Job Shop 일정계획을 위한 semi active 스케쥴 기반 유전 알고리즘

김준우*, 하성호**

요약

Job Shop 일정계획 문제는 가장 잘 알려진 NP-hard 조합 최적화 문제 중의 하나로, 대수적인 방법으로 최적해를 구하는 것이 곤란하다. 이에 따라 유전 알고리즘이나 시뮬레이티드 어닐링, 타부 서치 등과 같은 확률적 탐색 기법들이 job shop 일정계획 문제 풀이에 많이 적용되었다. Job Shop 일정계획 문제 풀이를 위한 유전 알고리즘들은 일반적으로 active 스케쥴에 해당하는 해들로 구성된 세대를 유지하도록 설계되는데, 이는 최적해가 active 스케쥴에 포함되기 때문이다. 하지만 Giffler and Thompson 알고리즘과 같은 active 스케쥴 생성 방법은 종종 많은 계산을 요구한다는 점에서 효율적이지 못할 수 있다. 대신, 본 논문에서는 semi active 스케쥴에 기반한 유전 알고리즘인 sa-GA를 제안한다. sa-GA에서 해는 자연스러운 공정들의 나열로 나타내어지고, 이는 semi active 스케쥴로 손쉽게 전환된다. 또한, 보다 빠른 유전 연산이 가능하여 종래의 active 스케쥴 기반 GT/GA 알고리즘에 비해 효율적이다. 실험 결과, sa-GA 역시 해 집단에서 active 스케쥴에만 집중하여 최적해를 신속히 구할 수 있음을 볼 수 있었다.

Semi-active Schedule based Genetic Algorithm for Job Shop Scheduling

Jun-Woo Kim*, Sung-Ho Ha**

ABSTRACT

Job Shop scheduling problem is one of the most well-known NP-hard combinatorial optimization problems, and it is hard to obtain the optimal solution via numerical methods. Accordingly, probabilistic search methods such as genetic algorithm, simulated annealing and tabu search have been widely applied for solving Job Shop scheduling problems. In general, geneticalgorithms for Job Shop scheduling are designed to maintain a population consisted of active schedules, because the optimal schedules are included in the active ones. However, methods for generating active schedules such as Giffler and Thompson algorithm can be inefficient in that they are often computationally intensive. Instead, this paper proposes a semi active schedule based genetic algorithm called sa-GA. In sa-GA, a solution is represented as a natural permutation of operations, which is easily transformed into a semi active schedule. In addition, the genetic operations can be performed more quickly, and these aspects make sa-GA more efficient than the traditional active schedule based genetic algorithms. The experiment results show that sa-GA also concentrates on the active schedules in the population, and the optimal schedules can be obtained quickly.

Key Words : Job Shop Scheduling, Combinatorial Optimization, Genetic Algorithm, GT-GA, Meta Heuristic

* 동아대학교 산업경영공학과(✉kjunwoo@dau.ac.kr)
** 경북대학교 경영학부
· 제1저자(First Author) : 김준우 · 교신저자(Correspondent Author) : 하성호
· 접수일(2012년 8월 3일), 수정일(1차 : 2012년 10월 16일), 게재확정일(2012년 10월 20일)

Ⅰ. 서 론

제품 및 서비스 생산 현장에서 유한한 설비와 자원을 최대한 효율적으로 활용하여 납기와 고객 만족도 및 작업물 흐름 등을 향상시키기 위해서는 적절한 일정계획(scheduling)이 필요하다[1]. 생산 시스템의 형태 중, Job Shop은 다품종소량의 산출물을 다루는 데 적합하고, 고객의 요구가 다양화되어 감에 따라 점차 증가 추세에 있다[2].

Job Shop 시스템에서의 일정계획 문제(Job Shop scheduling problem, JSSP)는 전통적으로 다음과 같이 정의된다. 작업 현장에는 서로 다른 기계 M_1, M_2, ..., M_m이 존재하고, 처리해야 할 작업 J_1, J_2, ..., J_n이 현장에 도착해 있다. 하나의 작업은 m개의 공정(operation)을 순서대로 거쳐 완성되는데, 각 공정에는 일정한 시간이 소요되며, 동일 작업의 공정들은 서로 다른 기계에서 수행되기 때문에 한 작업은 각 기계들을 정확히 한 번씩 방문한다. 추가로, 각 기계는 한 번에 한 개씩의 공정만을 처리할 수 있으며, 공정들은 실행 중 중단되지 않는 것으로 가정한다. 일반적으로 JSSP 풀이의 목적은 총 처리시간(makespan)을 최소화하는 일정계획(schedule)을 작성하는 것이다[3][4].

JSSP는 기계의 대수 m과 작업의 개수 n이 증가함에 따라 가능한 일정계획 대안의 수가 폭발적으로 증가하는 NP-hard 문제로, 가장 어려운 조합 최적화 문제 중의 하나임이 잘 알려져 있다[5][6]. 이에, JSSP의 좋은 해를 현실적인 시간 내에 구하기 위해 해 공간을 효과적으로 탐색하는 확률적 탐색 기법들이 많이 활용되고 있다. 이러한 방법들은 흔히 메타 휴리스틱(meta heuristic)이라고 불리우며, 유전 알고리즘, 시뮬레이티드 어닐링, 타부 서치 등이 포함된다[4][7].

이 중, 유전 알고리즘[8]은 생물들의 진화 과정을 모방한 최적화 방법으로 단일 해가 아닌 복수 해로 구성된 세대(generation)를 유지하면서 이를 바람직한 방향으로 진화시켜 나가는 탐색 기법으로, JSSP에서도 상당한 성과를 거두어왔다[4][5][9].

유전 알고리즘에서 해 하나는 여러 유전자(gene)들로 이루어진 염색체(chromosome)로 표현되고, 이들은 생태계에서와 같이 선택, 교차 및 돌연변이 연산 등을 거치며 변형 및 개선되어 간다[7][8]. 전통적인 유전 알고리즘에서 염색체는 이진 속성의 나열로 표현되나, 이는 JSSP에는 적합지 못하다[6][7][10]. 따라서, Job Shop 일정계획의 특성을 반영한 여러 가지 염색체 표현 방법과 이들에 맞는 유전 연산자들이 개발되었다[3].

Job Shop 일정계획은 각 작업 내 공정들의 시작 시간과 완료 시간을 결정해줌으로서 생성할 수 있고, 이는 종종 Gantt 차트로 표현된다. 이 때 각 작업 내 공정들 간의 선후 관계 제약으로 각 기계들의 작업 일정에는 일반적으로 유휴 시간들이 존재한다. 긴 유휴 시간을 포함하는 일정계획은 그 유휴 시간 이후에 처리되는 공정을 앞당겨 처리하여 총 처리시간을 감소시킬 여지가 생긴다. 따라서 JSSP의 최적해는 이러한 앞당김이 더 이상 불가능한 'active 스케줄' 중에서 나온다[11].

Active 스케줄을 생성하는 방법으로 가장 잘 알려진 것은 GT(Giffler & Thompson) 알고리즘이다[12]. 따라서 JSSP를 위한 유전 알고리즘에서는 GT 알고리즘을 이용하여 active 스케줄의 세대를 유지하고자 하는 경우가 많았다[10][11][13]. 하지만 active 스케줄 생성 작업은 빈번한 해의 스캔 및 다양한 공정의 식별을 요구하여 알고리즘 수행 시간을 증가시키는 요인이 된다.

GT/GA[11]는 각 공정의 시작시간과 완료시간으로 해를 표현하고, active 스케줄을 유지하면서 부모 해들이 선호하는 공정을 할당하여 유전 연산을 수행하는 유전 알고리즘이다. 이 알고리즘은 유전 연산 중에 해들의 실행가능성(feasibility)과 적법성(legality)이 유지되는 장점이 있는 반면, GT알고리즘으로 인해 속도가 느려질 수 있는 단점이 있다.

이로 인해 해를 다양한 형태의 문자열로 나타내고 문자열의 절단과 접합, 재배열을 통해 교차(crossover)나 돌연변이(mutation)같은 유전 연산을 수행하는 알고리즘들이 제안되었다[3][7][14][15][16]. 하지만 이들은 종종 교차나 돌연변이 중에 해의 실행가능성이나 적법성이 위배되어 적절한 복구(reparing) 작업을 요구하는 경우가 많다.

본 논문에서는 기존의 GT/GA[11]를 변형하여, 동일 기계에서의 처리 순서를 바꾸어 앞당김이 가능한 공정이 존재하는 semi active 스케쥴들을 이용하는 sa-GA를 제안한다. 이 알고리즘은 종래 GT/GA와 같이 실행가능성과 적법성이 보장되는 해를 생성하는 유전 연산을 포함하고 있는 동시에, 자연스러운 공정의 나열로 해를 표현한다. 이러한 공정의 나열은 작업을 할당하는 순서를 나타내기 때문에 손쉽게 semi active 스케쥴로 전환이 가능하다. 또한, 진화를 거듭하면서 세대 내에 active 스케쥴의 비율이 빠른 속도로 증가하기 때문에 결국 최적해를 찾는 데도 우수한 성능을 보인다.

본 논문의 2장에서는 sa-GA 알고리즘의 연구 배경에 대해 간단히 소개하고, 3장에서 제안하는 제안하는 알고리즘의 세부적인 작동을 설명한다. 이어 4장에서 간단한 Job Shop 일정계획 문제에 이를 적용한 결과를 소개하며, 끝으로 5장에서 결론 및 추후 과제를 제시한다.

II. 연구 배경

유전 알고리즘은 대표적인 조합 최적화 문제인 Job Shop 일정계획 문제에 대한 해법으로 널리 연구되었으며, 최근까지도 알고리즘의 효율성을 개선하거나, 변형된 일정계획 문제를 풀이하기 위한 연구가 활발하다[4][9][17][18][19].

Job Shop 일정계획 문제를 관찰하기 위해 본 논문에서는 다음과 같은 표기를 사용하기로 한다.

o_{ij} $(i = 1, 2, ..., n, j = 1, 2, ..., m)$: i 작업의 j 번째 공정

t_{ij} : o_{ij}의 처리 시간

m_{ij} : o_{ij}를 처리하는 기계 번호

예를 들어, 3개 작업과 3개 기계로 이루어진 job shop 문제의 경우 <표 1>과 같은 정보가 주어진다. 이러한 Job Shop 일정계획 문제의 최적해는 어떤 공정도 앞당김이 불가능한 active 스케쥴에서 얻어진다는 점이 널리 알려져 있으며, active 스케쥴은 하나 이상의 공정이 동일 기계 내 선행 공정의 앞으로 앞당김이 가능한 semi active 스케쥴의 부분 집합이다[12].

표 1. 3×3 job shop 일정계획 문제
Table 1. 3×3 Job Shop Scheduling Problem

	작업 (job)	공정 (operation)		
		1	2	3
처리 시간 (t_{ij})	J_1	1	3	6
	J_2	8	5	4
	J_3	5	4	8
작업 기계 (m_{ij})	J_1	3	1	2
	J_2	2	3	1
	J_3	3	2	1

이러한 이유에서 종래 유전 알고리즘들은 active 스케쥴들로 구성된 해 집단을 진화시켜가면서 최적의 일정계획을 탐색하는 경우가 많았으며[10][11][13], active 스케쥴을 생성하는데는 GT알고리즘이 많이 사용되었다. GT알고리즘은 공정을 한 개씩 조사하며 각 공정의 시작 시간과 완료 시간을 결정하는 방법으로 (1)에서 주어진 절차를 통해 수행된다[12].

과정 1. 작업 내 미할당 선행 공정이 없는 공정들의 집합 Cut을 선별한다.

과정 2. Cut 에 속한 각 공정들의 가장 이른 완료 시간들을 조사하여, 이 시간이 가장 앞선 공정을 선택한다.

과정 3. 과정 2에서 선택한 공정 및 이와 같은 기계에서 처리되면서, 선택한 공정과 처리 시간이 겹치는 공정들의 집합 Conflict를 선별한다.

과정 4. Conflict 중 공정 하나를 임의로 선택하여 가장 이른 시작 시간과 완료 시간에 처리되도록 할당한다.

과정 5. 미할당 작업이 있는 경우, 과정 1에서부터 반복한다. (1)

이러한 GT 알고리즘은 초기 해 집단을 생성하는데 사용되기도 하고, 교차와 돌연변이 같은 유전 연산에서 사용되기도 한다. 유전 연산에 GT 알고리즘을 직접 사용하는 경우로는 GT/GA[11][14]가 있다. GT/GA에서 해는 각 공정의 시작시간 및 완료시간을 명시하여 표현하며, 교차와 돌연변이가 통합된 형태의 유전 연산자를 사용한다. 자녀 해를 생성할 때는 공정을 하나씩 특정 시간에 할당하는 것을 반복하는데, 다음 번에 할당할 공정을 GT알고리즘을 통해 결정하되, 과정 4에서 집합 Conflict 의 공정 중, 임의의 한 개를 선택하지 않고, Conflict에서 부모 해 중 한 쪽에서 가장 이른 시간에 시작되는 공정을 선택한다. 이러한 방법을 통해 부모 해의 형질을 효과적으로 이어받는 교차 연산의 개념을 실현하고, 돌연변이를 하는 경우에는 부모 해가 선택하지 않은 공정 중 하나를 Conflict에서 선택하여 부모 형질과 다른 특성을 부여한다.

이러한 GT/GA는 부모 해의 형질을 효과적으로 자녀 해에 유전시킬 수 있고, 유전 연산을 거쳐도 해의 실행가능성과 적법성이 훼손되지 않는다는 장점이 있다. 반면, (1)의 GT알고리즘의 수행 시, 해에 속한 공정들에 대한 빈번한 스캔 및 공정 선택이 필요하여 전체적으로 유전 알고리즘의 속도가 저하될 수 있다.

반면, 전통적인 유전 알고리즘에서와 같이 해를 문자열로 표현하고, 이들의 절단 및 접합, 재배열을 통해 교차와 돌연변이를 실행하는 방법들도 제안되었다[3]. 현재 가장 빈번하게 사용되는 표현 방법은 공정 기반 표현(operation based representation) 또는 반복 순열(permutation with repetition)이라 불리는 것으로, 작업을 반복적으로 나열하여 해를 표현한다. 예를 들어, <표 1>의 문제의 해는 <표 2>처럼 표현 가능하다.

표 2. 공정 기반 표현
Table 2. Operation-based Representation

C_1	322311132
C_2	333222111

<표 2>의 염색체 C_1, C_2는 모두 문자열로 표현되며, 각 문자는 하나의 공정에 해당하며 동일 숫자는 동일 작업 소속을 의미한다. 나아가, 동일 번호 등장 횟수에 따라 각 공정의 순서가 정해진다. 예를 들어, <표 2>의 C_1, C_2는 각각 <표 3>의 S_1, S_2와 같은 의미를 갖는다.

표 3. 공정의 나열
Table 3. Permutation of Operations

S_1	$o_{31}, o_{21}, o_{22}, o_{32}, o_{11}, o_{12}, o_{13}, o_{33}, o_{23}$
S_2	$o_{31}, o_{32}, o_{33}, o_{21}, o_{22}, o_{23}, o_{11}, o_{12}, o_{13}$

이러한 표현을 사용하는 유전 알고리즘들에서는 보통 교차와 돌연변이 연산이 서로 분리되어 있다. 교차 연산은 문자열의 절단과 접합을 통해 수행되며, 선행 관계 유지 교차(PPX), 일반화된 순서교차(GOX), 선행 관계 유지 순서 교차(POX) 등의 여러 가지가 존재한다[20][21][22][23]. 또한, 돌연변이 연산은 교차가 완료된 후, 생성된 자녀 해들의 문자열을 재배열하는 형식으로 수행된다.

<표 2>와 같은 문자열 표현을 사용할 경우, 문제점은 문자열로 암호화된 염색체들에 유전 연산을 가하

면서 해의 적법성이나 실행가능성이 훼손될 가능성이 있다는 점이다. 이로 인해 특별히 고안된 유전 연산자를 사용하거나, 유전 연산 이후 해를 복구하는 작업을 거치기도 한다. 이러한 과정은 종종 GT 알고리즘과 같이 시간을 요구할 수 있으며, 나아가 교차 연산과 분리된 돌연변이 연산이 부모 해에 없는 형질을 발생시켜 해의 다양성을 유지한다는 목표를 충분히 실현하는지에 대한 의문이 남는다.

또 하나의 문제는 이러한 문자열 표현은 각 기계에서 각 작업을 다루는 순서만을 명시하고 있어, 하나의 해 표현에 여러 개의 semi active 스케쥴 여러 개가 대응될 수 있다는 점이다[11]. 이로 인해 <표 2>와 같은 해에서 active 스케쥴을 얻기 위한 시간이 소요되는 경우도 많다.

III. sa-GA 알고리즘

3.1 해의 표현과 할당할 작업의 선택

본 논문은 GT알고리즘 없이 기존의 GT/GA가 가진 장점을 취할 수 있는 유전 알고리즘인 sa-GA를 제안하고자 한다. 해는 문자열로 표현되며, 구체적으로는 <표 3>처럼 공정의 나열을 사용한다.

공정의 나열은 각 공정에 대한 할당을 실시하는 순서로 해석하고, 한 개 공정 할당 시, 항상 기존에 할당 완료된 공정보다 뒤쪽에 배치한다. 즉, 공정의 나열이 주어진 경우, 맨 앞 공정부터 시작시간과 완료시간을 할당하며, 시작시간은 항상 작업 내 직전 선행 공정의 완료 시간과 동일 기계 내에서 직전에 할당 완료된 공정의 완료 시간 중 큰 것으로 설정한다. 완료시간은 이 시작시간에 해당 공정 작업시간을 더하여 결정할 수 있다. 모든 공정들을 할당하면 각 공정의 시작시간과 완료시간이 주어진 semi active 스케쥴을 얻는다.

한편, <표 3>과 같은 공정의 나열은 각 위치에 할당 가능한 작업들 중 한 개를 선택하는 것을 반복하여 얻어질 수 있다. 예를 들어, <표 3>의 S_1은 o_{31}, o_{21}, o_{22}, ... 로 시작하는데, 맨 앞 o_{31}은 처음에 할당 가능한 o_{11}, o_{21}, o_{31} 중 선택된 것이다. 두 번째 위치에서는 o_{31}이 할당되었으므로, 할당 가능 공정이 o_{11}, o_{21}, o_{32} 세 개이고, 이 중 o_{21}이 선택되었다. 세 번로 할당 가능한 공정은 o_{11}, o_{22}, o_{32}의 세 개이고, 이 중 o_{22}가 선택되었다. 이와 같은 할당을 9번째 위치까지 계속할 경우, 3×3 JSSP에 대한 공정의 나열을 얻는다.

3.2 부모 해에 의한 할당 공정 선택

새로운 해의 특정 위치에 공정을 할당하고자 할 때는 기존의 다른 해가 선호하는 것을 선택할 수도 있다. 이는 할당 가능한 작업들 중 기존 해 안에서의 가장 일찍 등장하는 것을 선택하는 것에 해당하며, 다음과 같은 과정을 통해 별도의 공정의 나열로 표현된 해를 통해서 실행 가능하다.

과정 1. 각 작업 i에 대하여 잔여 공정의 개수 u_i를 조사한다.

과정 2. u_i>0인 작업 i 각각에 대하여, 공정 o_{im-u_i+1}이 기존 해 S에서 등장하는 위치 pos_i를 조사한다. u_i=0인 경우 pos_i=$m \times n$으로 둔다.

과정 3. $pos_k = \min(pos_i)$인 작업 k를 찾아, 현재 위치에 공정 o_{km-u_k+1}를 할당한다. (2)

<표 1>의 문제에 대한 해를 만들면서 초기 세 개 위치에 o_{11}, o_{31}, o_{12}를 순서대로 할당하였다고 가정하자. 이 때 u_1=1, u_2=3, u_3=2이고, 네 번째 위치에 할당 가능 공정들은 o_{13}, o_{21}, o_{32}이다. 만약 표 3의 S_1을 기준으로 하는 경우에는 pos_1=7, pos_2=2, pos_3=4가 되어 작업 2가 다음 할당 공정을 제공, o_{21}이 네 번째

위치에 할당된다. 반면, S_2가 기준이라면 pos_1=9, pos_2=4, pos_3=2로, 작업 3의 o_{32}를 할당한다.

3.3 부모 선택 기반 교차 연산

sa-GA에서 유전 연산은 기존의 GT/GA에서처럼 부모 해들의 선호도를 이용하여 수행된다. 예를 들어, 1~9번째 위치에 각각 S_1, S_1, S_2, S_2, S_2, S_2, S_1, S_1, S_1에 의해 선택된 공정을 순차적으로 할당한다면, 첫 번째 위치의 경우, 모든 작업의 잔여 공정 개수가 3이므로 할당가능 공정이 o_{11}, o_{21}, o_{31}이고, 이들의 S_1 내 위치는 각각 5, 2, 1이므로 S_1은 o_{31}을 선택하여 첫 번째 위치에 할당한다.

두 번째 위치에 대한 작업 1, 2, 3의 잔여 공정 개수는 각각 3, 3, 2가 되어 할당가능 공정은 o_{11}, o_{21}, o_{32}이며, 이들의 S_1내 위치가 5, 2, 4이므로 S_1에 의해 o_{21}이 두 번째 위치에 할당된다.

이러한 할당 작업을 마지막 9번째 위치까지 반복하면 적법성과 실행 가능성을 갖춘 후손 해인 [o_{31},o_{21}, o_{32},o_{33},o_{22},o_{23},o_{11},o_{12},o_{13}]가 만들어지고, S_1, S_2의 역할을 바꾸어 추가적인 후손 해를 생성한다. 참고로 이렇게 생성한 문자열은 기존의 선행관계 유지 교차의 결과와 동일하나, 부모 해의 부분 문자열 추출을 위해 부모 해를 스캔하는 과정이 필요없고, 해의 표현이 단순하다는 점에서 효과적이다. 또한, semi active 스케쥴 생성에 이용된다는 차이가 있다.

제안하는 교차 연산은 (3)과 같은 절차로 표현할 수 있으며, 부모 해 P_1, P_2에 대하여 일점 교차, 이점 교차, 균등 교차를 정의하는 것이 가능하다.

과정 1. 초기화 작업
  set k=1 (k : 생성할 후손 해의 위치 번호)
  set u_i=m for all i

과정 2. k번째 위치 공정을 결정
  P_1에 의해 할당할 공정 정하는 경우
    P_1 기준으로 (2)에 의해 할당 공정 선택
  P_2에 의해 할당할 공정 정하는 경우
    P_2 기준으로 (2)에 의해 할당 공정 선택

과정 3. 반복 여부의 결정
  k=$n \times m$인 경우, 종료
  그렇지 않은 경우,
  set k=k+1
  GoTo 과정 2 (3)

일점 교차(one-point crossover) : 일정계획 내 임의의 p, $p+1$번째($1 \le p \le m \times n-1$) 위치 사이를 절단점으로 정하고, 1~p번째 위치에 P_1 기준으로 공정 할당, 나머지 위치에 P_2 기준으로 할당하여 후손 해 작성 후, P_1, P_2의 역할을 바꾸어 하나 더 작성한다.

이점 교차(two-point crossover) : 일정계획 내 두 개 절단점을 임의로 정하고, 가운데 부분에는 P_1 기준으로 할당, 나머지 양쪽 끝 부분에는 P_2 기준으로 할당하여 후손 해 작성 후, P_1, P_2의 역할을 바꾸어 하나 더 작성한다.

균등 교차(uniform crossover) : 일정계획 내 첫 번째 위치부터 순차적으로 0~1의 난수를 하나 발생시킨 후, 0.5이하이면 P_1기준, 그렇지 않으면 P_2를 기준으로 공정 할당하여 해 작성 후, 역할을 바꾸어 하나 더 작성한다.

3.4 부모 선택 기반 돌연변이 연산

유전 알고리즘에서 교차만을 사용할 경우, 국소 최적해로의 조기 수렴 등의 문제가 발생할 수 있다. 따라서 해의 다양성을 유지하며 탐색을 수행해나가기 위한 돌연변이 연산이 사용된다.

문자열로 표현된 해를 사용하는 경우, 돌연변이 연산은 교차를 완료한 후, 생성된 후손 해 안의 특정 공정들의 위치 교환, 특정 구간 공정들의 위치역순 등의 방법으로 수행된다.

<표 3>과 같은 공정의 나열로 표현된 해에 이러한 방법들을 적용할 경우에는 공정 간 선행관계를 위배하여 실행 불가능한 해가 생성될 위험이 있다. <표 2>와 같이 중복 순열 표현을 사용하면 이러한 문제점이 사라지나, 위치 교환이나 역순의 결과가 반드시 부모 해와 이질적일 것이라는 보장은 없다.

반면, 앞 절의 부모 선택 기반 교차를 하는 경우에는 특정 위치에서 부모 중 아무도 선호하지 않는 공정을 할당하여 종래 GT/GA에서와 유사한 돌연변이 연산이 가능하다. 예를 들어, S_1, S_2는 모두 첫 번째 공정으로 o_{31}을 선택한다. 따라서 교차 연산은 이를 첫 번째 위치에 할당하는 반면, 부모 해들이 선택하지 않은 할당가능 작업인 o_{11}, o_{21} 중 하나를 임의로 선택하여 할당하는 경우, 부모 해들과 다른 특성이 후손 해에 부여된다. 나아가, 이러한 돌연변이 연산은 교차 연산과 통합되어 수행될 수 있고, 돌연변이 연산을 포함할 경우, (3)의 과정 2는 (4)와 같이 수정된다.

과정 2'. k번째 위치 공정을 결정
  P_1에 의해 할당할 공정 정하는 경우
    P_1 기준으로 (2)에 의해 할당 공정 선택
  P_2에 의해 할당할 공정 정하는 경우
    P_2 기준으로 (2)에 의해 할당 공정 선택
  돌연변이를 적용하는 경우
    부모 해들이 선택 않은 공정 임의 선택 (4)

이같은 통합 유전 연산자는 돌연변이 연산의 개념을 잘 실현하면서 별도의 복구가 필요없다.

Ⅳ. 실험 결과

본 장에서는 sa-GA의 특성을 관찰하기 위해 표 1의 문제에 적용한 결과를 소개한다. 이 때, 한 세대는 20개의 해로 구성하고, 세대 교체 시에는 전통적인 확률바퀴 방식의 선택을 거치며, 교차율=0.5, 돌연변이율=0.01을 사용하였다. 참고로, 최적해를 구하는 것과 동시에 알고리즘 수행 시의 여러 가지 특성을 관찰하기 위하여, elitism은 적용하지 않았다. 아울러, 비교적 규모가 작은 JSSP임을 감안하여 초기 50회 진화를 통해 얻어진 결과를 관찰하고자 한다.

먼저, <그림 1>에서 볼 수 있듯이, sa-GA는 비교적 초기에 최적의 총 처리시간인 24일을 갖는 일정계획을 잘 찾아내는 것을 볼 수 있다.

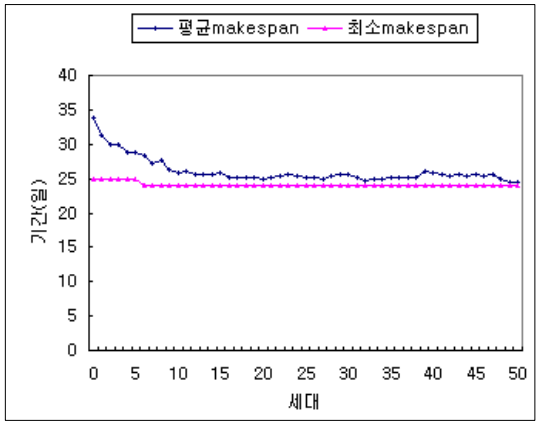

그림 1. sa-GA에 의해 탐색된 총 처리시간
Fig. 1. Makespan Obtained by sa-GA

이 최적해는 세대를 거듭하면서도 해 집단 안에 잘 보존되고 있고, 50회의 진화를 거치는 동안 평균 총 처리시간 역시 꾸준히 감소하여 안정화되는 모습을 보인다. 따라서, sa-GA 역시 해 집단을 개선해가며 최적해를 탐색하는 능력이 충분함을 알 수 있다.

한편, 50회의 진화를 거치면서 20개의 일정계획으로 구성된 해 집단 내에 포함된 active 스케쥴의 개수

를 관찰한 결과가 <그림 2>에 표시되어 있다. 초기 해 집단에는 active 스케쥴이 5개밖에 없고, 나머지는 모두 semi active 스케쥴이다.

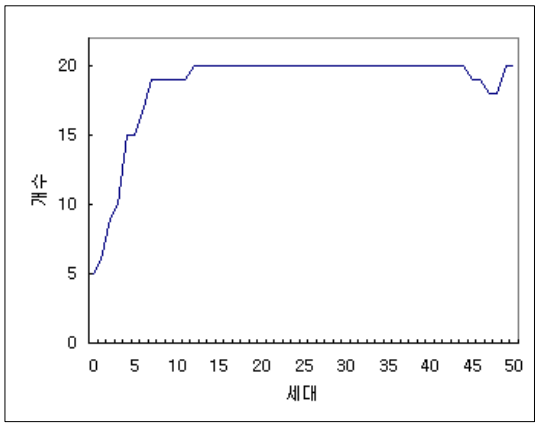

그림 2. Active 스케쥴 개수

Fig. 2. Number of the Active Schedules

그러나 각 공정들을 최대한 앞당겨 배치하는 active 스케쥴들이 일반적으로 보다 나은 총 처리시간을 가지므로, 세대 교체를 반복하면서 active 스케쥴이 빠르게 늘어나고, 12회 진화 이후부터는 해 집단 내 모든 일정계획이 active 스케쥴로 구성됨을 볼 수 있다. 즉, 인위적으로 active 스케쥴을 탐색하지 않고, 보다 빠르게 산출가능한 semi active 스케쥴을 이용하더라도 알고리즘을 수행하면서 active 스케쥴 위주의 탐사가 일어난다. 나아가, 때로 돌연변이에 의해 45~48회 진화에서처럼 다시 semi active 스케쥴이 등장하기도 하나, 곧 다시 active 스케쥴들이 그 자리를 대신함을 볼 수 있다.

끝으로, 동일한 횟수의 진화를 거치는데 sa-GA가 소요하는 시간을 전통적인 GT/GA가 소요하는 시간에 비교한 비율이 <그림 3>에 표시되어 있고, 인위적으로 active 스케쥴을 찾으려 하지 않는 sa-GA는 종래의 GT/GA에 비해 크게 적은 실행시간을 요구함을 볼 수 있다.

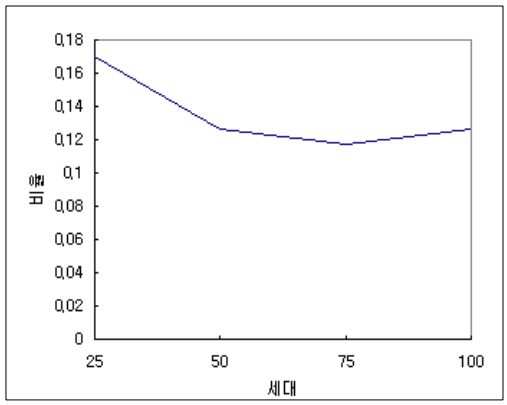

그림 3. 실행시간 비율

Fig. 3. Ratio of Execution Time

Ⅴ. 결론 및 추후 연구 과제

본 논문은 기존의 GT/GA 알고리즘을 개량하여 간단한 문자열로 나타낸 해를 semi active 스케쥴로 바꾸어가며 최적해를 탐색하는 sa-GA를 제안하였다. 이는 GT/GA처럼 JSSP의 본질적 특성을 이용한 유전 연산자를 갖고 있어 적법성이나 실행가능성을 유지할 수 있고, 교차나 돌연변이의 목표를 원활하게 실행한다.

sa-GA는 계산량이 많은 active 스케쥴 생성 과정을 거치지 않아 실행속도 면에서 크게 효율적이다. 또한, semi active 스케쥴을 이용하더라도 내부적으로는 active 스케쥴들에 초점을 맞추기 때문에 해 탐색 능력도 충분히 가지고 있다는 장점을 갖는다. 더불어, 기존의 공정 기반 표현 문자열을 사용하는 유전 알고리즘들과 달리 자연스러운 공정의 나열을 해의 표현으로 사용하기 때문에, 각 문자열이 어떤 공정을 의미하는지 해석하는 과정도 불필요하다.

본 논문의 저자들은 앞으로도 sa-GA의 특성을 보다 면밀하게 분석하여 알고리즘을 개선하는 작업을 지속하면서, JSSP를 비롯, 다양한 조합 최적화 문제에 적용해볼 계획이다.

참고문헌

[1] 이승정, 최회련, 이홍철, “Platform Development of Adaptive Production Planning to Improve Efficiency in Manufacturing System,” 한국산업정보학회논문지, 제16권, 제2호, pp.73-83, 1998.

[2] 배상윤, "The Information of Dispatching Rules for Improving Job Shop Performance," 산업경영시스템학회지, 제29권, 제4호, pp.107-112, 2006.

[3] Cheng, R., Gen, M., and Tsujimura, Y., "A Tutorial Survey of Job-shop Scheduling Problems using Genetic Algorithms - I. Representation," Computers and Industrial Engineering, Vol.30, No.4, pp.983-997, 1996.

[4] Qing-dao-er-ji, R., Wang, Y., and Si, X., "An Improved Genetic Algorithm for Job Shop Scheduling Problem," Proceedings of the 2010 International Conference on Computational Intelligence and Security, 2010.

[5] 정종백, 김정자, 주철민, “A Development of Hybrid Genetic Algorithms for Classical Job Shop Scheduling,” 대한산업공학회/한국경영과학회 2000 춘계공동학술대회 논문집, 2000.

[6] 박병주, 김현수, “A Hybrid Genetic Algorithm for Job Shop Scheduling,” 한국경영과학회지, 제26권, 제2호, pp.59-68, 2001.

[7] 김여근, 윤복식, 이상복, “Meta Heuristic,” 영지문화사, 1997.

[8] Holland, J.H., "Genetic Algorithm," Scientific American, Vol.266, pp.44-50, 1992.

[9] Li, Y., and Chen, Y., "A Genetic Algorithm for Job-Shop Scheduling," Journal of Software, Vol.5, No.3, 2010.

[10] 박병주, 최형림, 김현수, “A Genetic Algorithm-based Scheduling Method for Job Shop Scheduling Problem,” 경영과학, 제20권, 제1호, pp.51-64, 2003.

[11] Yamada, T., and Nakano, R., "A Genetic Algorithm Applicable to Large-scale Job-Shop Problems," Parallel Problem Solving from Nature, Vol.2, pp.281-290, 1992.

[12] Giffler, B., and Thompson, G.L., "Algorithms for Solving Production-Scheduling Problems," Operations Research, Vol.8, No.4, pp.487-503, 1960.

[13] Lee, H.P., and Sutinah, S., "A Modified Giffler and Thompson Genetic Algorithm on the Job Shop Scheduling Problem," MATEMATIKA, Vol.22, No.2, pp.91-107, 2006.

[14] Poon, P.W., and Carter, J.N., "Genetic Algorithm Crossover Operators for Ordering Applications," Computers and Operations Research, Vol.22, No.1, pp.135-147, 1995.

[15] Cheng, R., Gen, M., and Tsujimura, Y., "A Tutorial Survey of Job-shop Scheduling Problems using Genetic Algorithms, Part II: Hybrid Genetic Search Strategies," Computers and Industrial Engineering, Vol.36, No.2, pp.343-364, 1999.

[16] Abdelmaguid, T.F., "Representations in Genetic Algorithm for the Job Shop Scheduling Problem: A Computational Study," Journal of Software Engineering and Applications, Vol.3, No.12, pp.155-1162, 2010.

[17] Lestan, Z., Brezocnik, M., Buchmeister, B., Brezovnik, S., and Balic, J., "Solving the Job-shop Scheduling Problem with a Simple Genetic Algorithm," International Journal of Simulation Modelling, Vol.8, No.4, pp.197-205, 2009.

[18] Chen, Y., Hu, T.-T., Wu, G.-X., and Zhao, Z.-M., "The Improved Genetic Algorithm for the Complex Job-shop Scheduling," Proceedings of the 2010 IEEE 17th International Conference on Industrial Engineering and Engineering Management, pp.609-614, 2010.

[19] Zhang, G., Gao, L., and Shi, Y., "An Effective Genetic Algorithm for the Flexible Job-shop Scheduling Problem," Expert Systems with Applications, Vol.38, No.4, pp.3563-3573, 2011.

[20] Nakano, R., and Yamada, T., "Conventional Genetic Algorithms for Job Shop Problems," Proceedings of the 4th International Conference on Genetic Algorithms, pp.474-479, 1991.

[21] Bierwirth, C., "A Generalized Permutation Approach to Job Shop Scheduling with Genetic Algorithms," OR-Spektrum, Vol.17, No.213, pp.87-92, 1995.

[22] Lee, K.M., Yamakawa, T., and Lee, K.-M., "A Genetic Algorithm for General Machine Scheduling Problems," Proceedings of the 2nd International Conference on

Knowledge-Based Intelligent Electronic Systems, pp.60-66, 1998.

[23] 이혜리, 이건명, "A Comparative Study of Precedence-Preserving Genetic Operators in Sequential Ordering Problems and Job Shop Scheduling Problems," 퍼지 및 지능시스템학회 논문지, Vol.14, No.5, pp.563-570, 2004.

감사의 글

이 논문은 동아대학교 교내연구비 지원에 의하여 연구되었음.

저자소개

김준우(Jun-Woo Kim)

2001년 한국과학기술원 산업공학과 졸업(공학사)
2003년 한국과학기술원 산업공학과 졸업(공학석사)
2009년 한국과학기술원 산업 및 시스템공학과 졸업(공학박사)

2009년~2010년 한국기술교육대학교 산업경영학부 대우교수
2011년~현재 동아대학교 산업경영공학과 조교수
※ 관심분야 : 데이터마이닝, 지능형 시스템, 서비스관리, Operations Research, e-러닝, 융합 콘텐츠

하성호(Sung-Ho Ha)

1990년 연세대학교 경영학과 졸업(경영학사)
1998년 한국과학기술원 산업공학과 졸업(공학석사)
2001년 한국과학기술원 산업공학과 졸업(공학박사)

2001년~2002년 한국과학기술원 산업경영연구소 연구원
2002년~현재 경북대학교 경영학부 교수
※ 관심분야 : 데이터마이닝, e비즈니스, 지식경영, 의료경영, 지능의사결정