 작은 순서대로 한 번씩 $\max(J_i)$에 붙여서 총 작업시간이 T^*를 넘는 작업이 있으면 (5)로 이동하고, 그렇지 않으면 가장 작은 작업을 배분하고 나머지 $k-2$개로 (4)를 재적용한다. (단, T^*와 동일한 작업이 있으면 그 작업을 배분한다)
(5) T^*를 넘는 작업이 있으면, 첫 번째로 T^*를 넘는 작업의 초과시간과 T^*를 넘기 직전 작업의 미달시간의 차이를 계산하여 차이가 적은 작업을 배분하고 작업자 한 명에 대한 작업배분을 종료한다. 다음 작업자로의 배분은 (3)으로 이동하여 같은 순서를 밟는다.

2) 단계 2 (배분된 작업교환)

(1) 단계 1의 배분결과로 평활지수를 구하고, 작업자별로 오름차순(작업시간이 짧은 순)으로 정렬한다.
(2) 작업시간의 합이 가장 많은 작업자(T_{max})와 가장 적은 작업자(T_{min})를 선택하여 쌍 비교를 한다.
(3) T_{min}의 작업 중에서 가장 작은 작업을 선택한다. 그 작업보다 T_{max}중 더 큰 작업이 없으면 (4)로 이동하고, 있으면 그 중 가장 작은 작업과 교환하였을 때, 두 작업자 간 작업시간의 차이를 계산한다.
(4) 원래 T_{max} 중 가장 작은 작업을, T_{min}에 보냈을 때의 두 작업자 간 작업시간의 차이를 계산한다.

(5) (3)과 (4)에서 구한 두 작업자 간 작업시간의 차이를 비교하여 차이가 적게 나온 경우를 작업배분으로 적용한다. 단, (3)에서 T_{max}중 더 큰 작업이 없다면 (4)를 작업배분으로 적용한다.

(6) 평활지수를 구하여 (1)의 평활지수보다 개선되었으면 단계 (2)에서 단계 (5)를 밟는다. 개선되지 않으면 배분을 종료한다.

<그림 5>는 알고리즘을 구체적으로 나타낸 것이다.

```
작업배분 1단계

for (i=1; i <= jobNum; i++)
{
        for (j=jobNum; j > i; j--)
        {
                workerSum = workerTime[i] +
                workerTime[j]
                if all ( T*<= workerSum)
                        //작업배분 1단계 Rule 1
                {
                      작업자에게 맨 끝작업 배분
                }
                else      //작업배분 1단계 Rule 2
                {
                if abs( T*- workerTime[i]
                          + workerTime[j])
                 < abs( T*- workerTime[i]
                          + workerTime[j+1])
                   j 작업 배분
                else
                   j+1 작업 배분
                }
        }
}
```

```
작업배분 2단계 Rule 1

Tmax 오름차순정렬
for (i=1; i <= TmaxJobNum; i++)
if Tmin의 작은 작업 < Tmax(i) then 작업교환
        if (Rule 1의 Tmax - Tmin)
         < (Rule 2의 Tmax - Tmin)
        Rule 1 작업배분 적용
else
        Rule 2 작업배분 적용
```

```
작업배분 2단계 Rule 2

Tmax 오름차순정렬
Tmax = Tmax - Tmax(1)
Tmin = Tmin + Tmax(1)
```

```
메인 프로그램

작업을 내림차순으로 정렬
작업배분 1단계 실행
do
{
        작업배분 2단계 Rule 2 실행
        작업배분 2단계 Rule 1 실행
} while (이전 평활지수 > 이후 평활지수)
```

<그림 5> 휴리스틱 알고리즘 상세

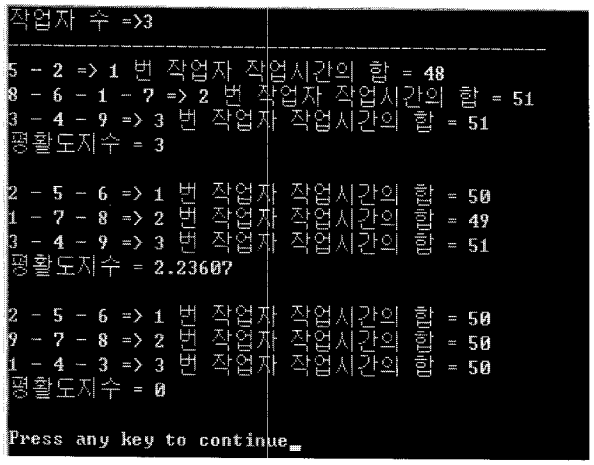
3. 적용결과 및 분석

위에서 제안한 휴리스틱 알고리즘은 C++언어로 구현하였다. 휴리스틱 알고리즘이 최적배분의 해를 어느 정도 찾을 수 있는지 평가하기 위해 두 가지 실험을 수행하였다. 첫 번째 실험은 평활지수 0인 작업배분이 존재하는 해 즉, 최적배분의 해가 있는 경우 휴리스틱 알고리즘이 최적배분의 해를 어느 정도 찾을 수 있는지 평가하는 것이다. 여기서 평활지수 0인 작업배분이란 각 작업자의 작업시간 합이 모두 동일한 경우를 말한다. 평활지수 0인 최적배분의 경우는 해가 한 개인 경우와 여러 개인 경우로 구분하였다. 최적배분의 해가 한 개 인 경우는 3.1절, 여러 개인 경우는 3.2절에서 수행하였다. 두 번째 실험은 평활지수 0인 작업배분이 존재하지 않는 해 즉, 최적배분의 해가 없는 경우 휴리스틱 알고리즘이 평활지수 0에 근사한 작업배분을 찾을 수 있는지 평가하는 것이다. 이 경우는 3.3절에서 수행하였다. 마지막으로 3.4절에서는 작업시간의 분산이 평활지수에 주는 영향을 평가하였다.

3.1 최적배분의 해가 한 개인 경우

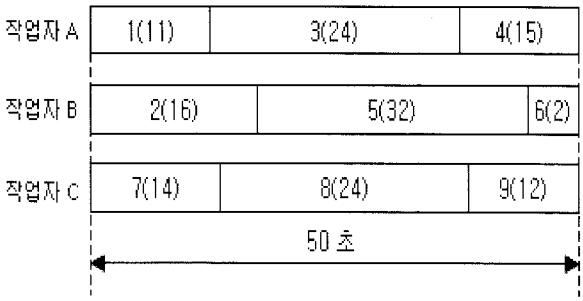
첫 번째 실험은 평활지수 0인 최적배분의 해가 한 개인 경우로 9, 12, 15개의 작업을 3, 4, 6명의 작업자에게 배분하는 예이다. 먼저 최적배분의 해가 한 개인 예를 만들고 그 예에 대해서 휴리스틱 알고리즘으로 평활지수 0인 작업배분을 어느 정도 찾을 수 있는지 확인한다. <그림 6>은 9개의 작업을 3명의 작업자에게 배분하기 위해 휴리스틱 알고리즘을 수행한 결과이다.

휴리스틱 알고리즘에 입력하는 자료는 작업번호와 그에 해당하는 작업시간 그리고 작업자의 수이다. 작업배분의 결과에는 각 작업자가 수행해야 할 작업번호와 작업시간의 합이 출력된다. 예컨대 "2-5-6 => 1번 작업자 작업시간의 합 = 50"이란 1번 작업자에게 2, 5, 6번 작업을 배분하고 그 때 작업시간의 합은 50이라는 의미이다. <그림 6>의 작업배분 결과는 <표 1>과 같이 나타낼 수 있다.



<그림 6> 휴리스틱 알고리즘 수행결과

<표 1>은 평활지수 0인 최적배분의 해가 한 개인 경우, 평활지수 0인 최적배분과 휴리스틱 알고리즘으로 발견한 작업배분의 해를 비교한 표이다. <표 1>에서 평활지수 0인 최적배분이란 유일하게 한 개가 존재하는 최적배분이고, 휴리스틱 알고리즘 수행결과란 제안한 휴리스틱 알고리즘으로 최적배분의 해를 찾은 것이다. <표 1>에서 작업자 A, B, C의 의미는 작업자가 세 명인 경우를 말하고, "작업자 A : 1, 3, 4 (작업시간 : 50)"은 A 작업자에게 1, 3, 4번 작업을 배분하며 작업시간 합은 50초라는 것을 의미한다. <표 1>의 휴리스틱 알고리즘의 수행결과가 "동일"의 의미는 휴리스틱 알고리즘이 평활지수 0인 최적배분의 해를 찾은 경우이다. <그림 7>은 <표 1>의 최적 배분의 해를 나타낸다. <그림 7>에서 괄호 앞의 숫자는 작업번호, 괄호의 숫자는 작업시간이다.



<그림 7> 휴리스틱 알고리즘의 작업배분 결과

다음은 작업자의 수를 4, 6명으로 늘려서 실험하였으며 그 결과는 <표 2>, <표 3>과 같다. <표 2>, <표 3>의 형식은 <표 1>과 동일하므로 설명은 생략한다.

<표 2>는 9개의 작업을 4명의 작업자에게 공평하게 배분하는 예로써 각 작업자의 작업시간 합은 30초이다.

<표 1> 9개 작업을 3명에게 배분

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	합
작업시간	11	16	23	16	32	2	14	24	12	150
작업자3명	평활지수 0인 최적배분					휴리스틱 알고리즘 수행결과				
	작업자 A : 1, 3, 4 (작업시간 : 50) 작업자 B : 2, 5, 6 (작업시간 : 50) 작업자 C : 7, 8, 9 (작업시간 : 50)					동일				

<표 2> 9개 작업을 4명에게 배분

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	합
작업시간	5	10	25	12	20	3	7	18	20	120
작업자4명	평활지수 0인 최적배분					휴리스틱 알고리즘 수행결과				
	작업자 A : 1, 3 (작업시간 : 30) 작업자 B : 2, 5 (작업시간 : 30) 작업자 C : 4, 8 (작업시간 : 30) 작업자 D : 6, 7, 9 (작업시간 : 30)					동일				

<표 3> 9개 작업을 6명에게 배분

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	합
작업시간	30	11	21	9	30	30	19	7	23	180

	평활지수 0인 최적배분	휴리스틱 알고리즘 수행결과
작업자6명	작업자 A : 1 (작업시간 : 30) 작업자 B : 3, 4 (작업시간 : 30) 작업자 C : 5 (작업시간 : 30) 작업자 D : 2, 7 (작업시간 : 30) 작업자 E : 6 (작업시간 : 30) 작업자 F : 8, 9 (작업시간 : 30)	동일

<표 3>은 9개의 작업을 6명의 작업자에게 공평하게 배분하는 예로써 작업시간의 합은 30초이다. <표 3>의 경우는 작업자 수가 많기 때문에 1, 5, 6번 작업자에게 30초의 작업을 한 개씩 할당해야만 최적배분이 된다.

<표 2>, <표 3>과 같이 휴리스틱 알고리즘은 모든 경우에 대해 평활지수 0인 최적배분의 해를 찾았다. 이번에는 작업의 개수를 12개로 증가시켜서 실험을 하였다. <표 4>~<표 6>은 12개의 작업을 3, 4, 6명의 작업자에게 배분한 결과이다.

<표 4>와 같이 작업자가 3명인 경우에는 최적배분의 해를 찾지 못하였다. 그러나 <표 5>, <표 6>과 같이 작업자가 4, 6명의 경우, 휴리스틱 알고리즘은 평활지수 0인 최적배분의 해를 찾았다. 추가로 15개의 작업을 3, 4, 6명의 작업자에게 배분하는 실험을 하였으며 그 결과는 <표 7>~<표 9>와 같다. <표 10>은 앞의 실험결과를 종합한 것이다. 실험은 결과의 정확성을 위해 작업시간을 다르게 하여 두 번씩 수행하였다.

<표 4> 12개 작업을 3명에게 배분

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	10	11	12	합
작업시간	12	8	7	13	6	11	9	14	4	14	10	12	120

	평활지수 0인 최적배분	휴리스틱 알고리즘 수행결과
작업자3명	작업자 A : 1, 2, 3, 4 (작업시간 : 40) 작업자 B : 5, 6, 7, 8 (작업시간 : 40) 작업자 C : 9, 10, 11, 12 (작업시간 : 40)	작업자 A : 8, 9, 5, 3, 7 (작업시간 : 40) 작업자 B : 2, 11, 6, 1 (작업시간 : 41) 작업자 C : 10, 12, 4 (작업시간 : 39) 평활지수 : 2.24

<표 5> 12개 작업을 4명에게 배분

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	10	11	12	합
작업시간	7	15	18	9	17	14	11	19	10	13	21	6	160

	평활지수 0인 최적배분	휴리스틱 알고리즘 수행결과
작업자4명	작업자 A : 1, 2, 3 (작업시간 : 40) 작업자 B : 4, 5, 6 (작업시간 : 40) 작업자 C : 7, 8, 9 (작업시간 : 40) 작업자 D : 10, 11, 12 (작업시간 : 40)	동일

<표 6> 12개 작업을 6명에게 배분

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	10	11	12	합
작업시간	32	25	15	19	21	8	16	29	24	11	9	31	240

	평활지수 0인 최적배분	휴리스틱 알고리즘 수행결과
작업자6명	작업자 A : 4, 5 (작업시간 : 40) 작업자 B : 1, 6 (작업시간 : 40) 작업자 C : 2, 3 (작업시간 : 40) 작업자 D : 7, 9 (작업시간 : 40) 작업자 E : 8, 10 (작업시간 : 40) 작업자 F : 11, 12 (작업시간 : 40)	동일

<표 7> 15개 작업을 3명에게 배분

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	합
작업시간	4	9	15	21	11	6	11	17	23	3	8	13	14	20	5	180

	평활지수 0인 최적배분	휴리스틱 알고리즘 수행결과
작업자3명	작업자 A : 1, 2, 3, 4, 5 (작업시간 : 60) 작업자 B : 6, 7, 8, 9, 10 (작업시간 : 60) 작업자 C : 11, 12, 13, 14, 15 (작업시간 : 60)	작업자 A : 9, 1, 10, 6, 15, 14 (작업시간 : 61) 작업자 B : 4, 11, 2, 7, 5 (작업시간 : 60) 작업자 C : 8, 3, 13, 12 (작업시간 : 59) 평활지수 : 2.24

<표 8> 15개 작업을 4명에게 배분

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	합
작업시간	5	18	11	16	11	9	13	17	16	8	17	9	30	6	14	200

	평활지수 0인 최적배분	휴리스틱 알고리즘 수행결과
작업자4명	작업자 A : 1, 2, 3, 4 (작업시간 : 50) 작업자 B : 5, 6, 7, 8 (작업시간 : 50) 작업자 C : 9, 10, 11, 12 (작업시간 : 50) 작업자 D : 13, 14, 15 (작업시간 : 50)	작업자 A : 13, 1, 9 (작업시간 : 51) 작업자 B : 2, 14, 10, 12, 6 (작업시간 : 50) 작업자 C : 5, 3, 7, 15 (작업시간 : 49) 작업자 D : 11, 4, 8 (작업시간 : 50) 평활지수 : 2.45

<표 9> 15개 작업을 6명에게 배분

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	합
작업시간	7	5	18	8	13	9	9	5	16	9	21	11	19	13	17	180

	평활지수 0인 최적배분	휴리스틱 알고리즘 수행결과
작업자6명	작업자 A : 1, 2, 3 (작업시간 : 30) 작업자 B : 4, 5, 6 (작업시간 : 30) 작업자 C : 7, 8, 9 (작업시간 : 30) 작업자 D : 10, 11 (작업시간 : 30) 작업자 E : 12, 13 (작업시간 : 30) 작업자 F : 14, 15 (작업시간 : 30)	동일

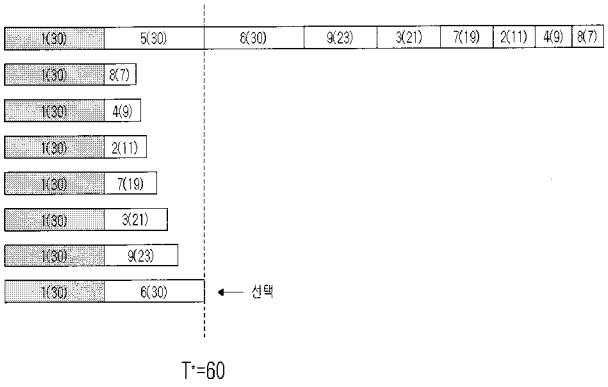
<표 10> 최적배분의 해가 한 개인 경우 휴리스틱 알고리즘의 평활지수

작업 수 \ 작업자 수 / 예		3	4	6
9	1	0	0	0
	2	0	0	0
12	1	2.24	0	0
	2	0	0	0
15	1	2.24	2.45	0
	2	2.24	2.45	0

<표 10>에서 평활지수가 0이라는 것은 휴리스틱 알고리즘으로 평활지수 0인 최적배분의 해를 찾은 것이다. 예 1, 2는 각 경우에 대해 작업시간을 다르게 하여 두 번 실험한 것이다. 이 표에 의하면 작업의 수가 고정된 경우 작업자의 수가 증가할수록 최적배분의 해를 발견하기 쉽다. 그 이유는 작업자의 수가 증가하면 한 명의 작업자가 맡는 작업의 수가 줄어들어서 최적배분의 해를 쉽게 발견할 수 있다. 그리고 작업자의 수가 고정된 경우라도 작업의 수가 감소하면 최적배분의 해를 발견하기 쉽다. 그 이유는 작업의 수가 감소한다는 것은 상대적으로 작업자 수가 증가하는 것과 동일한 효과가 있기 때문이다. 따라서 휴리스틱 알고리즘은 평활지수 0인 최적배분의 해가 한 개인 경우, 작업의 수가 감소하거나 작업자 수가 증가하면 최적배분의 해를 쉽게 찾을 수 있다.

3.2 최적배분의 해가 여러 개인 경우

이 절에서는 평활지수 0인 최적배분의 해가 여러 개 있는 경우로써, 휴리스틱 알고리즘이 최적배분의 해를 어느 정도 찾을 수 있는지 실험한다. <표 11>, <표 12>는 9개의 작업을 3, 4명의 작업자에게 공평하게 배분하는 예이다. <표 11>의 형식은 앞 절의 <표 1>과 동일하므로 설명은 생략한다. 작업자가 6명인 경우는 평활지수 0인 최적배분의 해가 단 한 개가 존재하므로 실험에서 제외하였다. <표 11>, <표 12>에서 작업자가 3명인 경우, 휴리스틱 알고리즘은 최적배분의 해를 발견하였다. 여기서 작업배분 결과는 평활지수 0인 최적배분과 휴리스틱 알고리즘 수행결과가 서로 다르게 나왔다. 예컨대 작업자 3명인 경우, 평활지수 0인 최적배분의 해는 작업자 A에 대해 1번 작업(30초), 3번 작업(21초), 4번 작업(9초)이지만 휴리스틱 알고리즘은 1번 작업(30초), 6번 작업(30초)이었다. 휴리스틱 알고리즘이 작업자 A에게 1번 작업(30초)과 6번 작업(30초)을 배분한 이유는 2.2절의 단계 1 작업배분의 (4)번 규칙으로 설명된다. (4)번 규칙은 시간이 가장 큰 1번 작업(30초)에 나머지 모든 작업을 작은 순서대로 한 번씩 붙여보는 것이다. 이때 6번 작업(30초)은 1번 작업(30초)과 붙고 총 작업시간이 T^*=60과 같아지기 때문에 6번 작업(30초)은 1번 작업과 함께 작업자에게 배분된다. <그림 8>은 이 과정을 보여준다.



<그림 8> 단계 1의 작업배분 과정

다음은 12, 15개의 작업을 3, 4, 6명의 작업자에게 배분하는 실험으로 결과는 <표 13>~<표 18>과 같다.

<표 11> 9개 작업을 3명에게 배분

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	합
작업시간	30	11	21	9	30	30	19	7	23	180
작업자3명	평활지수 0인 최적배분					휴리스틱 알고리즘 수행결과				
	작업자 A : 1, 3, 4 (작업시간 : 60) 작업자 B : 2, 5, 7 (작업시간 : 60) 작업자 C : 6, 8, 9 (작업시간 : 60)					작업자 A : 1, 6 (작업시간 : 60) 작업자 B : 5, 8, 9 (작업시간 : 60) 작업자 C : 3, 7, 2, 4 (작업시간 : 60) 평활지수 : 0				

<표 12> 9개 작업을 4명에게 배분

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	합
작업시간	10	20	8	22	12	18	7	12	11	120
작업자4명	평활지수 0인 최적배분					휴리스틱 알고리즘 수행결과				
	작업자 A : 1, 2 (작업시간 : 30) 작업자 B : 3, 4 (작업시간 : 30) 작업자 C : 5, 6 (작업시간 : 30) 작업자 D : 7, 8, 9 (작업시간 : 30)					작업자 A : 4, 3 (작업시간 : 30) 작업자 B : 2, 1 (작업시간 : 30) 작업자 C : 6, 8 (작업시간 : 30) 작업자 D : 5, 9, 7 (작업시간 : 30) 평활지수 : 0				

<표 13> 12개 작업을 3명에게 배분

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	10	11	12	합
작업시간	13	8	6	13	6	10	9	15	5	14	11	10	120
작업자3명	평활지수 0인 최적배분							휴리스틱 알고리즘 수행결과					
	작업자 A : 1, 2, 3, 4 (작업시간 : 40) 작업자 B : 5, 6, 7, 8 (작업시간 : 40) 작업자 C : 9, 10, 11, 12 (작업시간 : 40)							작업자 A : 8, 9, 5, 10 (작업시간 : 40) 작업자 B : 1, 3, 2, 4 (작업시간 : 40) 작업자 C : 11, 6, 12, 7 (작업시간 : 40) 평활도지수 : 0					

<표 14> 12개 작업을 4명에게 배분

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	10	11	12	합
작업시간	13	7	13	23	9	12	16	17	15	20	7	8	160
작업자4명	평활지수 0인 최적배분							휴리스틱 알고리즘 수행결과					
	작업자 A : 6, 9, 1 (작업시간 : 40) 작업자 B : 2, 3, 10 (작업시간 : 40) 작업자 C : 7, 11, 8 (작업시간 : 40) 작업자 D : 4, 5, 12 (작업시간 : 40)							작업자 A : 4, 8 (작업시간 : 40) 작업자 B : 10, 2, 1 (작업시간 : 40) 작업자 C : 7, 11, 12, 5 (작업시간 : 40) 작업자 D : 9, 3, 6 (작업시간 : 40) 평활도지수 : 0					

<표 15> 12개 작업을 6명에게 배분

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	10	11	12	합
작업시간	7	30	3	11	29	16	14	10	19	21	40	40	240
작업자6명	평활지수 0인 최적배분							휴리스틱 알고리즘 수행결과					
	작업자 A : 1, 2 (작업시간 : 40) 작업자 B : 3, 4 (작업시간 : 40) 작업자 C : 5, 6 (작업시간 : 40) 작업자 D : 7, 8 (작업시간 : 40) 작업자 E : 9, 10 (작업시간 : 40) 작업자 F : 11, 12 (작업시간 : 40)							작업자 A : 11 (작업시간 : 40) 작업자 B : 12 (작업시간 : 40) 작업자 C : 2, 8 (작업시간 : 40) 작업자 D : 5, 4 (작업시간 : 40) 작업자 E : 10, 9 (작업시간 : 40) 작업자 F : 6, 7, 1, 3 (작업시간 : 40) 평활도지수 : 0					

<표 16> 15개 작업을 3명에게 배분

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	합
작업시간	4	9	15	21	11	6	12	16	23	3	8	13	14	19	6	180

	평활지수 0인 최적배분	휴리스틱 알고리즘 수행결과
작업자3명	작업자 A : 1, 2, 3, 4, 5 (작업시간 : 60) 작업자 B : 6, 7, 8, 9, 10 (작업시간 : 60) 작업자 C : 11, 12, 13, 14, 15 (작업시간 : 60)	작업자 A : 1, 11, 6, 14, 9 (작업시간 : 60) 작업자 B : 15, 2, 5, 12, 4 (작업시간 : 60) 작업자 C : 10, 7, 13, 3, 8 (작업시간 : 60) 평활도지수 : 0

<표 17> 15개 작업을 4명에게 배분

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	합
작업시간	4	11	10	15	10	9	18	13	16	13	14	17	29	6	15	200

	평활지수 0인 최적배분	휴리스틱 알고리즘 수행결과
작업자4명	작업자 A : 1, 2, 3, 4 (작업시간 : 50) 작업자 B : 5, 6, 7, 8 (작업시간 : 50) 작업자 C : 9, 10, 11, 12 (작업시간 : 50) 작업자 D : 13, 14, 15 (작업시간 : 50)	작업자 A : 13, 1, 12 (작업시간 : 50) 작업자 B : 14, 5, 9, 7 (작업시간 : 50) 작업자 C : 4, 3, 2, 11 (작업시간 : 50) 작업자 D : 6, 10, 8, 15 (작업시간 : 50) 평활도지수 : 0

<표 18> 15개 작업을 6명에게 배분

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	합
작업시간	7	11	12	8	13	9	9	6	15	10	20	11	19	12	18	180

	평활지수 0인 최적배분	휴리스틱 알고리즘 수행결과
작업자6명	작업자 A : 1, 2, 3 (작업시간 : 30) 작업자 B : 4, 5, 6 (작업시간 : 30) 작업자 C : 7, 8, 9 (작업시간 : 30) 작업자 D : 10, 11 (작업시간 : 30) 작업자 E : 12, 13 (작업시간 : 30) 작업자 F : 14, 15 (작업시간 : 30)	작업자 A : 11, 10 (작업시간 : 30) 작업자 B : 13, 2 (작업시간 : 30) 작업자 C : 15, 3 (작업시간 : 30) 작업자 D : 9, 8, 6 (작업시간 : 30) 작업자 E : 1, 12, 14 (작업시간 : 30) 작업자 F : 4, 7, 5 (작업시간 : 30) 평활도지수 : 0

<표 13>~<표 18>과 같이 휴리스틱 알고리즘은 모든 경우에 대해 평활지수 0인 최적배분의 해를 찾았다.

추가로 9, 12, 15개 작업에 대해 작업시간을 다르게 하여 작업배분 실험을 2회 반복한 결과, 휴리스틱 알고리즘은 98% 이상의 최적배분의 해를 찾을 수 있었다.

따라서 휴리스틱 알고리즘은 평활지수 0인 최적배분의 해가 여러 개인 경우, 3.1절의 최적배분의 해가 한 개인 경우보다 최적배분의 해를 보다 쉽게 찾을 수 있다. <표 19>는 <표 13>~<표 18>을 정리한 표이다.

<표 19> 최적배분의 해가 여러 개인 경우 휴리스틱 알고리즘의 평활지수

작업 수 \ 작업자 수	3	4	6
9	0	0	-
12	0	0	0
15	0	0	0

3.3 최적배분의 해가 없는 경우

여기에서는 평활지수 0인 최적배분의 해가 없는 경우로써, 작업과 작업자 수의 증가에 따른 휴리스틱 알고리즘의 평활지수 변화를 확인한다. 실험은 9, 12, 15개의 작업을 3, 4, 6명의 작업자에게 균등하게 배분하는 문제이다. <표 20>은 휴리스틱 알고리즘으로 9개의 작업을 3, 4, 6명의 작업자에게 균등하게 배분한 결과이다. 이 절의 예들은 최적배분의 해가 없는 경우이기 때문에 휴리스틱 알고리즘으로 찾은 최적배분의 해 만을 표시하였다. 예컨대 <표 20>에서 작업자 3명인 경우는 작업자 A에게 6, 9, 4번 작업을 할당하고 작업시간의 합은 40초라는 의미이며 그 때의 평활지수는 1.41이라는 것을 의미한다.

<표 20> 9개의 작업

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	합
작업시간	19	13	11	15	7	20	18	10	5	118

작업자 3명	작업자 A : 6, 9, 4 (작업시간 : 40) 작업자 B : 1, 5, 7 (작업시간 : 44) 작업자 C : 2, 3, 8 (작업시간 : 34) 평활지수 : 1.41	작업자 4명	작업자 A : 6, 8 (작업시간 : 30) 작업자 B : 1, 3 (작업시간 : 30) 작업자 C : 2, 4 (작업시간 : 28) 작업자 D : 9, 5, 7 (작업시간 : 30) 평활지수 : 2
작업자 6명	작업자 A : 6 (작업시간 : 20) 작업자 B : 1 (작업시간 : 19) 작업자 C : 7 (작업시간 : 18) 작업자 D : 4, 9 (작업시간 : 20) 작업자 E : 2, 5 (작업시간 : 20) 작업자 F : 3, 8 (작업시간 : 21) 평활지수 : 4		

<표 21> 12개의 작업

(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	10	11	12	합
작업시간	30	9	10	10	11	4	24	23	9	14	18	22	184

작업자 3명	작업자 A : 2, 12, 1 (작업시간 : 61) 작업자 B : 9, 4, 11, 7 (작업시간 : 61) 작업자 C : 3, 5, 10, 8, 6 (작업시간 : 62) 평활지수 : 1.41	작업자 4명	작업자 A : 1, 11 (작업시간 : 48) 작업자 B : 7, 12 (작업시간 : 46) 작업자 C : 8, 6, 2, 4 (작업시간 : 46) 작업자 D : 10, 5, 3, 9 (작업시간 : 44) 평활지수 : 4.90
작업자 6명	작업자 A : 1 (작업시간 : 30) 작업자 B : 6, 7 (작업시간 : 28) 작업자 C : 8, 9 (작업시간 : 32) 작업자 D : 12, 4 (작업시간 : 32) 작업자 E : 11, 10 (작업시간 : 32) 작업자 F : 2, 3, 5 (작업시간 : 30) 평활지수 : 4.90		

<표 22> 15개의 작업

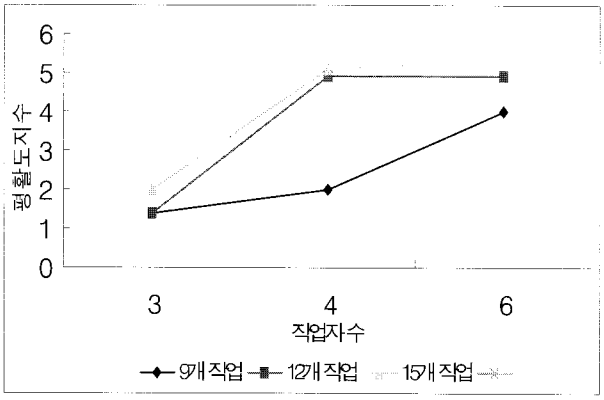
(단위 : 초)

작업번호	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	합
작업시간	21	9	11	12	8	7	14	13	6	18	8	29	10	21	23	127

작업자 3명	작업자 A : 11, 9, 6, 5, 10 (작업시간 : 68) 작업자 B : 2, 12, 3, 8, 14 (작업시간 : 66) 작업자 C : 4, 1, 7, 13 (작업시간 : 68) 평활지수 : 2	작업자 4명	작업자 A : 11, 14 (작업시간 : 52) 작업자 B : 13, 9, 6, 10 (작업시간 : 52) 작업자 C : 1, 5, 2, 8 (작업시간 : 51) 작업자 D : 7, 4, 3, 12 (작업시간 : 47) 평활지수 : 5.10
작업자 6명	작업자 A : 11, 9 (작업시간 : 35) 작업자 B : 14, 3 (작업시간 : 34) 작업자 C : 13, 8 (작업시간 : 34) 작업자 D : 1, 7 (작업시간 : 35) 작업자 E : 10, 6, 2 (작업시간 : 34) 작업자 F : 4, 12, 5 (작업시간 : 30) 평활지수 : 5.29		

<표 20>~<표 22>의 모든 실험결과에서 작업자가 3명인 경우의 평활지수는 각각 1.41, 1.41, 2로 낮게 나왔다. 이것은 작업자 수가 감소하면 작업자에게 배분할 수 있는 작업의 수가 증가하기 때문이다. 즉, 작업자에게 배분할 수 있는 작업의 수가 증가하면 작업시간 합의 편차를 줄일 수 있는 작업배분 경우의 수도 증가한다. 따라서 평활지수는 감소한다.

반대의 경우에는 작업시간 합의 편차를 줄일 수 있는 경우의 수가 감소하여 <그림 9>와 같이 평활지수는 증가한다.



<그림 9> 작업과 작업자 수에 따른 평활지수 변화

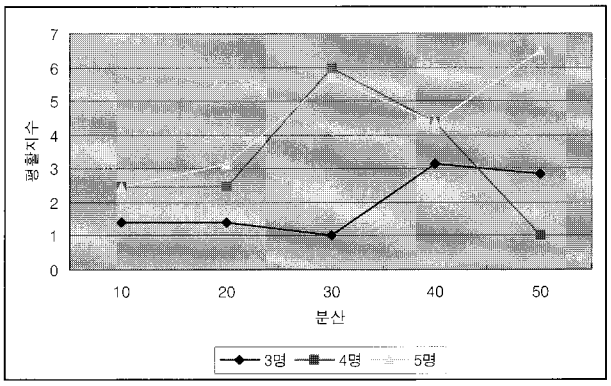
3.4 작업시간 분산과 평활지수 간의 관계

다음은 작업시간의 분산과 평활지수간의 관계를 확인하는 실험이다. 작업의 개수는 9, 12, 15, 18개로 하였고 각 경우의 분산은 10, 20, 30, 40, 50으로 설정하였다. 작업자의 수는 앞의 실험과 동일하게 3, 4, 6명으로 하였다

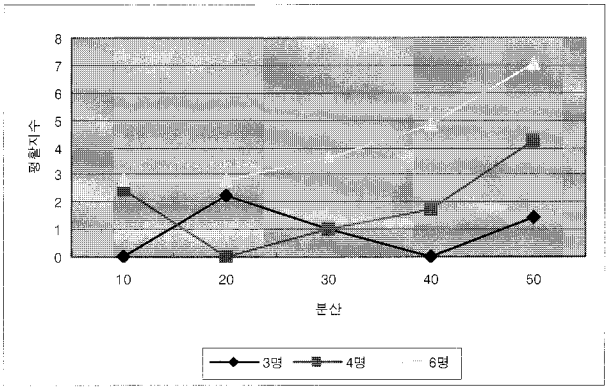
<표 23> 작업시간의 분산에 따른 휴리스틱 알고리즘의 평활지수

작업 / 작업자 / 분산	9			12			15		
	3	4	6	3	4	6	3	4	6
10	1.41	2.45	2.45	1.41	1	6.48	1.41	2.45	2.83
20	1.41	2.45	3.16	3.16	2.24	4.69	2.24	2.24	2.83
30	1	6	5.7	1.41	2.83	1.41	1	1	3.61
40	3.16	4.36	4.36	4.12	2.24	4	2.24	1.73	4.80
50	2.83	1	6.48	1	3.32	3.32	1.41	4.24	7.07

실험결과는 <표 23>과 같이 나타났다. <그림 10>, <그림 11>의 그래프는 <표 23>의 결과에서 작업의 개수가 9, 15개인 경우를 표시한 것이다. <그림 10>, <그림 11>과 같이 9개의 작업을 3, 6명의 작업자에게 배분했을 경우와 15개의 작업을 4, 6명의 작업자에게 배분하였을 경우, 평활지수는 작업시간의 분산이 커질수록 대체로 증가하였다. 또한 평활지수는 작업자의 수가 증가할수록 증가하였는데 이것은 최적배분의 해가 없는 경우에서 작업자의 수가 증가하면 평활지수가 증가한다는 <그림 9>의 결과와 동일하다.



<그림 10> 작업시간 분산에 따른 휴리스틱 알고리즘의 평활지수 변화 - 9개 작업



<그림 11> 작업시간 분산에 따른 휴리스틱 알고리즘의 평활지수 변화 - 15개 작업

4. 결 론

본 연구는 병렬라인 시스템에서 작업순서가 없는 검사공정을 대상으로 복수의 작업자에게 작업을 배분하는 휴리스틱 알고리즘을 개발하였다. 휴리스틱 알고리즘의 성능은 실험을 통하여 확인하였으며 그 효과 및 특성은 다음과 같다.

먼저 휴리스틱 알고리즘의 효과는 다음과 같이 세 종류의 실험을 통해 확인되었다. 첫째, 평활지수 0인 최적배분의 해가 한 개인 경우, 휴리스틱 알고리즘은 최적배분의 해를 72% 정도의 비율로 찾을 수 있었다.

둘째, 평활지수가 0인 최적배분의 해가 여러 개인 경우, 본 알고리즘은 최적배분의 해를 98% 정도의 비율로 찾을 수 있었다. 셋째, 평활지수 0인 최적배분의 해가 없는 경우, 본 알고리즘은 평활지수 6.0을 넘지 않는 범위에서 최적배분의 해를 찾을 수 있었다. 특히 최적배분의 해가 여러 개인 경우, 최적배분의 해를 98% 정도로 찾을 수 있어 제한된 범위 내에서는 본 알고리즘의 뛰어난 성능을 확인할 수 있었다.

다음으로 휴리스틱 알고리즘의 특성은 다음과 같다.

첫째, 평활지수 0인 최적배분의 해가 한 개인 경우, 본 알고리즘은 작업의 개수를 고정시키고 작업자 수를 증가시키면 최적배분의 해를 쉽게 발견할 수 있다. 둘째, 평활지수가 0인 최적배분의 해가 여러 개인 경우, 본 알고리즘은 작업의 개수나 작업자 수는 최적배분의 해를 찾는데 거의 영향을 주지 않는다. 셋째, 평활지수 0인 최적배분의 해가 없는 경우, 본 알고리즘은 작업자 수가 적을수록 평활지수는 감소한다. 넷째, 본 알고리즘의 평활지수는 작업시간 분산에 대체로 비례한다.

본 연구에서 제안한 휴리스틱 알고리즘은 최적배분의 해를 짧은 시간에 찾을 수 있기 때문에 병렬라인의 검사공정에서 발생하는 작업배분에 소요되는 계획시간을 크게 줄일 수 있다. 또한 이 알고리즘을 사용하여 작업배분을 할 경우 모든 작업자가 동시에 작업을 마칠 수 있고 작업의 불공평배분 문제를 방지할 수 있다.

이로 인한 기대효과로는 작업의 불공평 배분에서 비롯되는 작업자의 육체적 피로의 증가를 감소시킬 수 있고 정신적 스트레스를 예방할 수 있어서 검사작업의 질적 향상을 꾀할 수 있다.

5. 참 고 문 헌

[1] 김용호, 김영균, 오길호, "이동 에이전트 환경에서 최적화된 작업할당 방법에 대한 연구.", 한국정보과학회, 28 (2001) : 622-624

[2] 이근철, "혼합 흐름공정에서 라인밸런싱을 위한 휴리스틱 개발.", 한국산업경영시스템학회, 30 (2007) : 94-102

[3] 이명호, 유지수, "생산관리.", 박영사 (1999)

[4] 이상범, "현대생산운영관리.", 명경사 (2007)

[5] 이종환, "A Heuristic Approach for Workload Balancing Problems.", 한국산업경영시스템학회, 춘계학술대회 (2006) : 204-210

[6] 손일문, 이동춘, 이상도, "검사작업에서의 인지기술과 인간수행도 분석에 관한 연구.", 대한인간공학회, 1992년 학술대회논문집 제2권 (1992) : 90-95

[7] Becker, C., Scholl, A., "A servay on problems and method in generalized assembly line balancing.", European Journal of Operation Research, 168 (2006) : 694-715

[8] Leonid Oliker, "PLUM: Parallel Load Balancing for Adaptive Unstructured Meshed.", Journal of Parallel and Distributed Computing, 52 (1998) : 150-177

[9] Miltenburg, G. J., Wijngaard, J., "The U-line Balancing Problem.", Management science, 40 (1994) : 1378-1388

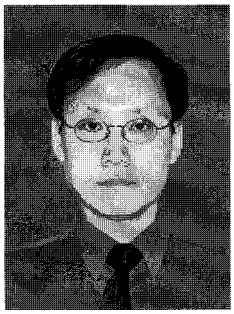
[10] M.Y. Wu, "On Runtime Parallel Scheduling for Processor Load Balancing.", IEE Tran. on Parallel and Distributed System, 8 (1997) : 173-185

[11] Scholl, A., Klein, R., "ULINO: Optimally balancing U-shaped JIT assembly lines.", International Journal of Production Research, 37 (1999) : 721-736

[12] Sparling, D., Miltenburg, J., "The mixed-model U-line balancing problem.", International Journal of Production Research, 36 (1998) : 485-501

저 자 소 개

이 석 환



인하대학교 산업공학과에서 공학사 및 공학석사를 취득 하였다. 현재 유한대학 전임강사로 재직중이다. 주요 관심 분야는 데이터 마이닝이다.

주소: 인천광역시 남구 용현동 253, 인하대학교 산업공학과

박 승 헌

인하대학교 금속공학과에서 공학사, 일본 Keio대학 관리공학과에서 공학석사 및 공학박사를 취득하였다. 현재 인하대학교 산업공학과 교수로 재직 중이다. 주요 관심 분야는 FMS와 각종 생산시스템의 설계 및 운영, 인터넷 마케팅과 데이터 마이닝 등이다.

주소: 인천광역시 남구 용현동 253, 인하대학교 산업공학과