Journal of the Society of Korea Industrial and Systems Engineering
Vol, 30, No. 1, pp.115－121, March 2007.

기계 가용성 제약을 고려한 흐름공정 상황하에서 Makespan을 최소화하기 위한 향상된 유전 알고리듬

이경화 · 정인재†

한양대학교 산업공학과

An Improved Genetic Algorithm to Minimize Makespan in Flowshop with Availability Constraints

Kyung-Hwa Lee · In-Jae Jeong†

Department of Industrial Engineering, Hanyang University

In this paper, we study flowshop scheduling problems with availability constraints. In such problems, n jobs have to be scheduled on m machines sequentially under assumption that the machines are unavailable during some periods of planning horizon. The objective of the problem is to find a non-permutation schedule which minimizes the makespan. As a solution procedure, we propose an improved genetic algorithm which utilizes a look-ahead schedule generator to find good solutions in a reasonable time. Computational experiments show that the proposed genetic algorithm outperforms the existing genetic algorithm.

Keywords : Flowshop scheduling, Machine availability, Schedule generator

1. 서 론

기존의 많은 흐름공정(flowshop)문제에서는 n개의 작업들이 m대의 기계를 모두 거칠 때 기계가 언제나 가용 가능하다는 가정을 한다. 하지만 이러한 가정은 실제 산업현장의 상황과 비추어 보았을 때, 현실적이지 못한 경우가 있다. 예를들면, 기계가 오작동할 경우 기계를 멈추고 수리해야 할 때 기계가 가용하지 않게 되고, 기계를 정기적으로 점검해야 하는 상황에서도 기계의 비가용구간이 존재하게 된다. 본 연구에서는 이러한 기계가용성 제약을 고려한 흐름공정 문제에 대해 다루고자 한다.

기계가용성 제약을 고려한 기존의 일정계획문제는 Schmidt (1984)에 의해 처음으로 연구되었다. Schmidt(1984)는 병렬기계 상황 하에서 작업의 일정을 결정하는 문제를 다루었고 Adiri et al.(1989)은 단일 기계 상황에서 기계가용 구간이 고정된 경우 문제가 NP-hard임을 보였다.

Lee(1991)는 기계가용성 제약을 고려한 병렬기계 상황 하에서 일정계획문제에 대해 변형된 LPT(longest processing time)알고리듬의 오차한계(error bound)가 1/2임을 보여주었다. 또한 같은 문제에 대해 Lee and Liman(1992)이 기존의 방법보다 간단한 방법으로 문제가 NP-hard임을 증명 하였다. Lee(1995)는 두 기계가 존재하는 흐름공정 상황 하에서 한 대의 기계에 비가용구간이 존재하는 문제에 대해 NP-hard임을 증명하였고, 동적계획으로 문제를 풀었으며, 오차한계를 가지는 휴리스틱 알고리듬을 제안하였다. Lee(1997)는 같은 문제 상황에서 작업이 resumable한 경우에 대해 NP-hard임을 증명하였고,

† 교신저자 E-mail : ijeong@hanyang.ac.kr

$O(nlogn)$의 복잡도(complexity)를 가지는 알고리듬을 제안하였다.

또한, Lee(1999)는 작업이 semi-resumable하고 비가용구간이 한 기계에만 존재한다는 가정이 있는 문제와 두 기계 모두 기계 비가용구간이 한 구간 존재하는 문제에 대해 NP-hard임을 증명했다. 그리고 동적계획 알고리듬과 오차한계를 가지는 휴리스틱 알고리듬을 제안하였다.

Blazewicz, J. et al.(2001)은 두 기계가 존재하는 흐름공정 상황하에 두 기계 모두 비가용구간이 존재하는 문제에 대해 휴리스틱 알고리듬을 제안하였다. Kubiak et al.(2002)에 의해 같은 상황에서 기계 비가용구간이 한 기계에 K개 존재하는 문제에 대해 NP-hard임을 증명하였고, 분지 한계법을 제시하였다.

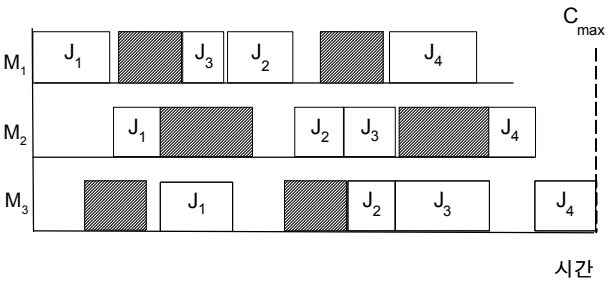
최근에는 Aggoune(2004)에 의해 m대의 기계 모두에 기계 비가용구간이 정확히 두 구간이 존재하는 경우에 makespan을 최소화 하는 문제에 대해 처음으로 다루고 있으며, 이 문제 역시 NP-hard임을 증명하였고, 이 문제에 대해 순열(Permutation) 스케줄이 최적해를 보장할 수 없다고 보였다. 또한, 이 문제에 대한 해법으로 유전 알고리듬과 타부서치 알고리듬을 제안하였다. 실험결과 두 알고리듬간의 성능에 대한 차이가 거의 없는 것으로 나타났다. 따라서 본 연구에서는 Aggoune(2004)이 제안한 알고리듬보다 향상된 유전 알고리듬을 제안하고 이를 실험을 통해 보이고자 한다.

본 연구는 제 2장에서 문제정의, 제 3장에서 유전 알고리듬의 개요, Aggoune(2004)의 알고리듬, 제안하는 알고리듬에 대해 설명한다. 그리고 제 4장에서는 두 알고리듬을 비교하기 위한 실험방법과 실험결과를 설명하고 마지막으로 제 5장에서 결론을 맺고자 한다.

2. 문제 정의

기계가용성 제약을 고려한 흐름공정상황의 일정계획 문제는 <그림 1>의 간트 차트를 이용하여 설명하면 아래와 같다.

<그림 1>은 기계가 3대 있고, 작업의 수가 4로 구성된 흐름공정을 나타낸다. J_i는 작업 i를 나타내고, J_i가 차지하는 구간의 길이는 작업시간을 나타낸다. 각각의 기계에는 기계를 가용할 수 없는 구간 즉, 기계 비가용구간이 정확히 두 구간 존재 하는데 <그림 1>에서는 빗금친 구간이 기계 비가용 구간을 의미한다. 이러한 비가용구간이 시작하는 시점은 기계마다 다르며, 구간의 크기 또한 다르다. Lee(1995)의 연구에서 살펴보았듯이 두 기계가 존재하는 흐름공정 상황하에서 기계 비가용구간이 존재하는 문제에 대해 NP-hard이므로 본 연구에서 고려하는 다수의 기계가 존재하고 기계마다 비가용구간이 두 구간 존재하는 문제 역시 NP-hard이다. 이러한 문제에 대해서 Aggoune(2004)의 연구에서 순열 스케줄이 꼭 좋은 해를 얻는 것은 아님을 증명을 하였다. 따라서 본 연구에서도 비순열(non-permutation)스케줄을 허용하고 작업이 non-preemptive함을 가정한다. 이러한 문제 상황에서 makespan을 최소화하는 해를 구하는 것이 이 문제의 목적이다. Aggoune(2004)은 위의 문제에 대한 해법으로 유전 알고리듬과 타부서치를 제시하였는데 실험결과 두 알고리듬이 유사한 성능을 가진다고 보고하였다.



<그림 1> 기계가용성 제약을 고려한 flowshop 공정

3. 제안하는 유전 알고리듬

본 장에서는 이 문제를 해결하기 위한 유전 알고리듬의 절차에 대해 설명한다. 먼저 Aggoune(2004)이 제안한 Schedule generator를 기반으로 한 유전 알고리듬에 대해 설명하고, 본 연구에서 제안하는 Schedule generator를 설명하고자 한다.

3.1 Aggoune의 유전 알고리듬

유전 알고리듬은 자연선별과 유전법칙을 모방한 확률적 탐색기법이다. 이 알고리듬은 Holland에 의해 1975년 처음으로 소개되었는데 복수개의 잠재해들로 이루어진 해의 집단을 운용하여 해 공간을 탐색하는 방법을 말한다. 유전 알고리듬은 좋은 해의 세대유전을 통해 좋은 해를 빠른 시간에 탐색할 수 있다. 조기 수렴하여 부분최적에 빠질 수 있고 해 공간의 탐색만을 강조하면 임의 탐색에 가까워 좋은 해를 찾아가지 못하게 되는 단점도 존재한다(김여근 외, 1999). 하지만 유전자 알고리듬은 일정계획, 그룹 테크놀로지, 외판원 문제 등과 같이 조합 최적화문제의 최적 또는 유사 최적해를 구하는 데 강력한 도구로써 사용되고 있다(Gen, M. and Cheng, 1997).

Aggoune(2004)이 제안한 유전 알고리듬의 전반적인 과정을 요약하면 다음과 같다. 초기해를 임의생성방법으

로 모집단 크기만큼 생성하고, 생성한 모집단의 적응도를 평가한다. 이때 염색체를 스케줄링 문제의 해로 전환 시킨 뒤 makespan을 계산하여야 하는데 이 역할을 하는 것이 Schedule generator이다. Schedule generator는 각각의 작업의 선후행 관계를 만족시키도록 설계되어야 한다. 다음 절에서 Aggoune(2004)이 제안한 Schedule generator 방법을 보다 자세히 설명하고 개선점을 알아보도록 한다. 적응도 평가가 끝나면 선별을 하게 되는데 이때는 확률바퀴 방법을 사용하여 선별한다. 그 후 교차를 하여 세대를 유전하고 돌연변이 절차를 거쳐 새로운 모집단을 생성한다. 종료조건에 만족하지 않으면 다시 적응도평가를 하여 위의 과정을 반복하고, 종료조건에 만족하면 종료하는 절차로 이루어져 있다.

염색체 하나는 이 문제의 해 하나를 표현하고 있다. 염색체를 구성하고 있는 각각의 유전인자(gene)는 기계에 투입하는 작업을 의미한다. 기계는 m대 이므로, (작업 수)*(기계 수)가 유전인자의 개수가 된다. 이렇게 구성된 염색체를 모집단 크기만큼 생성하고 세대유전을 한다. Aggoune(2004)에서 사용된 교차방법은 순서교차의 방법 중에 하나의 절단점을 중심으로 교차하는 방법을 사용하였다. 돌연변이는 교차를 통해 생성된 해의 탐색공간이 국지적으로 빠지지 않기 위해 하는 것으로 임의로 한 유전인자를 선택하여 임의로 선택된 다른 유전인자와 맞바꾸는 방법을 사용하였다.

3.2 Aggoune의 Schedule generator

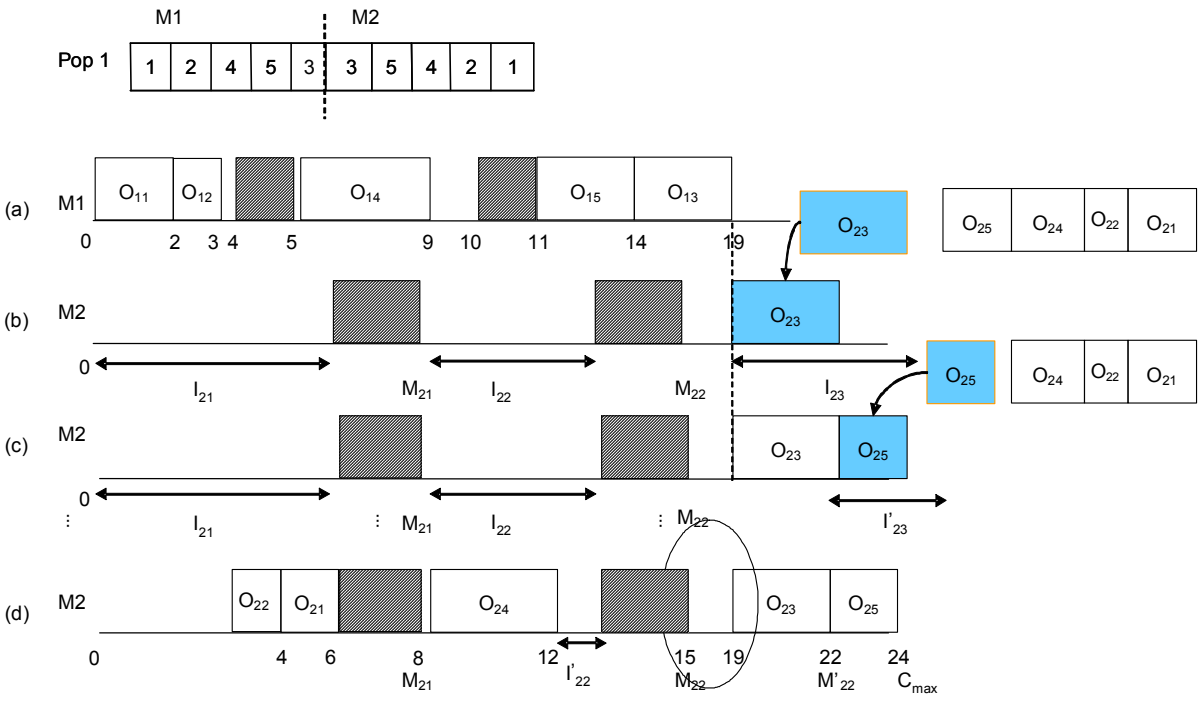
Schedule generator는 앞서 말한 바와 같이 구하고자 하는 해의 makespan을 계산해주고, 스케줄을 가능해로 만드는 역할을 한다. 따라서, Schedule generator에 의해 일정이 결정되는데, 이것으로 인해 각 기계마다 할당되는 작업의 순서가 달라질 수 있다.

Schedule generator의 구체적인 방법에 대해서 예를 들어 자세히 설명하고자 한다. 고려하는 예제는 두 대의 기계로 이루어진 흐름공정에서 5개의 작업을 해야 하는 상황으로 작업시간은 <표 1>에 나타나 있다. 기계 비가용 구간의 범위는 기계1에[4, 5, 10, 11], 기계2에[6, 8, 13, 15]로 기계마다 각각 두 구간 존재한다.

<표 1> 작업시간

작업 / 기계	J_1	J_2	J_3	J_4	J_5
M1	2	1	4	3	5
M2	2	1	3	3	2

<그림 2>는 위의 예제를 Aggoune(2004)이 제안한 Schedule generator를 일회 적용시킨 결과를 나타내고 있다. 먼저 표지(notation)를 살펴보면 O_{ij}는 기계 i에서 작업 j의



<그림 2> Riad Aggoune의 알고리듬의 구현의 예

작업을 의미한다. I_{il}는 기계 i에서 비가용구간 사이의 구간 l의 크기를 의미하고, 비가용구간 사이 구간의 개수는 K(비가용구간의 개수)+1개 존재한다. M_{ik}는 기계 i의 k번째 비가용구간이 끝나는 시간을 의미한다.

<그림 2>에서 $Pop1$은 초기 모집단의 하나의 해를 나타내고 있다. $Pop1$ 안에 숫자로 표현된 것이 각각의 작업을 나타내며, $Pop1$은 작업의 투입순서를 표현하고 있다. a)에 나타난 바와같이 초기에 $Pop1$의 순서로 기계 1에 각 Job의 첫 번째 작업(operation)을 할당하였다.

<그림 2>의 b), c), d)는 Aggoune에 의해 제안된 Schedule generator의 절차에 따라 기계 2에 작업을 할당하는 것을 나타내고 있다. b)를 보면 기계 2의 초기 작업 투입순서가 3-5-4-2-1순이므로, 먼저 기계 2에 J_3의 두 번째 작업, O_{23}을 할당하고, 두 번째로 J_5의 두 번째 작업, O_{25}을 할당하는 순서로 마지막 J_1의 두 번째 O_{21}을 할당하고 끝나게 된다. O_{23}을 기계 2에 할당하고자 할 때 가능한 곳은 I_{23}이다. 왜냐하면 기계 1에서 O_{13}이 마친 시점이 I_{23}에 해당하기 때문이다. c)에 나타난 것처럼 다음의 O_{25}을 할당하는데 이것역시 I_{23}에 할당해야 하므로, I_{23}에 할당한다. d)를 보면, 다음 O_{24}는 I_{21}엔 할당할 수 없고, I_{22}에 할당 할 수 있으므로, I_{22}에 할당하고, I_{22}의 크기를 I'_{22}으로 하고 그 값을 $I'_{22}=I_{22}-O_{24}$로 갱신(Update) 한다. 이러한 과정을 거쳐 마지막 작업까지 모두 할당한다. 본 예제에서 작업을 모두 할당하여 나온 makespan의 값은 24이다.

위와같은 방법으로 하였을 경우 마지막 비가용 구간 후의 사이 구간에 작업을 할당할 때 작업의 투입순서대로 할당되기 때문에 선행조건을 고려할 경우 유휴시간이 발생시키게 된다. 이러한 요인이 makepan을 커지게 만든다. 또한 긴 작업시간을 가지는 작업이 앞쪽에 배치되게 되면, 작업시간이 긴 작업이 먼저 할당될 확률이 높아지고 이것은 다음기계에서 해당 작업이 할당될 수 있는 범위를 좁혀 makespan을 증가시키는 요인이 된다. 따라서 이러한 문제를 개선하여 유휴시간의 발생을 최소화한 Schedule generator를 제안하고자 한다. 다음 절에서 본 연구에서 제안하는 Schedule generator에 대해서 알아보고 그 절차를 설명하고자 한다.

3.3 제안하는 Schedule generator

먼저 표지에 대해서 살펴보면, P_{ij}는 기계 i에서 작업(Job) j가 작업하는 시간이고, S_i는 기계 i에서 가공하는 작업의 집합을 나타낸다. 제안하는 Schedule generator의 절차는 <그림 3>과 같다.

```
step1.  for( i=1 ; i<= mc ; i++)
    step1.1   for( l=1 ; l<= k    ; k++)
                  for( j=1 ; j <= |  Jil ; j++)
                        if Pij <= Iil 이면 할당
                          Si = Si – { j }
                          Iil = Iil-  Pij ,  j = j + 1
                        else
                          j = j + 1
    step1.2   for( l=K+1)
                  for( j=1 ; j<=   Ji ; j++)
                        if P-1j <= Mik 이면 할당
                          Si = Si – { j }
                          Iil = Iil-  Pij ,  j = j + 1
                        else
                          j = j + 1.
```

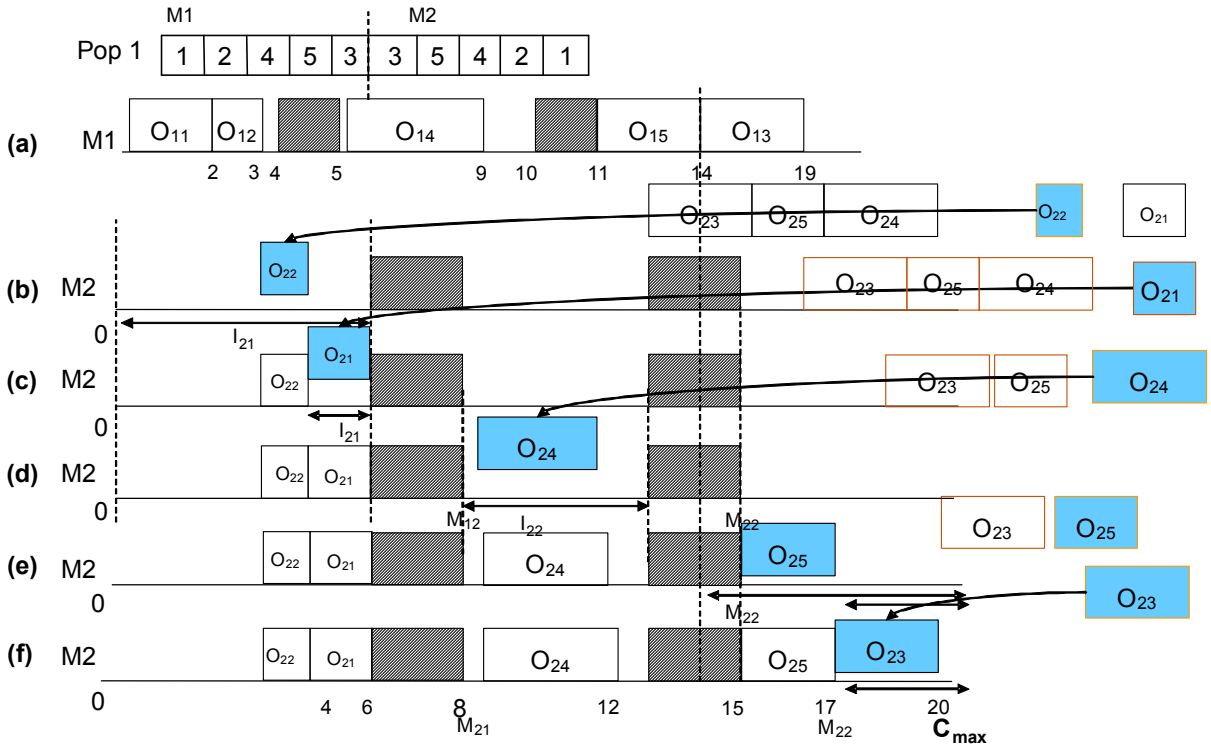
<그림 3> 제안하는 Schedule generator의 절차

Step1.1에서 P_{ij}가 I_{ij}보다 작으면, 작업 j를 사이구간(Interval)l에 할당하고 I_{ij}을 $I_{ij}-P_{ij}$로 갱신한다. 그렇지 않으면 작업의 할당을 잠시 미뤄두고 j를 하나 증가시킨다. 사이구간 l에 대해 모든 작업이 탐색되면 l을 하나 증가시키고, 할당되지 않은 작업 j를 위의 방법과 같이 할당한다.

마지막 사이구간 K+1은 step1.2의 방법을 따른다. 여기서 고려될 사항은 $i-1$번째 기계에서 작업 j가 마친 시점이다. 그것을 i번째 기계의 마지막 비가용구간이 마친 시간(M_{ik})과 비교하여 M_{ik}가 더 큰 경우 작업 j를 할당하고, M_{ik}를 $M_{ik}+P_{ij}$으로 갱신한다. 만약 그렇지 않으면 작업을 보류해 두고 다음 작업을 탐색한다. 모든 구간에 대해 탐색을 마치고, 기계 i에 j의 작업이 모두 할당되면, i를 증가시키고 step1.1과 1.2의 과정을 반복한다. 모든 기계에 모든 작업이 할당되면 탐색을 종료한다. 이해를 돕고자 <그림 4>의 예를 보며 설명하고자 한다.

<그림 4>는 3.2절에서 보여준 <그림 2>의 예제와 동일한 문제를 가지고 제안하는 Schedule generator의 방법으로 작업순서를 결정하고 있다. 앞서 말했듯이 O_{ij}는 i 기계에서 J_i의 작업(operation)을 의미한다. (a)에 나타난 바와 같이 기계 1에 이미 작업을 할당하였고, 기계 2에서 작업을 할당하는 과정을 <그림 4>의 (b)~(g)에서 나타내고 있다.

(b)에서 첫 번째 탐색구간인 I_{21}을 보자. 먼저 작업 O_{23}이 I_{21}에 할당될 수 있는지 여부를 탐색한다. 이때 기계 1에서 작업 O_{23}의 선행작업인 작업 O_{13}이 I_{21}이 이전에



<그림 4> 제안하는 알고리듬 구현의 예

끝나지 않으므로 O_{23}의 작업할당을 보류해두고, 다음 순서의 작업을 할당한다(step1.1). O_{25}, O_{24}의 선행 작업이 I_{21}구간이 끝나는 시점 전에 마쳐지지 않았으므로 작업의 할당을 보류해 둔다(step1.1). 다음 작업 O_{22}의 경우 작업시간이 I_{21}구간의 크기보다 작으므로 I_{21}에 할당하고, I_{21}의 값을 $I_{21}-P_{22}$(O_{22}의 작업시간)으로 갱신한다. 같은 방법으로 하여 (c)에 나타난 바와 같이 I_{21}구간에 작업 O_{21}을 할당한다. 이와같은 방법으로 (d)에서 O_{24}를 할당한다.

(e)에서 처럼 $l=K+1$이 되면(마지막 비가용구간 이후의 사이구간), step1.2를 따른다. I_{22}에서 할당되지 못한 작업O_{23}을 보자. 여기서 고려될 사항은 O_{23}의 작업이 기계 1에서 마친 시점이다. 즉, O_{13}의 마친 시점을 기계 2의 마지막 비가용구간이 마친 시간 즉, M_{22}와 비교하여 M_{22}가 더 크면, O_{23}을 할당하고, 그렇지 않으면 작업을 보류해 두고 다음 작업을 탐색한다. (e)에서 보면 M_{22}가 O_{13}이 마친 시점 보다 작다. 따라서 O_{23}의 작업할당은 보류되고, 다음 작업인 O_{25}을 탐색한다(step1.2). M_{22}가 O_{15}보다 크므로 O_{25}을 M_{22}바로 뒤에 할당하고 M_{22}를 $M_{22}+P_{25}$(O_{23}의 작업시간)으로 갱신한다(step1.2). (f)에서보면 할당되지 않은 작업 O_{23}을 갱신된 M_{22}와 비교하여 O_{23}을 M_{22}뒤에 할당 한다(step1.2). 모든 기계에 모든 작업이 할당되었으므로 종료한다.

이와 같은 방법으로 모든 기계의 모든 작업에 대해 기계에 작업을 할당한다. 그리고 나서 Schedule generator에 의해 makespan의 값을 얻는다. 예제를 나타낸 <그림 2>과 <그림 4>의 makespan을 비교하면 제안하는 알고리듬이 4의 크기(<그림 2>에서 동그라미 표시한 부분) 만큼 유휴시간을 줄일 수 있었고, 이것은 makespan을 줄이는데 효과적인 방법임을 보인다.

다음 장에서는 실험을 통하여 제안하는 알고리듬과 기존의 Aggoune(2004)에서 제안한 알고리듬의 결과를 비교하여 제안하는 알고리듬이 얼마나 향상되었는지 알아보고자 한다.

4. 실험결과

기존의 전통적인 흐름공정문제에 대한 벤치마킹 문제들은 다수 존재한다. 하지만, 기계가용성제약을 고려한 흐름공정문제에 대한 벤치마킹 문제는 많이 존재하지 않는다. 그래서 Aggoune(2004)의 연구에서 사용한 벤치마킹 문제를 가지고 실험을 하고자 한다. 이 문제를 FSPAC (Flowshop Scheduling Problem with Availability Constraint) 문제라고 한다.

<표 2>은 문제유형을 표로 정리한 것이다. FSPAC 문제는 총 3개의 문제 유형(Problem Type)이 있다. 각각의

<표 2> Problem Type

	Job 수	기계수	작업시간	비가용구간의 크기	문제 수
Problem Type 1	10	5	U[50, 150]	$\frac{\Sigma 작업시간}{job수}$	각 Type 20 문제
problem Type 2	10	10			
problem Type 3	20	10			

문제유형별로 기계 *5*, *10*, *10*일 때, 작업의 수가 *10*, *10*, *20*개 존재하며, 각각의 기계마다 작업의 작업시간은 다르고, 그 값은 U[50, 150] 사이에 임의 생성방법으로 발생시킨다. 기계마다 비가용구간의 크기는 서로 다르며, 비가용구간의 크기는 각 기계에서 작업한 작업의 평균 작업시간으로 하였다. 예를 들어 기계 1의 작업 수는 5이고, 총 작업시간이 500이라면, 500/5로 100이 기계 비가용구간이 된다.

기계 비가용구간이 시작되는 시점은 다르므로 시작시점을 임의 생성방법으로 발생시킨다. 단, 각 기계마다 적어도 하나의 작업 후에 비가용구간이 발생하도록 한다. 이러한 문제 유형에 대해 각각의 문제 유형별로 총 20개의 문제를 생성하여 풀었다. 표로 정리하면 위의 <표 2>와 같다.

유전 알고리듬 모수로는 모집단의 수는 50개로 하였고, 교차율은 0.9이며, 돌연변이율은 0.05하여 실험을 하였다. 이것은 Aggoune(2004)에서 제시한 모수이며, 제안하고자 하는 알고리듬의 결과와 정확한 비교를 하기위해 동일하게 모수로 선택하여 실험하였다.

본 연구에서의 실험은 C++프로그래밍으로 알고리듬을 구현하였고, 실험한 컴퓨터는 PC Intel® 팬티엄 4.3GHz, DDR 512RAM을 사용하였다. 그리고 계산시간(CPU Time)의 단위는 초(Second)로 측정하였다. 다음 절에서는 이러한 문제의 유형에 따라 알고리듬을 실행하였을 때 나온 결과를 비교 분석하도록 하겠다.

<표 3>에서 %gap(Percentage gap)은 IA(제안하는 알고리듬)의 Aggune의 알고리듬 대비 향상도를 나타내는 것으로 다음의 식으로 표현된다.

$$\% \ gap = \frac{(RA - IA)}{RA} \times 100$$

<표 3> 실험결과

Problem Type 1				Problem Type 2				Problem Type 3			
문제	%gap	CPU Time (sec.)		문제	%gap	CPU Time (sec.)		문제	%gap	CPU Time (sec.)	
		RA	IA			RA	IA			RA	IA
1	26%	125	125	1	3%	345	345	1	11%	749	7501
2	16%	115	115	2	16%	656	656	2	14%	1371	1373
3	25%		89	3	13%	293	293	3	20%	779	780
4	18%	100	100	4	15%	345	345	4	19%	537	538
5	16%	84	84	5	5%	524	525	5	8%	567	568
6	29%	87	87	6	22%	268	269	6	19%	604	605
7	17%	98	99	7	21%	572	572	7	7%	5752	576
8	9%	90	90	8	14%	602	603	8	11%	560	560
9	15%	87	87	9	16%	417	417	9	11%	756	756
10	10%	97	97	10	12%	615	615	10	15%	510	510
11	20%	151	151	11	10%	529	530	11	17%	798	799
12	15%	94	94	12	18%	438	438	12	15%	752	752
13	21%	105	105	13	11%	268	268	13	15%	583	585
14	12%	86	87	14	10%	274	274	14	9%	617	617
15	20%	99	99	15	9%	371	372	15	23%	518	518
16	14%	124	124	16	14%	522	523	16	12%	531	532
17	13%	115	116	17	11%	575	576	17	7%	560	561
18	16%	93	94	18	15%	350	351	18	13%	779	780
19	16%	140	140	19	17%	395	396	19	15%	952	953
20	14%	76	76	20	6%	267	267	20	9%	895	897
Min	9%	76	76	Min	3%	267	267	Min	7%	510	510
Avg.	17%	103	103	Avg.	13%	431	432	Avg.	14%	700	700
Max	29%	151	151	Max	22%	656	656	Max	23%	1371	1373

여기서 RA는 Aggoune이 제안한 알고리듬을 적용해서 얻은 makespan의 결과이고 IA는 제안하는 알고리듬을 적용해서 얻은 makespan의 결과를 의미한다.

<표 3>에서 보는 바와같이 Aggoune(2004) 제안한 알고리듬의 결과와 제안하는 알고리듬의 결과를 비교하였을 때 거의 동일한 계산시간으로 평균적으로 Problem Type 1에서는 7%, Problem Type 2에서는 13%, Problem Type 3에서는 14% 향상됨을 보이고 있다.

실험을 통해 얻어진 결과에서 나타내는 바와같이 Aggoune (2004)의 알고리듬과 제안하는 알고리듬을 비교하였을 때 문제의 크기가 작은 문제에 대해 동일한 계산시간 안에 향상된 정도가 더 높음을 나타내었다. 또한 모든 문제 유형에 대해 제안하는 알고리듬이 평균적으로 15% 향상된 결과를 얻을 수 있었다.

5. 결 론

본 연구는 NP-hard 문제인 기계가용성 제약을 고려한 흐름공정상황 하에서 makespan을 최소화하는 문제로 본 연구에서 이 문제에 대해 알고리듬을 제안하였다.

기계가용성 제약을 고려한 문제는 한 기계에 한 구간 존재하는 것부터 m대의 기계에 두 구간 존재하는 경우까지 연구 되어왔다. 문제를 해결하는 방법으로 타부서치와 유전 알고리듬 등이 소개되어 왔다. 본 연구에서는 m대의 기계에 각각 기계 비가용구간이 두 구간 존재하는 문제를 해결하는 향상된 알고리듬을 제안하였다. 그 결과 기존에 Aggoune(2004)이 제안하였던 알고리듬보다 거의 동일한 계산시간을 가지고 평균적으로 15%향상된 결과를 얻을 수 있었다. 특히, 작은 크기의 문제에 대해서 더 나은 향상도를 보였다.

향후 연구과제로는 제안하는 알고리듬과 최적해를 비교하는 방법에 대한 연구가 필요하다. 또한 개별공정(Jobshop)과 같이 더 복잡한 생산환경에 대해 적용시킬 수 있을지 조사해 보는 것도 의미가 있을 것이다.

참고문헌

[1] 김여근, 윤복식, 이상복; "메타휴리스틱 : 유전 알고리듬・시뮬레이티드 어닐링・타부서치", 영지문화사, 1999.

[2] Adiri, I., Bruno, J., Frostig, E., and Rinnooy Kan, A. H. G.; "Single machine flow-time scheduling with a single breakdown," *Acta Informatica*, 26 : 679-696, 1989.

[3] Aggoune, R.; "Minimizing the makespan for the flow shop scheduling problem with availability constraints," *European Journal of Operational Research*, 153 : 534-543, 2004.

[4] Blazewicz, J., Breit, J., Formanowicz, P., Kubiak, W., Schmidt, G.; "Heuristic algorithms for the two-machine flowshop problem with limited machine availability," *Omega Journal*, 29 : 599-608, 2001.

[5] Blazewicz, J., Formanowicz, P., Kubiak, W., Przysuch, M. Schmidt, G.; "*Parallel branch and bound algorithms for the two-machine flow shop problem with limited machine availability*," Bulletin of the Polish Academy of Sciences, Technical Sciences, 2000.

[6] Gen, M. and Cheng, R.; *Genetic Algorithms and engineering design*, A Wiley Interscience Publication, 1997.

[7] Holland, J.; *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975.

[8] Kubiak, W., Blazewicz, J., Formanowicz, P., and Breit, J., Schmidt, G.; "Two-machine flow shops with limited machine availability," *European Journal of Operational Research*, 136 : 528-540., 2002

[9] Lee, C.-Y.; "Parallel machines scheduling with non-simultaneous machine available time," *Discrete Applied Mathematics*, 30 : 53-61, 1991.

[10] Lee, C.-Y., Liman, S. D.; "Capacitated two-parallel machines scheduling to minimize sum of job completion times," *Discrete Applied Mathematics*, 41 : 211-222, 1993.

[11] Lee, C.-Y.; "Machine scheduling with an availability constraint," *Journal of Global Optimization*, 9 : 395-416, 1995.

[12] Lee, C.-Y.; "Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint," *Operations Research Letters*, 20 : 129-139, 1997.

[13] Lee, C.-Y.; "Two-machine flowshop scheduling with availability constraints," *European Journal of Operational Research*, 114 : 420-429, 1999.

[14] Schmidt, G.; "Scheduling on semi-identical processors," *Zeitschrift fur Operations Research*, 28 : 153-162, 1984.