韓國經營科學會誌
第27卷 第2號
2002年 6月

동적 Job Shop 일정계획을 위한 유전 알고리즘*

박병주** · 최형림*** · 김현수*** · 이상완****

A Genetic Algorithm for Dynamic Job Shop Scheduling*

Byung Joo Park** · Hyung Rim Choi*** · Hyun Soo Kim*** · Sang Wan Lee****

■ Abstract ■

Manufacturing environments in the real world are subject to many sources of change and uncertainty, such as new job releases, job cancellations, a change in the processing time or start time of some operation. Thus, the realistic scheduling method should properly reflect these dynamic environment. Based on the release times of jobs, JSSP (Job Shop Scheduling Problem) can be classified as static and dynamic scheduling problem. In this research, we mainly consider the dynamic JSSP with continually arriving jobs. The goal of this research is to develop an efficient scheduling method based on GA (Genetic Algorithm) to address dynamic JSSP. we designed scheduling method based on SGA (Single Genetic Algorithm) and PGA (Parallel Genetic Algorithm). The scheduling method based on GA is extended to address dynamic JSSP. Then, This algorithms are tested for scheduling and rescheduling in dynamic JSSP. The results is compared with dispatching rule. In comparison to dispatching rule, the GA approach produces better scheduling performance.

Keyword : Dynamic Scheduling, Job Shop, Genetic Algorithm

논문접수일 : 2001년 11월 9일 논문게재확정일 : 2002년 5월 27일
* 이 논문은 2001년도 두뇌한국21사업에 의하여 지원되었음.
** 동아대학교 에이전트 기반 전자상거래팀(BK21) Post-Doc.
*** 동아대학교 경영정보과학부 교수
**** 동아대학교 기계산업시스템공학부 교수

1. 서 론

고객의 납기 요구를 만족시키려 하거나, 매우 큰 작업장의 작업순서를 결정하기 위한 일정계획문제는 경우의 수가 많고 매우 다양한 제약들을 포함하고 있다. 이외에도 새로운 작업의 시작, 작업의 취소, 기계의 고장, 납기일의 변경 등과 같은 많은 자원들의 변화와 불확실성을 가지고 있으며, 상충되는 서로 다른 목적들을 달성시켜야 하는 어렵고 복잡한 문제이다. 기존의 일정계획 기법은 실제 현장의 큰 규모 문제들을 해결하는데서 생기는 많은 문제점 외에도 다수의 제약들로 인하여 현실적인 일정계획을 이루지 못했다. 특히 실제 현장에서 빈번히 생길 수 있는 새로운 job의 시작, 취소 등과 같은 경우들을 제대로 반영하지 못하고 있다. 보다 현실적인 일정계획 기법이 되기 위해서는 현장에서 일어나는 이러한 동적인 변화들을 효율적으로 수용할 수 있어야 한다.

일반적으로 JSSP(Job Shop Scheduling Problem)는 job의 개시시간(release time)을 기준으로 해서, 정적(static) 또는 동적(dynamic) JSSP로 분류된다[3]. 정적 JSSP는 모든 job들이 0 시점에서 시작하기 위한 준비가 되어 있는 것으로 가정하고 있다. 이에 반해 동적 JSSP는 job의 개시시간이 특정 한 시점에 고정되어 있지 않다. 동적 JSSP에 관련된 연구는 정적 JSSP에 비해 적은 편이고, 그 또한 주로 시뮬레이션 방법과 우선순위 규칙(priority rule)을 이용한 방법론에 집중되어 있다. 동적 JSSP에 유전 알고리즘을 사용한 연구는 최근 Bierwirth[3]와 Fang 등[6, 7]에 의해 이루어졌다. Fang 등은 TSP(Traveling Salesman Problem)를 위해 고안된 간접 표현방법을 사용한 유전 알고리즘을 제안했다. 이 연구에서는 일부 공정들의 가공시간 변경이나 시작 시간의 변경에 의한 재 일정계획(rescheduling) 문제를 다룰 수 있는 두 개의 가능한 방법을 기술하고 있다. 하나는 변화된 새로운 자료로 일정계획 프로그램을 다시 실행하는 재 일정계획, 또 다른 하나는 새롭게 재 일정계획 하여 스케쥴을 얻을 충분한 시간이 없을 경우, 재 일정계획 전에 구해진 스케쥴 정보를 이용하여 일정계획 하는 방법인데, 이 방법은 새로운 재 일정계획으로 얻을 수 있는 최적 스케쥴의 발견을 방해 할 수 있다. Fang 등의 연구에서 사용된 간접 표현방법은 유전 알고리즘으로 얻은 이전 결과들의 사용을 어렵게 하기에 동적 환경에서는 적합하지 못하다. 왜냐하면 스케쥴의 보존과 스케쥴의 국소 부위 고정을 염색체 수준에서 행할 수가 없기 때문이다. Fang 등은 자신의 접근법을 확정적 동적 문제에 시험하였고, 우선순위 규칙보다 나은 결과를 얻었다. 그러나 새로운 job 개시, 기계 고장과 같은 확률적 사건을 포함하는 확률적 동적 문제는 다루지 않았다. 한편 Bierwirth 등[3]은 재 일정계획 시 새로운 job이 도착하기 전 유전 알고리즘의 마지막 집단에서 선택된 개체 정보를 활용하는 방법을 제시하였다. 단지 적은 수의 공정이 들어와 집단 내에서 정보의 작은 부분을 변화시켰기 때문에 이전 집단은 여전히 유용한 정보를 가지고 있다는 관점이다. 이전에 선택된 개체에 새로 들어온 job의 공정들을 추가해 가는 방법을 제안했으나, 실제 이러한 방법을 실행하지는 못하였다.

두 연구에서 제시된 방법들은 유전 알고리즘의 긴 수행시간 때문에 재 일정계획을 전체 최적을 통한 해의 개선이 아닌, 수행시간을 줄일 수 있는 방향으로 진행되고 있다. 동적인 환경에서는 수행시간이 중요하다. 스케쥴을 새로 구성하는데 걸리는 시간동안 작업이 계속 진행되기 때문에 빠른 시간에 새로운 스케쥴을 수립해야 할 필요가 있다. 그러나 해의 질(quality)도 중요하다. 그래서 본 연구에서는 이전 연구의 문제점을 보완하여 전체 최적을 이루기 위한 수행시간을 단축한 유전 알고리즘을 제시하고자 한다. 그러기 위해서 이전 유전 알고리즘에서의 보정(repair), 포싱(forcing) 과정들을 없애고 단순화한 유전알고리즘을 설계하였다. 여기서는 동적인 문제에 적합한 이해하기

쉽고 유연한 표현방법, 초기 모집단 구성 방법, 교차와 돌연변이 연산자가 좋은 해를 산출할 수 있도록 사용된다.

동(動)은 정(靜)의 연속이기에, 동적인 문제는 정적인 문제들의 연속이라 할 수 있다. 동적 일정계획에서 새로운 job의 시작이나 취소와 같은 예측하지 못했던 사건들을 고려해 일정계획 하기 위해서는, 사건이 발생 될 때 동적 문제는 정적 문제로 분해할 수 있다[13]. 그래서 동적 일정계획 문제는 정적 문제에서의 유전 알고리즘이 기반이 된다. 본 연구에서는 정적 문제에서 좋은 수행도를 보인 혼합 유전 알고리즘[1, 12]을 기반으로 동적 JSSP를 해결할 수 있는 일정계획 방법을 제시하고자 한다.

2. 동적 Job Shop 일정계획 문제

JSSP는 현존하는 많은 탐색기법들의 적용으로 다양한 응용이 가능하다는 이유와 문제의 어려움 때문에 아직도 활발한 연구 영역으로 남아 있다. 간략하게 JSSP는 작업의 선후관계와 자원제약을 지키면서 평가기준을 최적으로 달성하도록 자원들을 할당하는 문제로 정의되어진다. 또한 job의 개시시간에 기초해서 정적 또는 동적 일정계획 문제로 분류되어진다.

전통적인 정적 JSSP에는 여러 가정들이 있는데, 그 중 job의 수는 알려져 있고 고정되어 있다는 가정과 모든 job들은 시점 0에 시작하기 위해 준비되어 있다는 가정이 완화된 것이 동적 JSSP이다. 동적 JSSP에서 job의 개시시간은 한 시점에 고정되어 있지 않고, 각 job은 다양한 시간에 도착한다. 동적 JSSP는 job의 개시시간을 기준으로 해서 확정적(deterministic) 또는 확률적(stochastic, non-deterministic) 동적 JSSP로 분류되어진다[3]. 확정적 동적 JSSP는 새롭게 추가되는 job의 개시시간이 미리 알려져 있다고 가정한다. 그러나 확률적 동적 JSSP는 새롭게 추가되는 job의 개시시간은 알려져 있지 않다. 이는 일정계획 하는 0 시점에서 연속적으로 주문되는 새로운 job의 개시시간을 알지 못하는 경우이다. 그래서 시뮬레이션을 할 때 새로운 job의 개시시간은 알려진 확률분포에 의해 기술된 확률변수로 취급한다.

그러나 동적 문제를 확정적 동적과 확률적 동적 문제로 구분한 기존의 정의에는 오해의 소지가 있는 것 같다. 동적 일정계획 문제는 새로운 job의 주문과 같이 새로운 환경 변화에 따른 재 일정계획이 이루어지는 경우로 쉽게 생각 할 수 있다. 그러나 확정적 동적 문제의 경우는 전혀 동적인 변화를 반영하고 있지 않다. 일정계획 하는 시점도 job의 개시시간이 알려져 있기에 정적 일정계획과 같은 0 시점이다. 그래서 이를 확률적 동적 문제와 같이 동적 일정계획 문제로 분류를 하다보니 동적 일정계획 문제에 대해 혼동이 생기는 것 같다. 예를 들어 확률적 동적 문제를 불확실한 사건, 즉 새롭게 추가되는 job이 어떤 확률분포에 따라 들어오는지를 찾아 새로운 job의 시작시간을 추정하여 0 시점에 일정계획 하는 문제로 오해하기도 한다.

그래서 본 연구에서는 새롭게 이들을 정리하고자 한다. 기존 정의에서처럼 job의 개시시간을 기준으로 하지 않고, 일정계획이 이루어지는 시점을 기준으로 해서 0 시점에서 모든 일정계획이 이루어지면 정적 문제, 그렇지 않으면 동적 문제라 부르고자 한다. 확정적 동적 문제는 0 시점에 시작 할 수 없는 작업이 있지만 알려져 있기에 일정계획은 0 시점에서 이루어진다. 그래서 이를 확정적 정적 문제라 부르고, 확률적 동적 문제만을 동적 문제라 부르는 것이 좋을 것 같다. 이를 정리하면 정적 문제와 동적 문제는 재 일정계획 여부로 나누어지는데, 지속적인 재 일정계획이 이루어지면 동적 문제 그렇지 않으면 정적 문제라 한다. 그리고 정적 문제에서도 모든 작업이 0 시점에 시작 가능한 작업만을 다룰 경우를 정적 문제라 하고, 그렇지 않으면 확정적 정적 문제라 부른다.

3. 유전 알고리즘을 이용한 Job Shop 일정계획

3.1 정적 JSSP를 위한 유전 알고리즘

정적 JSSP를 위한 유전 알고리즘에서는 항상 해의 실행가능성을 유지할 수 있는 표현 방식과 G&T(Giffler & Thompson) 알고리즘[9]을 이용한 초기 모집단 생성, 새로운 교차와 돌연변이 연산자 그리고 선택방법을 사용한다. 이들 방법들의 수행도는 이미 입증되었다[1, 12]. 이들 방법들은 동적인 환경에서 개시시간이 늦은 job의 가공 순서를 급격하게 변화시키지 않고, 부모 세대의 유전 형질 즉 순서를 잘 물려받을 수 있도록 만들어졌다.

3.1.1 염색체 표현(Representation)

염색체 표현은 job 번호를 공정의 수만큼 반복시키는 순열형태로 표현한다[2]. 하나의 유전인자는 하나의 공정을 의미하고 표현형태는 처리되는 공정의 job 번호로 표현한다. 이처럼 job 번호의 반복을 통해 표현한 염색체는 job의 공정들이 스케줄되어지는 순서를 나타낸다. 예를 들어 3 (job) × 3 (기계) 문제가 순열 형태의 염색체 [3 2 2 1 1 2 3 1 3]로 표현되었다면, 세 번씩 반복된 숫자는 job의 번호를 나타낸다. job 번호가 세 번씩 반복되는 것은 각 job들이 3개의 공정을 가지고 있기 때문인데, job 번호의 첫 번째 반복은 그 job의 첫 공정을 두 번째 반복은 두 번째 공정을 의미한다. 이 염색체는 job 번호가 공정의 수만큼만 표시된다면 항상 실행가능성을 유지한다.

염색체의 생성은 G&T 알고리즘을 이용한다. G&T 알고리즘은 단계 4의 G 집합 내에 속해 있는 공정들을 하나씩 선택하는 과정을 전체 공정의 수만큼 수행하여 일정계획 하게 되는데, 이때 하나씩 선택되어지는 공정을 포함하는 job의 번호를 선택순서대로 전체 공정 수만큼 연결하여 염색체를 생성한다. 예를 들어 <표 1>과 같은 3개 job과 3대 기계를 가진 문제에서, G&T 알고리즘을 통한 염색체 생성과정은 다음과 같다.

〈표 1〉 3개 job, 3대 기계(3×3)를 가진 JSSP

Job	공정번호 (기계번호, 가공시간)
1	① (3, 1) → ② (1, 3) → ③ (2, 6)
2	④ (2, 8) → ⑤ (3, 5) → ⑥ (1, 4)
3	⑦ (3, 5) → ⑧ (2, 4) → ⑨ (1, 8)

G&T 알고리즘의 단계 1에서 집합 C는 공정 {①, ④, ⑦}이 된다. 단계 2에서 t(C) = 1로 산출되고, m*는 기계 3이 된다. 단계 3은 기계 3에서 시작시간이 t(C) 보다 빠르고 가장 작은 값을 가진 공정들을 집합 G로 둔다. G 집합은 공정 {①, ⑦}이 된다. 단계 4에서 하나의 공정을 임의대로 선택한다. 만약 공정 ⑦이 선택되었다면 그것을 기계 3에 일정계획 한다. 그리고 집합 C에서 그 공정을 삭제한다. 이 결과 첫 번째 염색체의 유전인자는 3번 job의 공정(⑦)이 선택되었기 때문에 3이 된다. 다음 집합 C는 공정 {①, ④, ⑧}로 수정된다. t(C) = 6이 되고 m*는 기계 3이 된다. 기계 3을 이용하는 공정은 ①밖에 없다. 그래서 집합 G는 {①}이 된다. 그래서 공정 ①을 기계 3에 일정계획 한다. 이 순서에 따라 두 번째 염색체의 유전인자는 1번 job의 공정(①)이 선택되었기 때문에 1이 된다. 이 과정을 9개 공정이 모두 선택될 때까지 반복하면 [3 1 2 1 3 2 3 1 2]라는 염색체가 만들어진다.

3.1.2 염색체 생성

(1) 기 호

n : job의 수
m : 기계의 수
$P_j(k)$: job j 의 기계 k 에서 가공시간 (j = 1, 2, ⋯, n, k = 1, 2, ⋯, m)
$r_j(k)$: job j 의 기계 k에서 시작시간

(2) 수정된 G&T 알고리즘 (active* 스케줄)

G&T 알고리즘의 단계 3을 아래와 같이 수정하고, 여기서 얻어지는 스케줄을 active* 스케줄이라 한다.

단계1) 각 job들 중에서 가장 먼저 일정계획 해야 할 공정들을 집합 C로 둔다. 집합 C 내의 모든 공정 j(k)에 대한 시작시간 $r_{j(k)}=0$ 이라 둔다.

단계2) $t(C)=\min_{(j,k)\in C}\{r_{j(k)}+P_{j(k)}\}$를 계산한다.

그리고 $t(C)$가 최소가 되는 기계를 m^*로 둔다.

단계3) 기계 m^* 상에서 $r_{j(m^*)}<t(C)$ 인 모든 공정 $j(m^*)$들 중에서 $r_{j(m^*)}$이 가장 작은 공정들을 집합 G로 둔다.

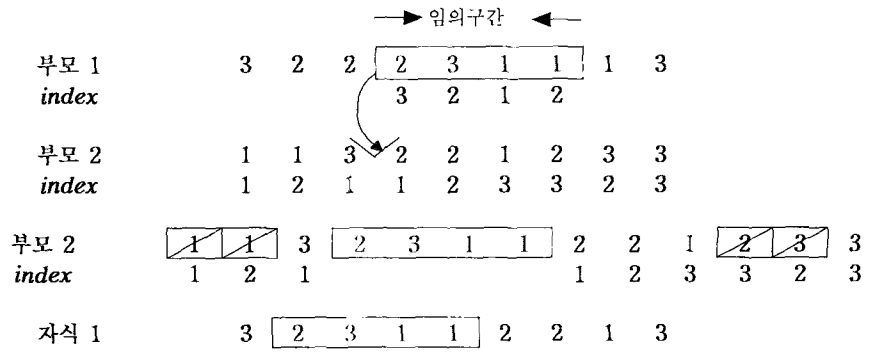
단계4) 집합 G에서 하나의 공정을 임의대로 선택하고 그것을 일정계획 한다.

단계5) C에서 그 공정을 삭제한다. 그리고 job에서 그 공정의 후행 공정을 집합 C에 포함시킨다. C에서 $r_j(k)$를 수정하고 모든 공정이 일정계획 될 때까지 단계 2)로 돌아간다.

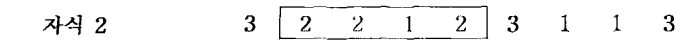
G&T 알고리즘으로 얻은 active 스케쥴과 non-delay 스케쥴을 초기 모집단으로 사용한 경우보다 수정된 G&T 알고리즘으로 얻은 active* 스케쥴로 초기 모집단을 구성한 경우 더욱 좋은 해를 산출할 수 있었다[12]. 이 active* 스케쥴은 수 없이 많은 active 스케쥴 중에서 축소된 active 스케쥴 집합에 속하는 실행가능한 스케쥴이다.

3.1.3 교차연산자(Crossover)

교차연산자는 G&T 알고리즘으로 얻은 염색체들이 좋은 스케쥴을 가지고 있기에 가능하면 그들 순서관계를 유지하면서 진화시켜 나갈 수 있는 연산자가 필요하다. SGA(Single Genetic Algorithm)에서 사용된 교차연산자는 먼저 임의 구간을 산출한 뒤 그 구간 내 부모 1의 유전인자들을 부모 2에 삽입한다. 삽입 위치는 임의 구간이 시작된 유전인자 바로 앞이다. 만약 첫 번째 부모에서 임의구간의 시작 위치가 4번째라면 삽입 위치는 부모 2의 4번째 유전인자 앞이 된다. 그리고 나서 임의 구간 내의 유전인자와 같은 인덱스(index)를 가진 유전인자들을 부모 2에서 삭제한다. [그림 1]에서처럼 부모 1의 임의 구간 내 유전인자 2가 부모 2에 삽입되고 난 후 부모 2에서 삭제되는 유전인자 2는 같은 인덱스 값 3을 가진 유전인자가 된다. 인덱스 값은 job 번호로 표현된 유전인자가 그 job의 몇 번째 공정인지를 나타낸다. 이들 과정을 부모 1과 2를 바꾸어서 수행하여 두 개의 자식 개체를 생성한다. 그리고 두 자식 개체 중 평가기준에 적합한



부모 1과 2를 바꾸어 같은 구간으로 두 번째 자식을 생성한다.



[그림 1] 교차연산자 과정

한 개체만을 다음 세대로 보낸다. 그 과정은 [그림 1]과 같다. 이 교차 연산자 외에 PGA(Parallel Genetic Algorithm)에서는 3개 교차연산자가 추가 사용된다[12].

PGA는 단일 집단을 여러 집단으로 나누어 독립적으로 진화시키면서 일부 개체의 교류를 통해 수렴현상을 줄여 보다 나은 해의 탐색을 가능하게 할 수 있다. 단일 집단을 여러 집단으로 나누어 서로 독립적인 유전 알고리즘으로 전개하고, 일부 개체가 한 집단에서 다른 집단으로 이주(migration)할 수 있는 PGA를 섬모델 PGA라 한다. 본 연구에서는 링형의 정적 연결 구조에 집단마다 다른 초기 모집단, 교차연산자, 선택방법, 파라미터를 사용하며, 이주가 동시에 이루어지는 섬모델 PGA를 사용하였다. 그리고 부 집단의 수를 2개와 4개로 하였다[12].

3.1.4 돌연변이(Mutation)

[그림 2]는 이웃탐색에 기반한 돌연변이(Neighborhood-Search-based Mutation) 연산[4]의 예를 보여준다. 순열 표현 형태의 염색체에서 이웃은 서로 다른 3개의 유전인자를 교환하여 얻을 수 있는 모든 염색체의 집합이 된다. 이 돌연변이 연산자는 case 2에서 case 6까지의 부모와 다른 5개 이웃해를 서로 비교하여 가장 좋은 것을 유전시키는 방법과 부모 염색체와 5개의 이웃해 모두를 비교해서 가장 좋은 것을 다음 세대로 전달하는 두 개의 형태로 사용한다. 전자의 경우는 일반적인 돌연변이 과정에서 사용하고 후자의 경우는 교차 연산자 수행 후 더 나은 개체 생성 가능성을 확인하기 위해 사용하는 돌연변이 연산자로 사용한다.

3.1.5 선택방법(Selection)

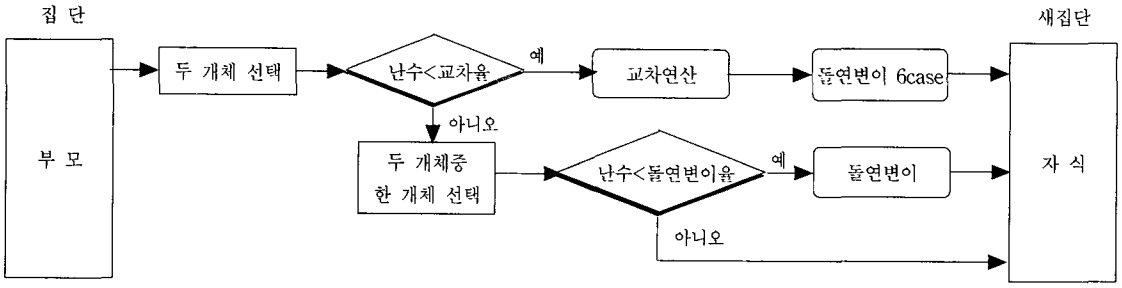
SGA에서 선택방법으로 씨종자 선택을 사용한다. 씨종자 선택은 일상에서 사용하는 개체 선택과 좋은 개체 보존 방법을 유전 알고리즘 진화과정에 도입한 것이다[1]. 가축을 사육하는 곳에서는 주로 개체 증식을 위해서 우수한 개체를 주로 씨종자로 사용하여 다음 세대를 구성해 나간다. 이 과정을 선택 방법에 적용하여 두 부모 중 부(父)에 해당하는 개체는 한 집단 내에서 정해진 순위 내에 드는 우수한 개체를 선택하고, 모(母)는 전체 집단 내에서 임의대로 선택한다. PGA에서 사용된 Goldberg & Deb[10]가 제시한 토너먼트 선택은 각 개체마다 선택확률을 부여하지 않는다는 점에서 씨종자 선택과 유사한 방법으로 병렬 시행에서 계산상 효과적이다.

3.1.6 교체(Replacement)

다음 세대의 구성은 현 세대에서 선택과 유전 연산자들을 이용하여 새롭게 구성한다. 새로운 개체들을 초기 모집단의 개수만큼 생성하여 다음 세대를 구성하고 난 뒤 엘리티즘을 적용하여 나쁜 개체는 엘리티즘 적용 개수만큼 좋은 개체로 다시 대체한다. 한 세대의 생성과정은 [그림 3]과 같다.

부모 염색체	1	2	3	1	2	3	1	2	3
	↓				↓				↓
이웃해 염색체									
case 1	1	2	3	1	2	3	1	2	3
case 2	2	2	3	1	1	3	1	2	3
case 3	2	2	3	1	3	3	1	2	1
case 4	3	2	3	1	2	3	1	2	1
case 5	3	2	3	1	1	3	1	2	2
case 6	1	2	3	1	3	3	1	2	2

[그림 2] 돌연변이 연산의 예



[그림 3] 세대의 생성 과정

3.2 동적 JSSP를 위한 유전 알고리즘

3.2.1 확정적 정적 JSSP에서 G&T 알고리즘

확정적 정적 JSSP에는 0 시점 보다 늦은 개시시간을 가진 job이 있다. 이 문제에서 초기 모집단을 산출하는 G&T 알고리즘은 작업 개시시간이 늦은 job들이 작업순서 상 뒤에 위치한 염색체를 생성해야 한다. job j의 작업 개시시간을 r_j라 하면, 확정적 정적 JSSP에서 다른 개시시간을 가진 job을 고려하여 초기 모집단을 생성할 수 있도록 G&T 알고리즘의 단계 1을 다음과 같이 수정한다.

단계 1)

집합 C를 각 job의 가장 먼저 일정계획 해야할 공정들의 집합으로 둔다. 집합 C에서 모든 공정 j(k)에 대해 $r_{j(k)} = r_j$ 로 둔다.

확정적 정적 JSSP를 다루기 위해 먼저 확정적 정적 JSSP에 맞게 수정된 G&T 알고리즘으로 개시시간이 다른 job을 고려한 초기 모집단을 생성한다. 그리고 이 모집단에 정적 JSSP를 위한 유전 알고리즘을 기반한 일정계획 기법을 적용한다.

G&T 알고리즘으로 모집단을 생성하는 과정에서 C 집합 내 공정들의 $r_{j(k)}$은 다음의 식으로 수정되어진다. 먼저 정적인 경우에 job j의 기계 k상에서 가공시간 $p_{j(k)}$를 가진 job j의 시작시간은 식 (1)과 같다.

$$r_{j(k)} = \max(r_{(j-1)(k)} + p_{(j-1)(k)}, r_{l(k)} + p_{l(k)}) \quad (1)$$

여기서 공정 $l(k)$은 기계 k상에서 공정 $j(k)$에 직전 선행하는 job l의 공정을 나타낸다. 그러나 job j의 작업 개시시간 $r_j > 0$인 것을 다루는 확정적 정적 문제는 job의 첫 공정에 대한 일정계획을 다루는 것이라 할 수 있다. 확정적 정적 문제에서 job j의 첫 번째 공정의 시작 시간은 식 (2)로 구할 수 있다.

$$r_{j(k)} = \max(r_j, r_{l(k)} + p_{l(k)}) \quad (1 \le j \le n,\ n\text{은 } job\text{의 수}) \quad (2)$$

3.2.2 동적 JSSP에서 G&T 알고리즘

동적 JSSP는 Raman 등[13]에 의해 제시된 것처럼 새로운 job 추가와 같은 사건 발생 시 확정적 정적 문제가 되고, 이들 문제의 연속으로 이루어진다. [그림 4]에서 문제 P_0는 시점 $t_0 = 0$에서 정적 JSSP로 둔다. t_0에서 일정계획 문제는 작업 개시시간 $r_i = 0$을 가진 n개의 job들로 이루어진다. $t_1 > 0$에서, 새로운 job의 시작과 같은 확률적 사건이 일어날 때, P_0는 정적 일정계획 기법에 의해 모든 공정들의 시작시간은 결정되어져 있다. 시점 t_1에서 새로운 job이 추가된다면, 새로운 문제 P_1이 만들어진다. P_1에서 출발시간 $t_{j(k)} < t_1$ 인 공정들을 P_0에서 제거한다. t_0에서 t_1까지의 기간 내에 시작되는 공정들을 가지는 P_0의 job j를 수정한다. 수정할 때 남아있는 공정이 없다면 job j를 완전히 제거한다. 그리고 재 일정계획 시 job이 완전히 제거

되지 않고 수정되어진다면 job j의 새로운 작업 개시시간을 다음과 같이 수정한다.

$$r_j = \max_{1 \le k \le m} (r_{j(k)} + P_{j(k)} \mid r_{j(k)} < t_1) \quad (3)$$

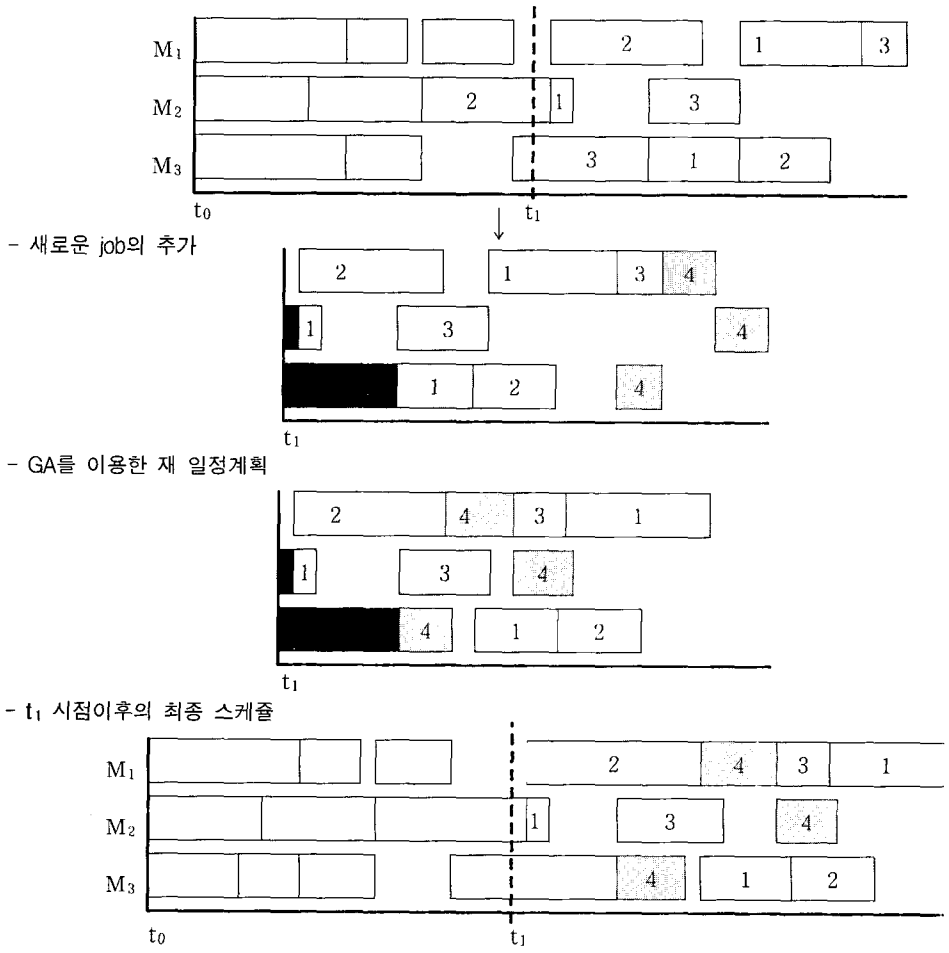
작업 개시시간의 수정에서 새로운 작업이 t_1에서 개시되기 전, 생산 중인 job j의 공정 중에서 t_1 시점에 가공중인 것이 있을 수 있다 ($r_{j(k)} < t_1 < r_{j(k)} + p_{j(k)}$). 이 상황에서 공정 $j(k)$를 가공하는 기계 k는 시점 t_1에서 활용이 불가능하다. 이 부분은 [그림 4]에서 검은 음영으로 표시되어있다. 이를 고려하여 모든 기계 k에 대한 작업 가능시간 a_i는 식 (4)와 같이 계산한다.

$$a_i = \max_{1 \le j \le n} ((r_{j(k)} + p_{j(k)} | r_{j(k)} < t_1),\ t_1) \quad (4)$$

새로운 문제에서 남아 있는 job들의 첫 공정의 개시시간은 식 (5)로 구할 수 있다.

$$r_{j(k)} = \max (r_j, a_i) \quad (5)$$

동적 JSSP에서 확정적 정적 문제는 새로운 job이 시스템에 들어올 때 또는 취소될 때마다 산출되어진다. 게다가 job의 개시시간은 기계의 작업 가능시간을 고려할 필요가 있다. 만약 기계를 시각



[그림 4] 동적 job shop 일정계획 과정

a_i에서 이용 가능하다면, 동적 문제에서 분해된 확정적 정적 문제를 위한 G&T 알고리즘의 단계 1은 다음과 같이 기계 작업 가능시간을 고려할 수 있도록 수정한다. 이 수정된 G&T 알고리즘을 적용하여 초기 모집단을 산출한다.

단계 1)

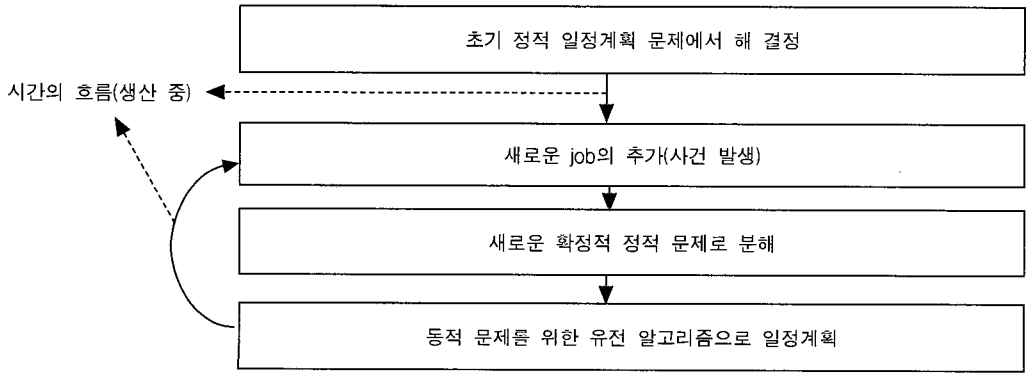
집합 C를 각 job의 가장 먼저 일정계획 해야할 공정들의 집합으로 둔다. 집합 C에서 기계 i에서 이루어지는 모든 공정 j(k)에 대해서 $r_{j(k)} = \max(r_j, a_i)$로 둔다.

새로운 job이 추가되면, 새로운 문제 P_1이 t_1에서 시작되어 새로운 확정적 정적 일정계획 문제 P_1을 가지게 된다. 이미 수행된 공정들은 삭제되고, 추가된 job의 공정들이 포함된 새로운 문제 P_1에서 동적 JSSP에 맞게 수정된 G&T 알고리즘으로 초기 모집단을 구성하고 각 작업의 개시시간과 기계들의 작업 가능시간을 고려해서 다음 job이 개시될 때까지의 일정을 유전 알고리즘을 이용해 [그림 4]에서처럼 재 일정계획 한다. 동적 일정계획은 이처럼 새로운 job이 개시되어질 때 새로운 문제를 산출하고 이들을 해결하고 난 뒤, 해를 실행하는 과정이 연속적으로 이루어진다. 그 과정은 [그림 5]와 같다.

한편 일정계획이 이루어지고 난 뒤 생산 중에 새로운 job이 추가된 경우, 이전 job에 대한 납기일을 고객과 합의하여 이전의 스케쥴을 유지해야 경우가 있다. 이 경우에는 이전 job의 납기일을 지키면서 새로운 job의 완성시간을 줄일 수 있도록 재 일정계획 해야 한다. 그러나 이전 job의 납기일에 여유가 없거나, 새로운 job의 수주 결정이 빠른 시간에 이루어져야 하는 경우는 전체 최적을 위한 재 일정계획이 의미가 없거나 불가능하다. 이런 경우는 Bierwirth[3]의 연구에서 제시되었던 것처럼, 새로운 job이 들어오기 전의 스케쥴을 유지하면서 새로운 스케쥴을 산출하는 방법으로 동적 일정계획 방법을 활용할 수 있다. 이전 스케쥴에서 수행된 공정을 삭제하고 추가된 job의 공정을 보탬으로 더 큰 스케쥴 문제를 형성하여 일정계획 하지 않고, 새롭게 들어온 job의 시작 가능시간을 이전 스케쥴의 각 기계에서 종료시점에 맞춰 기계 작업 가능시간으로 하여 이전 스케쥴을 유지하면서 일정계획 하면 훨씬 작은 크기의 문제로 빠른 시간 내에 일정계획 할 수 있다. 이 방법은 동적 일정계획을 훨씬 간편하게 해준다[5].

3.2.3 평가 함수(Performance Measure)

확정적 정적 문제와 동적 문제의 일정계획 과정은 초기 모집단의 생성과정에서 사용된 G&T 알고리즘과 평가 함수 값을 구하는 과정만 달리 할 뿐 정적 일정계획 기법과 동일하다. 즉, 초기 모집단이 생성되고 나면 정적 문제의 유전 알고리즘 과정과 동일하다. 단지 유전 알고리즘에서 각 개체에



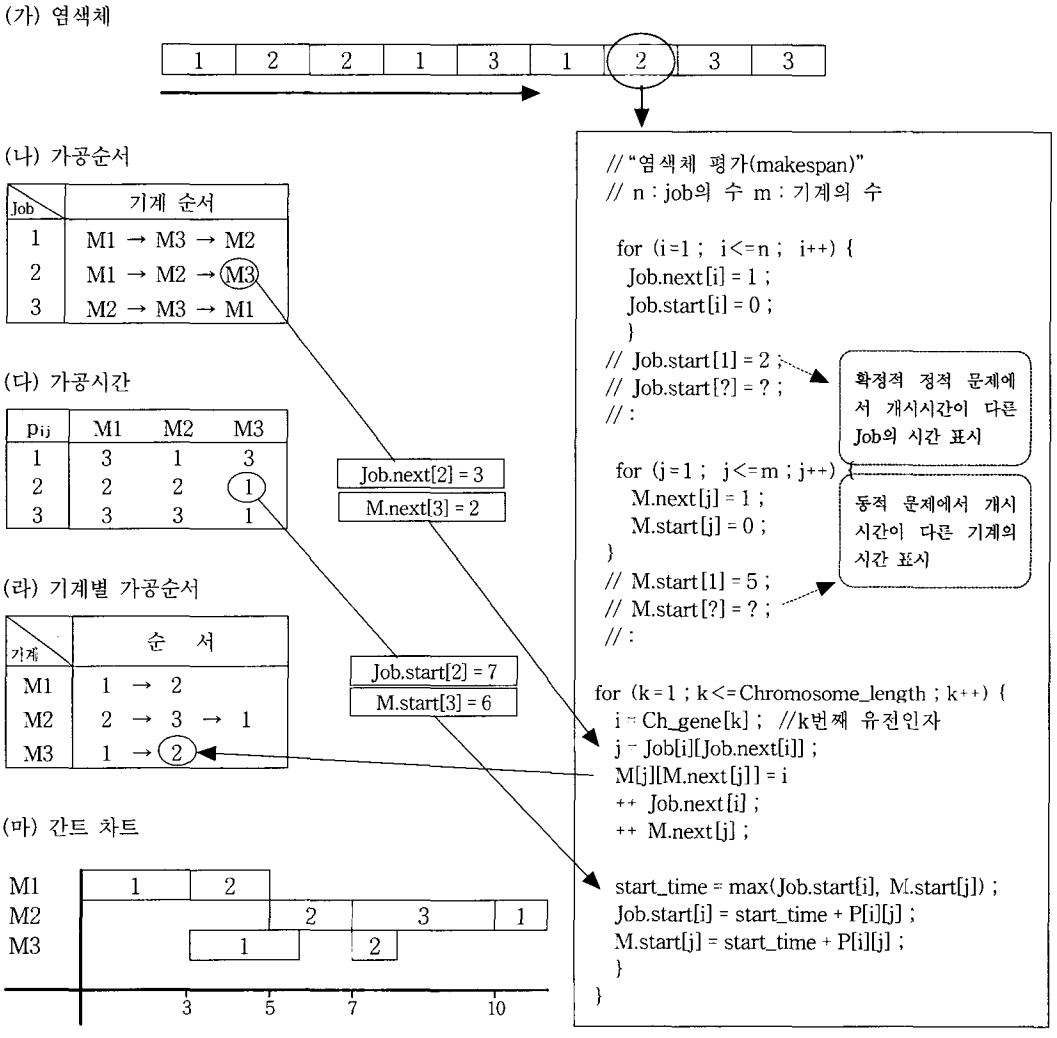
[그림 5] 동적 JSSP 일정계획 과정

대한 평가 함수로 사용된 Makespan 값을 구하는 과정이 각 job이나 각 기계의 시작 가능시간을 고려할 수 있도록 변형된다. [그림 6]에서 볼 수 있듯이 확정적 정적 문제와 동적 문제는 각 job의 개시시간과 기계사용 가능시간을 고려할 수 있도록 되어있다.

JSSP에서 최소의 makespan을 가진 스케줄은 종종 높은 기계 효율을 의미한다. 순열 형태의 염색체로 표현되었을 때 makespan은 왼쪽에서 오른쪽으로 유전인자를 읽어, job의 선후관계를 지키면서 기계에 할당하여 구한다. 확정적 정적 문제에서 makespan을 구하기 위해서는 각 job의 개시시간이 기록되어야 하고, 동적 문제에서는 각 job의 개시시간 뿐만 아니라 각 기계의 작업 가능 시간이 기록되어야 한다. 정적, 동적 문제에서 평가 함수 값을 구하는 과정은 [그림 6]과 같다.

4. 수행도 평가

실세계의 생산 환경은 새로운 job 개시, 기계 고



[그림 6] 유전 알고리즘에서 염색체 평가(Makespan을 구하는 과정)

장, job 취소, 납기일 변경 등과 같은 임의대로 발생되는 많은 변화가 일어난다. 실세계 일정계획 문제들은 동적인 특성 때문에 계산상 복잡한 NP-hard 문제로 알려져 있다[8]. 본 연구에서는 job이 연속적으로 도착하여, 새로운 job의 개시를 허용하는 동적 JSSP를 고려하였다. 실제 동적 JSSP에서 가장 자주 사용되어지는 방법은 우선순위 규칙(priority rule)이다. 기계를 이용할 수 있을 때, 현재 기계에서 이용 가능한 job들의 우선순위가 계산되어지고 높은 우선 순위 job들을 스케쥴 한다. 우선순위는 job, 공정 또는 기계들의 가공시간, 납기일, 개시시간, 기계 할당 등과 같은 파라미터들을 기초로 구해진다. 일반적으로 우선순위 규칙은 계산상으로 효과적이며 합당한 시간에 좋은 해를 찾아 준다는 점에서 유용하다. 하지만 해의 수준에 있어서는 만족스럽지 못하다. 따라서 본 연구에서는 유전 알고리즘을 기반으로 한 일정계획 기법을 통해 해의 수준을 높이고자 한다.

동적 JSSP에서는 shop 내에서의 흐름시간 감소 또는 makespan의 감소가 주요 목적이 된다. 유전 알고리즘에서 개체 선택을 위한 평가기준은 makespan을 사용하고, 동적 문제에서 실험의 최종 평가는 평균 흐름시간(Mean Flow Time : MFT)과 makespan을 사용한다. 이들 결과들을 우선순위 규칙, SPT(Shortest Processing Time first)와 비교하여 수행도를 평가한다.

동적 JSSP에서 최종 평가에 평균 흐름시간을 사용한 이유는 makespan으로는 마지막에 도입된 한 job이 마쳐지기 전에 잘 수행된 job들의 처리순서에 대해서는 평가할 수 없는 경우가 있기 때문이다. 즉 마지막 job이 큰 시간간격을 두고 들어와 이전 job들의 작업이 모두 수행된 경우는 makespan으로 평가가 어렵다는 얘기다. 그래서 모든 새로운 job들이 들어와 재 일정계획이 이루어지고 난 뒤의 최종 평가에는 평균 흐름시간을 포함시킨다. 평균 흐름시간은 job j의 완성시간 C_j와 개시시간 r_j를 고려하여 식 (6)으로 구한다.

$$\frac{1}{n}\sum_{j=1}^{n}(C_j - r_j) \qquad (6)$$

4.1 확정적 정적 JSSP에서의 수행도 평가

확정적 정적 JSSP에서 정적 일정계획 기법을 확장한 SGA와 PGA의 수행도를 평가하기 위해 4개의 문제를 가지고 실험하였다. 이 문제는 Fang 등[7]의 연구에서 사용된 실험 자료를 구할 수 없어, JSSP에서 가장 대표적인 MT10 문제[11]를 변형하여 구성하였다. 문제 1은 MT10 문제에서 job 3, 5가 100, 200 시점에 시작할 수 있는 경우, 문제 2는 job 3, 5, 7이 50, 150, 250 시점에 시작할 수 있는 경우, 문제 3은 job 3, 5, 7이 100, 150, 200 시점에 시작할 수 있는 경우, 문제 4는 job 1, 6, 8이 200, 350, 400 시점에 시작할 수 있는 경우로 하였다. 이 문제에 유전 알고리즘을 기반으로 한 확정적 정적 일정계획 기법을 적용한 뒤, G&T 알고리즘과 SPT 규칙의 결과와 비교하였다. 적용 결과는 <표 2>와 같다. 여기서는 PGA가 SGA 보다 더 나은 최선해와 평균값을 구하고 있으며, G&T 알고

〈표 2〉 확정적 정적 JSSP의 실험 결과

일정계획 방법	문제 1		문제 2		문제 3		문제 4		실행횟수
	Makespan (최선해)	평균	Makespan (최선해)	평균	Makespan (최선해)	평균	Makespan (최선해)	평균	
SGA	930	973.52	940	986.43	940	984.6	998	1012.01	50
PGA	930	965.58	940	976.6	937	972.96	998	1006.88	50
G&T	1046	1072.02	1040	1079.18	1030	1066.78	1048	1076.98	50
SPT	1445	1445	1570	1570	1375	1375	1429	1429	1

리즘과 우선순위 규칙으로 얻은 해에 비해서 훨씬 우수함을 확인할 수 있다. 여기서 평균값은 50회의 실행에서 얻은 Makespan 값의 평균이다.

모든 실험에서 교차나 돌연변이율, 엘리티즘 크기등의 유전 파라미터 값들은 많은 실험을 통해 얻은 값들을 사용하였다. 그리고 PGA는 4-PGA 중에서 4개의 부집단과 각 부집단에 사용하는 유전 연산자를 각각 달리한 모델을 적용하였다[12].

4.2 확률적 동적 JSSP에서의 수행도 평가

새로운 job의 추가로 분해된 확정적 정적 문제는 정적 일정계획 기법에서 확장된 동적 일정계획 기법으로 해결한다. 동적 일정계획 기법의 수행도를 평가하기 위해 확장된 SGA와 PGA 동적 일정계획 기법을 적용하여 수행도를 비교하고, 이들 결과를 우선순위 규칙, SPT와의 비교를 통해 제시한 일정계획 기법의 수행도를 평가한다.

새로운 job이 연속적으로 도착하는 동적 문제에서, 하나의 사건처럼 발생하는 새로운 job의 개시시간은 포아송 분포에 따라 발생시켰고, 평균 도착시간 간격은 50으로 하였다. 그리고 job의 공정 수는 10개로, 가공 순서는 임의대로 정하였으며, 가공 시간은 (1, 100) 사이에서 일양분포로 산출하였다. 그리고 새로 들어오는 job의 개수는 10개로 하여, 4번의 실험을 하였다. 0 시점에 시작되는 초기 정적 문제는 MT10 문제에서 최적해 값을 가지는 스케쥴로 하였다. 그리고 각 job이 추가에 따른 재일정계획에서 유전 알고리즘의 수행횟수(run)는 10회로 하였다. <표 3>은 4번의 실험에서 확장된 동적 일정계획 기법을 적용하여 얻은 결과이다. 이들 결과들은 우선순위 규칙과 비교한 결과에서 많은 개선을 이루어냄을 볼 수 있다. 이처럼 동적 JSSP로의 성공적인 확장은 정적 일정계획 기법들의 높은 수행도를 동적 JSSP에서도 기대할 수 있게 한다.

5. 결 론

제한된 자원 하에서 이루어지는 작업들의 일정계획은 생산관리의 효율성 측면에서 매우 중요한데, 기존의 일정계획 기법들은 다수의 제약들로 인해 현실적인 일정계획을 제공하지 못하고 있다. 특히 실제 현장에서 빈번히 생길 수 있는 새로운 job의 시작, 취소 등을 일정계획 문제에서 제대로 반영하지 못하고 있다. 그러나 보다 현실적인 일정계획 기법이 되기 위해서는 현장에서 일어나는 이러한 동적인 변화들을 일정계획 기법이 수용할 수 있어야 한다.

따라서 본 연구에서는 동적 JSSP를 위해 유전 알고리즘을 기반으로 한 일정계획 기법을 개발하였다. 이 목적을 달성하기 위해 유전 알고리즘에서 초기 모집단 구성 방법, 표현 방법, 유전연산자 등을 동적인 환경에 맞도록 설계하였다. 이 유전 알고리즘에 기반한 일정계획 기법은 표준 벤치마크 JSSP에서 좋은 결과들을 산출하였다. 이는 기법의 설계에서 문제 중심의 지식을 유전 알고리즘에 효과적으로 통합하였음과 유전 탐색을 강화시킬 수 있음을 보여준 것이다. 그리고 알고리즘 내에서 새로운 초기 모집단 구성 방법과 함께 보정 절차나 포싱 과정의 생략으로 동적 환경에서의 변화된 상

〈표 3〉 동적 JSSP의 실험 결과

일정계획 방법	실험 1		실험 2		실험 3		실험 4	
	Makespan	MFT	Makespan	MFT	Makespan	MFT	Makespan	MFT
PGA	1350	1033.00	1300	1052.35	1312	1014.30	1350	1020.80
SGA	1354	1088.20	1300	1083.60	1313	1027.40	1354	1052.40
SPT	2106	1196.30	1919	1156.85	1846	1043.25	2021	1170.85

황을 빠르게 반영할 수 있도록 하여 수행시간에서 많은 개선을 이루었다. 또한, 동적 JSSP 문제에서 PGA의 효과를 파악하였다. 적은 수행횟수로 이루어진 동적 문제에서 두 개의 평가기준 모두에서 가장 우월한 해를 산출해 내었다.

그리고 동적 JSSP를 기존의 방법들에 비해 쉽고, 효율적으로 다룰 수 있는 일정계획 기법을 제시함으로써 보다 현실적으로 현장에서의 일정계획 기법 설계에 유용하게 적용될 수 있을 것으로 생각된다. 비록 연구된 문제들이 실세계 일정계획 문제를 단순화 시켰을지라도, 인상적인 결과들은 실세계 일정계획 문제에 적용할 접근법의 기초로 일정계획 기법에 대한 믿음을 줄 수 있으리라 생각된다.

참 고 문 헌

[1] 박병주, 김현수, "Job Shop 일정계획을 위한 혼합 유전 알고리즘", 「한국경영과학회지」, 제26권, 제2호(2001), pp.59-68.

[2] Bierwirth, C., "A Generalized Permutation Approach to Job Shop Scheduling with Genetic Algorithms," OR-Spektrum, Special Issue : Applied Local Search, Pesch, E. and Vo, S.(eds), Vol.17, No.213(1995), pp.87-92.

[3] Bierwirth, C., H. Kopfer, D.C. Mattfeld, and I. Rixen, "Genetic Algorithm based Scheduling in a Dynamic Manufacturing Environment," IEEE Conf. on Evolutionary Computation, IEEE Press, (1995), pp.439-443.

[4] Cheng, R., A study on Genetic Algorithms-based Optimal Scheduling Techniques, Ph.D. thesis, Tokyo Institute of Technology, 1997.

[5] Choi, H.R., Kim H.S., Park, Y.J., Park, B.J. and A. Whinston, "An Agent for Selecting Optimal Order Set in EC Marketplace," Pacific Asian Conference on Information Systems 2001, June 20-22, Seoul, pp.490-508.

[6] Fang, H., "Genetic Algorithms in Time tabling and Scheduling," Ph.D. thesis. Department of Artificial Intelligence, University of Edinburgh, 1994.

[7] Fang, H., P. Ross and D. Corne, "A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling, and Open-Shop Scheduling Problems," Proc. Fifth Int'l Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, (1993), pp.375-382.

[8] Garey, M.R. and D.S. Johnson, Computers and Intractability : A Guide to the Theory of NP-Completeness, Freeman, San Francisco, CA, 1979.

[9] Giffler, J. and G.L. Thompson, "Algorithms for Solving Production Scheduling Problems," Operations Research, Vol.8(1960), pp.487-503.

[10] Goldberg, D.E. and K. Deb, "A Comparative Analysis of Selection Schemes used in Genetic Algorithms," In G. Rawlins, ed., Foundations of Genetic Algorithms, Morgan Kaufmann, 1991.

[11] Muth, J.F. and G.L. Thompson, Industrial Scheduling, Prentice-Hall, Englewood Cliffs, N.J., 1963.

[12] Park, B.J., Choi, H.R. and Kim, H.S., "A Hybrid Genetic Algorithms for Job Shop Scheduling Problems," Genetic and Evolutionary Computation Conference Late-Breaking Papers, 2001, July 7-11, E. Goodman, ed., ISGEC Press, San Francisco, pp. 317-324.

[13] Raman, N., R.V. Rachamadugu and F.B. Talbot, "Real-time Scheduling of an Automated Manufacturing center," European Journal of Operational Research, Vol.40 (1989), pp.222-242.