기 술 동 향

유전 알고리즘의 기초와 응용

김영석[1,#] · 샤오샤오[2]

1. 경북대학교 기계공학부 교수
2. 경북대학교 기계공학과 박사과정

Basic of Genetic Algorithm and Its Applications

Y. S. Kim, X. Xiao

1. School of Mechanical Engineering, Kyungpook National University
2. Graduate School of Mechanical Engineering, Kyungpook National University

1. 서 론

1960 년에 레젠버그(I. Resenberg)가 제안하고 1975 년에 홀랜드(J. Holland)와 그의 동료 데 정(De Jong)이 발전시킨 유전 알고리즘(genetic algorithm, GA)은 생물체가 환경에 적응하면서 진화(evolution)해 가는 모습을 모방하여 탄생한 확률적 전역 최적화 알고리즘(stochastic global optimization algorithm)이고, 인공신경망과 함께 가장 널리 알려진 알고리즘 중의 하나이다. [1-5]

수학에서 “제약조건하에서 어떤 함수의 값을 최대화 또는 최소화 하는 해를 구하는 알고리즘”을 일반적으로 최적화 알고리즘이라고 부른다. 여기서 다루어야 하는 함수를 목적함수라고 한다. 널리 알려져 있는 최적화 알고리즘으로는 선형계획법(linear programing, LP), 비선형계획법(nonlinear programing, NP), 동적계획법 (dynamic programing, DP) 그리고 진화적 알고리즘(evolutionary algorithm, EA) 등이 있다.

진화적 알고리즘은 다양한 문제에 대해서 경험적으로 솔루션을 제시하는 메타-휴리스틱(Meta Heuristic) 이라고도 불린다. 여기서 메타(meta)란 특정문제에 특화되어 있지 않고 광범위한 문제에 대응할 수 있는 것을 의미하고, 휴리스틱(heuristic, 경험적)이란 반드시 정확한 솔루션을 얻을 수 있는 것은 아니지만 비교적 단시간에 어느 정도까지 근사한 해를 얻을 수 있는 방법을 의미한다.

진화적 알고리즘에 속하는 유전 알고리즘은 이론적으로 전역 최적점을 찾을 수 있으며, 수학적으로 명확하게 정의되지 않은 다양한 문제 - 탐색 공간의 구조나 목적함수의 미분가능성을 필요로 하지 않아 선형계획법이나 비선형계획법가 적용되기 어려운 분야의 문제- 에도 적용할 수 있기 때문에 매우 유용하게 이용된다. 특히 이 유전 알고리즘은 항공공학에서 비행기 날개 공학설계, 자동차 네비게이션과 배송 경로문제(travelling salesman problem, TSP 순환 세일즈맨 문제), 생산공정 일정계획 문제(job shop scheduling problem), 개미 식민지 알고리즘(ant colony algorithm), 그리고 인간처럼 행동하는 학습로봇(learning robot) 제조, 최적 소성가공 공정의 설정 등등에 널리 이용되고 있다. [6-9]

2. 유전 알고리즘의 개요

Fig. 1 과 같이 유전 알고리즘에서 염색체 (chromo

some)란 생물학적으로는 유전 물질(정보)을 담고 있는 숫자열의 집합을 의미하며, 유전 알고리즘에서는 하나의 해 (solution)를 의미한다. 또한 유전자(gene)는 염색체를 구성하는 요소로써, 하나의 유전정보를 나타낸다. 만일 염색체가 [A B C]라면, 이 염색체에는 각각 A, B 그리고 C 의 값을 갖는 3 개의 유전자가 존재한다.

유전 알고리즘은 찰스 다윈(Charles Darwin; 1809-1882)이 제안한 생물들의 자연 선택(natural selection)에 따른 적자생존 법칙을 컴퓨터 알고리즘에 적용한 것이다. 여기서 자연 선택이란 특수한 환경 하에서 생존에 적합한 형질을 지닌 개체군이, 그 환경 하에서 생존에 부적합한 형질을 지닌 개체군에 비해 '생존'과 '번식'에서 이익을 본다는 이론으로 생물학적 진화 메커니즘(evolutionary mechanism)의 핵심이다. 이 유전 알고리즘으로 다양성과 복잡성이 존재하는 생물 세계의 유전원리를 완전히 모방할 수는 없지만 유전 알고리즘은 다음 세 가지 주요 특성을 따르도록 하였다.

첫 번째, 환경에 적합한 개체가 살아남는다.

두 번째, 살아남은 개체는 번식을 계속한다.

세 번째, 살아남은 개체는 각각 부모(parents)가 되어서 자손을 생성하는데, 자손(offspring)은 부모로부터 유전자를 받고, 때때로 유전자에 돌연변이가 발생하며 다음 세대(next generation)를 살아간다.

유전 알고리즘에서는 이 위의 세 가지를 반복적으로 수행하면서 환경에 적합한 우수한 개체들이 살아남는 솔루션을 제시한다.

유전 알고리즘은 대상으로 하는 문제를 유전자로 표현하고(genetic representation, 코딩이라고 부름) 하고 이들 유전자들이 연결된 고유의 염색체 세트를 갖는 모집단(population, 인구)을 정의하는 것으로부터 시작한다. 여기서 개인의 유전자 집합은 예를 들면 A1 [1,0,0,1,0,1], A2 [0,0,1,1,0,0]와 같이 0 과 1 의 이진 값의 문자열을 사용하거나 숫자 열 A1 [12,5,23,8], A2 [2,21,18,3], 문자열 A1 [a,b,c,d,a], A2 [a,c,d,b,a] 등으로 표현된다.

이 후 환경에의 유전자 적합성 (fitness), 유전자 재조합(recombination), 유전자 교차(crossover 또는 번식(breeding)) 및 유전자 돌연변이(mutation)를 기반으로 한 자연 선택을 통해 생물학적 진화 이론에서 영감을 받은 최적화 절차를 수행한다. 이 과정을 고정된 반복 횟수 동안 또는 주어진 반복 횟수 동안 최상의 솔루션에서 더 이상 개선이 나타나지 않을 때까지 반복한다. [10-11]

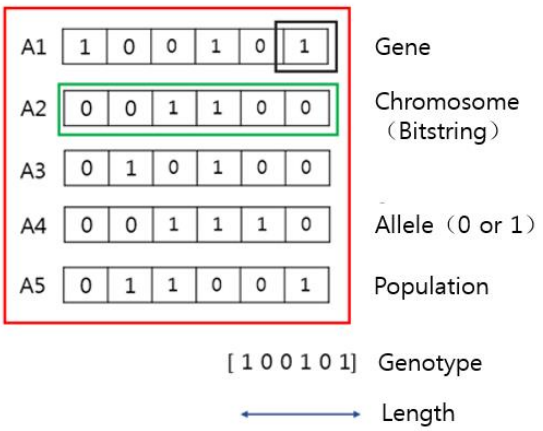


Fig. 1 The framework of the GA (population, chromosomes and genes)

GA 알고리즘에서는 일정한 확률로 돌연 변이를 만들어주는데, 이 돌연 변이들이 유전 알고리즘의 정체성을 나타내면서 알고리즘의 성능을 강화한다. 돌연변이가 필요한 이유는 솔루션 공간의 다른 부분을 탐색할 수 있는 새로운 경로를 도입하여 해가 지역 최적값에 빠져버리는 것을 방지하며 더 나은 솔루션을 찾을 수 있는 기회를 제공해주기 때문이다.

실제 GA 프로그램에서는 유전자 표현은 비트열(bitstring)로, 유전자 적합성(fitness)은 함수평가(function evaluation)로, 유전자 교차는 비트열 교차(crossover of bitstring)로, 돌연변이는 비트 뒤집기(flipping bits, inversion(역전)) 등으로 표현된다. 유전자의 적합성은 문제에 따라서 목적함수의 값, 이동거리, 실행시간, 구속조건 등등으로 평가할 수 있다.

다음 세대를 만들기 위하여 부모를 선택하는 방법으로는 적합성 비례 선택(fitness proportionate selection 일명 확률바퀴 선택(roulette wheel selection, 룰렛 휠 선택)), 토너먼트 선택(tournament selection) 그리고 엘리트주의 선택(elitist preserving selection), 순위선택(rank selection), 볼츠만 선택(boltzman selection) 등이 자주 이용된다. 적합성 비례 선택이란 모집단에 대해 각 개인의 적합성을 파악하여 적합성의 순서대로 선택 확률을 할당하는 방법이다. 또한 토너먼트 선택이란 모집단에서 정해진 수

의 개인을 무작위로 선택하고 그룹에서 가장 적합성이 높은 사람을 첫 번째 부모로 선택하는 방법이다. 여기서 적합성이란 각 개인이 다음 세대로 유전자를 전달하는 번식을 위해 얼마나 적합한지를 나타낸다. 개체가 번식을 위해 선택될 확률은 해당 개체의 적합성 점수(fitness score)를 기준으로 한다. 한편 엘리트주의 선택이란 가장 적합성이 높은 유전자를 반드시 다음 세대로 복사하는 방법이다.

부모는 다음 세대 후보를 생성하기 위한 기초이며 모집단의 각 위치에 대해 한 쌍의 부모가 필요하다. 선택되어 짝을 지은 부모(mating pool)는 자손(offspring)을 만드는 데 사용된다.

인간의 생식에서와 같이 성적 유전자의 재결합(recombination)은 교차 연산자(crossover operator)를 사용하여 수행된다. 여기에는 비트 문자열에서 임의의 교차 지점을 선택한 다음 첫 번째 부모에서 교차 지점(crossover point)까지, 그리고 교차 지점에서 두 번째 부모에서 문자열 끝까지의 비트 문자열을 서로 연결하여 자식을 만드는 것(세대 교체)이 포함된다. 교차가 일어나는 지점은 한 위치(1 점 교차), 두 위치(2 점 교차), 여러 위치(다점 교차), 균일 위치(균일 교차) 일 수 있으며 모든 유전자 위치에서 동일한 확률로 발생한다.

교차(교배, 짝짓기)는 모든 부모 쌍에 대해서 행해지는 것이 아니고 일정한 비율의 커플이 선택된다. 이것을 교차확률(cross probability) 이라고 한다. 교차 확률은 보통 80~90% 정도로 설정된다. 만일 교차확률이 100%라면 모든 자손은 교차로 만들어지며, 0%이면 부모의 염색체 복사본으로 완전히 새로운 자손이 만들어진다는 것을 의미한다.

또 하나의 현상으로 돌연변이가 있다. 돌연변이는 생성된 자식의 유전자가 무작위로 뒤바뀌는 것을 말한다. 이 돌연변이로는 기존 유전자 한 개를 빼고 그 위치에 다른 대립 유전자(對立遺傳子, allele 알리)를 삽입하는 것(insert), 무작위로 선택한 두 점 사이의 유전자의 순서를 변경하거나 다른 위치로 변경하는 것(change) 등이 있다. 단, 돌연변이를 수행할 때는 기존 유전자와 동일한 유전자가 삽입되지 않아야 하는 등의 조치가 필요하다. 일반적으로 돌연변이율(mutation rate 또는 돌연변이 확률(mutation probability))은$1/L$로 매우 작게(1% 정도) 설정된다. 여기서 L은 비트열의 길이이다.

예를 들어, 염색체의 길이가 20 비트의 비트열로 인코딩된 경우에 돌연변이 확률이 1%라면 랜덤하게 선택된 20 비트 중에서 1%의 비트가 뒤집힌다는 것을 의미한다. 이런 돌연변이는 현실의 생물 세계에서 지금까지 존재하지 않았던 유전자로 변이하여 유전적 다양성을 높이는 데 기여한다. (Fig. 2)

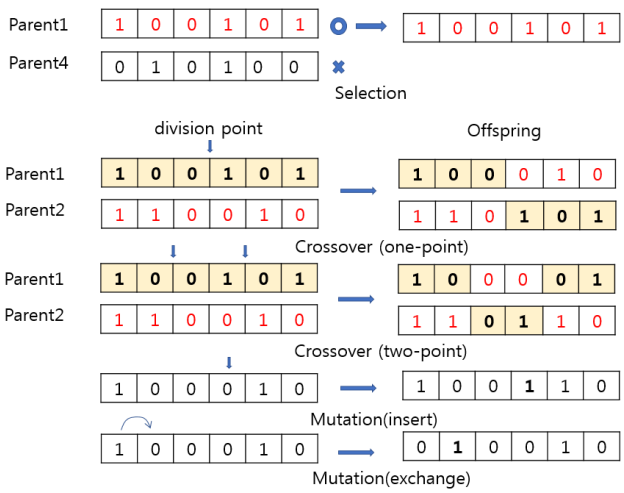


Fig. 2 Selection, crossover, mutation in GA

3. 유전 알고리즘 구조

초기의 모집단은 항상 고정된 크기를 갖는다. 새로운 세대가 형성되면 적합성이 가장 낮은 개체가 소멸되고 새로운 자손을 위한 공간이 제공된다. 즉, 이 유전 알고리즘은 현재 존재하는 염색체들의 집합으로부터 적합성이 가장 좋은 염색체를 선택하고, 선택된 해의 방향으로 이전 세대보다 더 나은 해(개인, 염색체)를 찾아가기 위해서 반복된다. [12]

이상의 유전 알고리즘의 메커니즘을 단계별로 표현하면 아래와 같다:

(1) 초기 염색체의 모집단을 랜덤하게 생성하고 교차율과 돌연변이율을 지정.

(2) 초기 염색체들에 대한 적합성 계산. (확률과 누적확률 계산)

(3) 적합성을 기준으로 짝지을 염색체를 선택하는 연산. (선택을 위한 난수 생성과 확률바퀴 선택 등 방법을 이용한 염색체 선택)

(4) 선택된 부모 염색체들로부터 자손들을 생성. (교차 연산자를 이용한 자손의 생성)

(5) 돌연변이 연산.

(6) 생성된 자손들의 적합성 계산. (좋은 염색체를 유지)

(7) 종료 조건 판별. (반복 계산에서 이전 세대와 크게 다른 자손을 생성하지 않아 모집단이 수렴되었다고 판별된 경우, 일반적으로 N 회 반복했을 때 ","개체의 적응도가 설정한 값을 넘었을 때 계산을 종료함)

(8-1) 종료 조건이 거짓인 경우, 3)으로 이동하여 반복.

(8-2) 종료 조건이 참인 경우, 유전 알고리즘을 종료. (최종 세대 중에서 가장 적합성이 우수한 유전체가 해가 됨)

Fig. 3 에 유전알고리즘의 흐름도를 나타내었다.

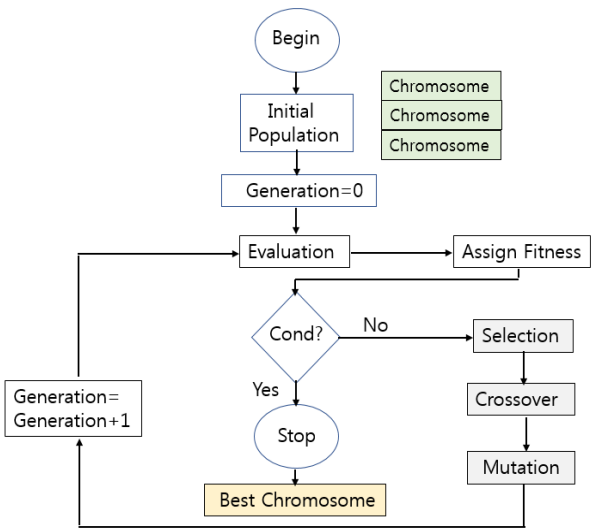


Fig. 3 Flowchart of GA

유전자 알고리즘의 장점으로는 다음과 같은 점을 들 수 있다.

(1) 너무 우수한 집단은 세대를 거치면 부진해질 수 있지만 적당히 우수하고 모자란 집단은 가혹한 도태가 필요하지만 오히려 좋은 결과를 가져올 수 있다.

(2) 복수 개의 개체 사이의 상호 협력(선택, 교차, 돌연변이 등)에 의한 해의 탐색을 통해 좋은 해를 발견하기 쉽다.

(3) 유전자 알고리즘에서는 뉴럴 네트워크의 역전파 알고리즘 등에서 요구되는 번거로운 미분 연산 등이 필요하지 않으며 알고리즘이 단순하고 평가 함수가 불연속인 경우에도 적용이 가능하다.

한편 단점으로는 다음과 같은 문제점이 있다.

(1) 최적의 솔루션에 매우 가까운 두 개체이면 교차에 의해 최적해를 얻기 힘든 경우가 있다.

(2) 대상으로 하는 문제를 유전자 알고리즘으로 해결하기 위한 일반적인 방법이 없다. 특히 문제를 유전자형으로 표현(코딩)하는 것은 전적으로 설계자의 경험에 의존한다.

(3) 개체수, 선택 방법이나 교차 방법의 결정, 돌연변이의 비율 등 파라미터의 수가 많다.

4. 유전 알고리즘 적용 사례

4.1 방문 판매원 문제

이하에 방문 판매원 문제를 이용하여 유전 알고리즘이 어떻게 작동하는지를 설명한다. 방문 판매원 문제는 방문 판매원이 판매를 위해 여러 도시를 방문해야 한다고 가정할 때 여행에 소요되는 비용(유클리드 거리(Euclidean distances)에 따른 비용)을 최소화하는 최단 경로를 찾는 문제이다. [13]

가장 확실한 해결책은 다양한 가능성을 고려하고 각각에 대한 예상 거리를 계산하고 최단 경로를 선택하는 무차별 대입 방법이다. 이것이 TSP 를 해결하는 확실한 방법이지만 이 접근 방식에서는 TSP 의 도시 수가 n이면 가능한 경로 수는 $(n-1)!$로 많기 때문에 비현실적이다. 대안으로 유전 알고리즘을 고려할 수 있다.

TSP 의 판매원은 모든 도시를 방문한 후에 원래의 출발지 목적지로 돌아와야 한다는 전재조건이 있다. 즉, 판매원이 도시 i에서 시작하여 도시 i로 돌아올 때까지 나머지 모든 도시를 방문해야 한다. 또한 도시 i에서 도시 j로 가는 거리가 도시 j 에서 도시 i로 가는 것과 같다고 가정한다. 이런 유형의 문제를 대칭 방문판매 문제(Symmetric Traveling Salesman Problem) 라고 한다.

이 TSP 모델 정식화를 위한 주요 변수들과 구속조건은 다음과 같다.

(1) $i,j \in Capitals$: 판매원이 방문할 도시의 집합;

(2) $Pairings = \{(i,j) \in Capitals * Capitals\}$: 허용된 짝짓기의 집합;

(3) $S \subset Capitals$: 도시 집합의 하위 집합;

(4) $d_{i,j} \in R^+$: 모든 (i,j) 짝에 대해서 수도 i에서 수도 j까지의 거리;

(5) 수도 i에서 수도 j까지의 거와 동일하다 (즉, $d_{i,j} = d_{j,i}$):

$$d_{i,j} = d(x_i, y_i), (x_j, y_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (1)$$

(6) $x_{i,j} \in \{0,1\}$: 도시 i와 도시 j가 연결되었으면 이 변수는 1, 그렇지 않으면 0이다;

(7)최단 경로(경로의 총 거리를 최소화):

$$Min\ Z = \sum_{(i,j)\in Pairings} d_{i,j} * x_{i,j} \quad (2)$$

(8)제약조건:

$$\begin{cases} x_{i,j} = x_{j,i} & \forall (i,j) \in Pairings \\ \sum_{(i,j)\in Pairings} x_{i,j} = 2 & \forall i \in Capitals \\ \sum_{(i\neq j)\in S} x_{i,j} \leq |S| - 1 & \forall S \in Capitals \end{cases} \quad (3)$$

제약조건의 첫 번째는 대칭 구속 조건, 두 번째는 한 도시가 두 도시를 연결하는 조건조건, 세 번째는 하위 집합의 모든 도시를 방문하고 원해 도시로 돌아가는 경로는 없다는 조건을 나타낸다.

아래 Fig. 4 와 같이 5 개의 도시를 방문하는 방문 판매원 문제에 대해서 설명한다. 문제를 간단히 하기 위하여 방문 판매원은 A 도시에서 출발하고 A 도시로 돌아온다고 가정한다. 각각의 도시간의 거리를 Table 1 에 나타내었다.

A 도시에서 출발하여 모든 도시를 방문한 후에 A 도시로 돌아오는 경로는 Fig. 5 와 같이 다양하게 선택할 수 있다. 각각의 경우에 두 도시 간의 이동한 거리를 합하면 최종 이동거리를 계산할 수 있고 가장 적은 이동거리를 나타내는 경로를 찾을 수 있다. 이 경우와 같이 방문해야 하는 도시 수가 5 개로 적은 경우에는 최적 이동경로를 $(5-1)!$ =24 회 계산으로 쉽게 찾을 수 있다. 이 경우에 최적 경로는 0->4->1->2->3->0 경로 또는 이 경로의 역방향 경로인 0->3->2->1->4-0 경로로 이동거리는 128km 가 된다. 하지만 방문해야 하는 도시 수가 20 개 정도로 많아지면 엄청난 횟수의 계산이 필요하다.

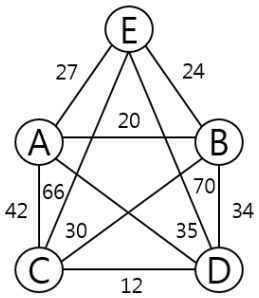


Fig. 4 Simple TSP model

Table 1 Distance between each city [km]

		0	1	2	3	4
		A	B	C	D	E
0	A	-	20	42	35	27
1	B	20	-	30	34	24
2	C	42	30	-	12	66
3	D	35	34	12	-	73
4	E	27	24	66	73	-

Route 1 : 0 1 2 3 4 0 : distance =162
Route 2: 0 3 4 1 2 0 : distance =204
Route 3: 0 1 3 2 4 0 : distance =159
Route 4: 0 4 1 2 3 0 : distance =128 (minimum)
Route 5: 0 1 2 4 3 0 : distance =228
Route 6: 0 2 1 4 3 0 : distance =204(=Route 2)
Route 7:
Route 8: 1 0 3 4 2 1 : distance =206
Route 9: 1 0 3 2 4 1 : distance =157
Route 10:

Fig. 5 Total distance of each route

A 도시에서 출발하여 모든 도시를 방문한 후에 A 도시도 돌아오는 방문 판매원 문제를 파이썬에 의해 유전 알고리즘으로 나타낸 것이 부록 1 의 프로그램이다. [14-16] 실제의 경우에는 모든 도시에서 출발하는 경우를 고려해야 하고 후보가 되는 여러 부모 유전체 중에서 가장 우수한 유전체의 선택, 두 개의 유전자의 교차로부터 우수한 자손 생성 그리고 돌연변위를 반복적으로 수행하여 이동거리가 가장 적은 최적 경로를 찾아야 한다. 하

지만 여기서는 유전자의 선택, 유전자의 교차와 돌연변위가 어떻게 파이썬에서 구현되는지를 설명하기 위해 반복계산은 하지 않았다. 독자들은 #print 의 #을 제거하여 실행시켜 보면 반복 계산이 어떻게 이루어지고 있는지 알 수 있을 것이다.

4.2 점진판재성형 문제

앞에서 설명한 유전 알고리즘을 점진 판재성형 (Incremental sheet forming, ISF) 공정에서 최적 성형조건을 찾는 문제에 적용해 보자. 점진 판재성형이란 기존의 프레스 가공에서와 같은 고가의 금형을 제작하지 않고 소형의 구형공구로 판재를 점진적으로 장출 소성변형 시켜 가는 방법으로서 빠른 시제품 제작과 소량 생산에 적합한 성형방법이다. [17-18] 이 점진 판재성형의 공정변수로 공구 직경 (tool diameter, mm), 공구 회전속도(spindle speed, rpm), 매 스텝당 z-방향 깊이(step depth, mm), 공구 이송속도(feed rate, mm/min), 등은 성형제품의 품질에 크게 영향을 미친다. [19]

이하에 두께가 1.0mm 인 Al3004 의 원뿔절두체 (varying wall angle conical frustum, VWACF) 모델의 점진 판재성형에서 최대 성형각도를 얻기 위한 최적 변수의 조합을 유전 알고리즘을 통해 구하는 과정을 나타내었다. (Fig. 6)

점진 판재성형에서 최적 성형각도를 예측하기 위한 목적함수는 다음과 같다: [20]

$$f(angle) = -[85.696 - 0.39a + 0.009b - 0.48c - 0.07d + 0.316a^2 + 0.213b^2 - 1.24c^2 - 0.394d^2 - 0.893ab - 0.033ac - 0.069ad + 0.449bc - 0.053bd - 0.121cd] \quad (4)$$

여기서 a 는 공구 직경, b 는 공구 회전속도, c 는 매 스텝당 z-방향 깊이, d 는 공구 이송속도를 나타낸다.

또한 각 변수들의 제약 조건은 다음과 같다.

$$\begin{cases} 6mm \le a \le 10mm \\ 60rpm \le b \le 180rpm \\ 0.2mm \le c \le 0.6mm \\ 400mm/min \le d \le 1200mm/min \end{cases} \quad (5)$$

각 변수들의 제약 조건으로 모집단 크기가 100, 교배율이 0.8, 돌연변위 확률이 0.007, 그리고 확률적 균일 선택방법을 이용하였다.

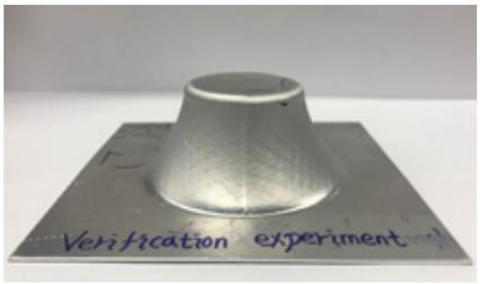


Fig. 6 ISF of VWACF model

유전 알고리즘에서는 일반적으로 가능한 충분히 큰 임의의 비트열과 많은 수의 모집단을 사용하기를 추천한다. 그러나 여기서는 간단히 각 변수들을 5 비트열로 나타내었고 6 개의 부모만을 가정하였다. 따라서 각 부모는 20 개의 유전자를 랜덤하게 갖는 비트열로 표현할 수 있고 앞에서부터 5 개씩이 각각의 변수에 대응하는 값이다.

모집단 각각 비트열의 목적함수에 대한 적합성을 평가하기 위해서는 비트열을 먼저 정수값으로 디코딩해야 한다. 다음에 디코딩한 정수를 원하는 범위로 환산하여 이를 목적 함수에 적용한다. 구체적인 예를 들어 설명해보자.

Fig. 7 과 같이 부모 1 의 경우에 첫 번째 5 개의 비트열 [1, 1, 0, 1, 0]이 나타내는 정수값(integer)은$1 * 24 + 1 * 23 + 1 * 21$=26 이므로 실제의 값(value)는 6+(26/32)*(10-6)= 9.25mm 를 나타낸다. 또한 부모 1 의 경우에 네 번째 5 개의 비트열 [0, 1, 1, 0, 0]은 $1 * 22 + 1 * 22$ =12 이므로 400+(12/32)*(1200-400)=700mm/min 을 나타낸다는 것을 알 수 있다. 이와 같은 방법으로 부모 1 의 비트열을 디코딩하여 정수값을 구하면 [26, 11, 3, 12]가 되고 실제 값은 [9.25 101.25 0.2375 700] 가 된다. 이 값을 목적함수에 대입하고 마찬가지로 모든 부모들에 대해서 구하여 적합성을 평가하여 순위를 정한다. 이 들 중에서 가장 성형 깊이가 (성형각도에 대응) 깊은 두 부모를 선택하여 유전자의 비트열에 대해서 유전 알고리즘의 교차, 돌연변이 등의 조작을 통하여 가장 우수한 개체를 선택하는 과정을 반복하면 최적 해가 얻어진다.

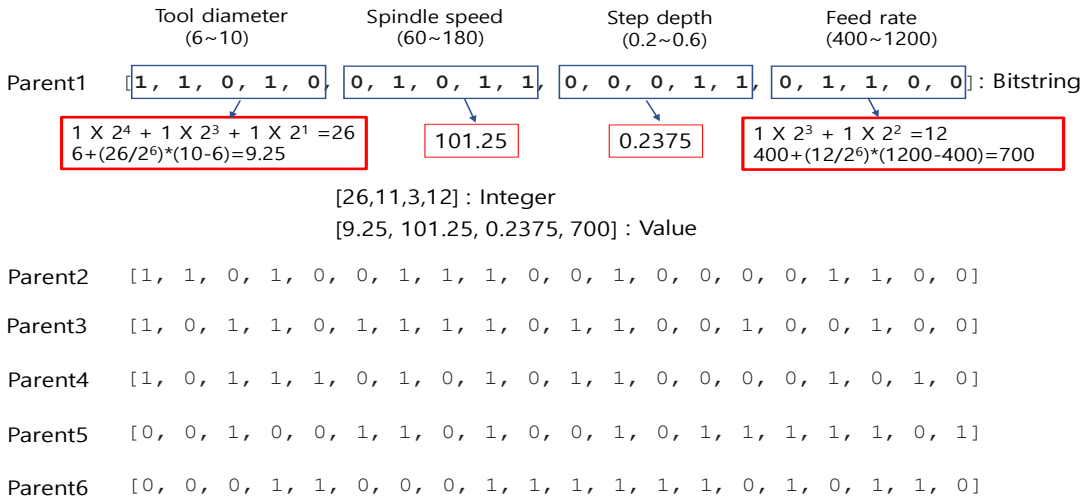


Fig. 7 Calculation of each parent

부록 2 에 나타낸 실제 프로그램을 이용하여 Table 2 에 나타낸 GA 에서 구해진 최적 변수들의 조합은 공구 직경 9.9mm, 회전속도 179.5rpm, Z-방향 피치 0.2, 이송속도 400mm/min 이고 이들 최적 변수조합으로부터 최대 성형각도 86.82° 를 얻었다. 이 조건에서 실제 점진 판재성형 실험결과는 87.071° 이었다. 즉, GA 에 의해 최적화된 예측값과 비교하면 1% 미만의 오차를 갖는다는 것을 알 수 있다.

Table 2 Best combination of parameters and max angle

Parameter	Optimum value
a: Tool diameter (mm)	9.9
b: Spindle speed(rpm)	179.5
c: Step depth(mm)	0.2
d: Feed rate(mm/min)	400
Max angle (°)	86.82

4.3 작업 스케줄 문제

여기서는 다양한 기계에 의해서 다양한 작업이 이루어지고 있는 제조 공장에서 어떻게 효율적으로 작업 스케줄을 정해야 하는지에 대한 작업 스케줄 문제(Job Shop Scheduling Problem, JSSP) 에 유전 알고리즘을 적용한 사례를 소개한다. [21-22] 여기서 개별생산 (Job Shop) 이란 다수의 범용 기계들이 존재하여 다양한 제품을 소량 생산할 수 있는 유연생산 시스템 또는 소규모 공장을 말한다. 반면에 흐름생산 (Floor Shop)은 자동차와 같이 하나의 제품 군에 대해서 충분한 수요가 있는 경우에 지배적인 제품흐름을 용이하게 하도록 배열된 대량생산 시스템 또는 대규모 공장을 말한다.

대상으로 한 소규모 공장에서 4 개의 기계를 (Machine) 이용하여 3 개의 작업(Job)을 수행하여 3 종류의 제품을 생산하고 있는 것을 가정하자. 이 3 작업 x 4 기계 (3-Jobs x 4-Machines) 문제에서 각 작업에 소요되는 기계 순번과 작업에 요구되는 소요시간을 (J,M)의 쌍으로 다음 Table 3 과 Table 4 에 나타내었다. Table 3 은 작업을 기준으로, Table 4 는 기계를 중심으로 나타낸 표이다. 여기서 J는 작업을, M은 기계를 나타낸다.

여기서 $J1$은 작업 1(Job1)을 나타내고, $M2$는 기계2 등을 나타낸다. $J1$의 $(M2,5)$는 기계 $M2$를 이용하여 작업 $J1$을 수행하는데 5 시간(flow time, 작업흐름시간)이 소요된다는 것을 의미한다. 따라서 작업 $J0$(Job0)는 기계 $M0$을 3 시간, 기계 $M1$을 3 시간, 기계 $M2$를 4 시간, 기계 $M3$을 1 시간 사용하여 총 11 시간에 작업이 완료된다는 것을 의미한다. 여기서 작업흐름 시간이란 실제 가공에 소요되는 시간, 기계와 기계 사이에 반제품 이송시간, 가공할 제품을 기계에 셋팅하는 시간 등을 모두 포함한 것이다.

Table 3 Flow time required based on job

	Pair(Job, Machine working time)			
$J0$	($M0$,3)	($M1$,3)	($M2$,4)	($M3$,1)
$J1$	($M3$,1)	($M0$,2)	($M2$,5)	($M1$,2)
$J2$	($M1$,1)	($M3$,3)	($M2$,4)	($M0$,2)

Table 4 Flow time required based on machine

	$J0$	$J1$	$J2$
$M0$	3	2	2
$M1$	3	2	1
$M2$	4	5	4
$M3$	1	1	3

이 공장에서 3 개의 제품이 $J0$, $J1$, $J2$ 공정에서 순서대로 가공되며, 현재 사용하고 있지 않고 공회전 또는 정지 상태(idle time)에 있는 기계가 있으면 사용할 수 있는 것으로 하였다. 이 문제에 유전 알고리즘의 도입을 위해 제조 공정에 대한 유전자 비트열을 [000011112222]로 표시한다. (Fig. 8)

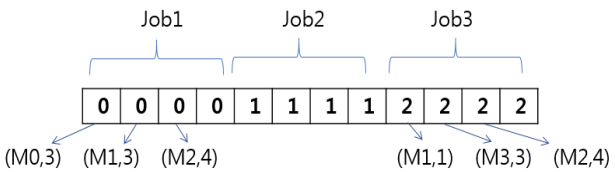


Fig. 8 Bitstring of the manufacturing process

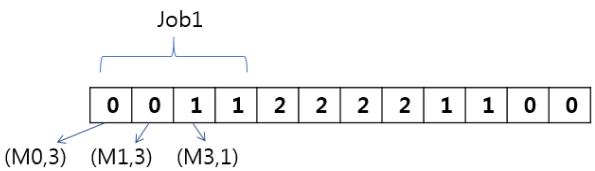


Fig. 9 Case of [001122221100]

여기서 유전자 비트열의 앞에서 4 개의 유전자 0 은 작업 $J0$, 다음의 4 개의 1 은 작업 $J1$, 2 는 작업 $J2$ 를 나타낸다. 또한 작업 $J0$ 의 첫 번째 0 은 기계 $M0$을 3 시간 사용하는 작업, 두 번째 0 는 기계 $M1$ 을 3 시간 사용하는 작업 등을 나타낸다. 이 유전자 비트열은 $J0$ 작업을 수행하는 동안에 사용이 되고 있지 않은 기계들을 활용하여 $J1$ 또는 $J2$ 작업을 수행하고 있다. 그러나 이 스케줄이 최적인지는 알 수 없다. 만일 유전자 비트열이 [001122221100]이라면 작업 $J0$ 에서 기계 $M0$ 를 3 시간, 기계 $M1$ 을 3 시간, 기계 $M3$을 1 시간, 기계 $M0$를 2 시간 사용한 후, 작업 $J1$ 등을 수행한다는 것을 나타낸다. (Fig. 9)

JSSP 에서의 주요 목표는 제품 생산을 위해 할당된 작업이 모두 끝날 때까지 총 소요시간(total processing time 또는 makespan)이 최악의 실행시간(Worst-case execution time, WCET)을 넘지 않으면서 최적의 작업 순서를 찾고, 각 작업 센터 간의 유휴시간을 줄이는 데 중점을 둔다. (존슨 규칙(Johnson's rule)이라고 부름)

최적의 작업 순서(최적 해)를 찾기 위한 기준으로 다양한 지표를 사용할 수 있으나 일반적으로 다음 지 Table -(1) makespan, (2) mean completion time (작업 별 끝나는 시점의 평균), (3) maximum flow time(작업 별 흐름 공정에서 작업하는 시간을 합산하였을 때, 가장 최대 값), (4) total tardiness(작업 별 마감 기한과 실제 완료 시간 간 차이를 다 합산한 것) – 들이 이용된다.

작업과 시간의 관계를 다음과 같이 간트 차트(Gantt Chart)로 나타내면 편리하다. 간트 차트는 프로젝트 일정관리를 위한 바 형태의 도구로서 각 업무별로 일정의 시작과 끝을 그래픽으로 표시하여 전체 일정을 한 눈에 파악할 수 있게 하여 생산성 향상에 기여한다. 간트 차트에는 Job Gantt Chart 와 Machine Gantt Chart 가 있으며 여기서는 Machine Gantt Chart 만을 나타내었다. 간트 차트에 의한 프로젝트 관리 도구로서 Microsoft Office Project, Teamgantt, FineReport 외 다양한 SW 가 있다. [24]

Table 5 은 유전자 비트열 [000011112222]에 대한 작업 스케줄의 간트 차트, Table 6 은 기계 스케줄의 간트 차트의 예이다. 여기서는 간단히 간트 차트를 나타내었지만 이 간트 차트에 의한 작업 스케줄이 최적인지에 대한 판단을 하지 않았다. 유전자 비트열 [000011112222]가 나타낼 수 있는 작업 스케줄은 매우 많기 때문에 유전 알고리즘에 의존하지 않고 최적 작업 스케줄을 구하는 것은 매우 어렵다.

Table 5 Gantt chart of the work schedule for the bitstring [000011112222]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
J0	M0	M0	M0	M1	M1	M1	M2	M2	M2	M2	M3												
J1	M3			M0	M0						M2	M2	M2	M2	M2	M1	M1						
J2	M1	M3	M3	M3												M2	M2	M2	M2	M0	M0		

[000011112222] Makespan=21

Table 6 Gantt chart of the machine schedule for the bitstring [000011112222]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
M0	J0	J0	J0	J1	J1															J2	J2		
M1	J2			J0	J0	J0										J1	J1						
M2							J0	J0	J0	J0	J1	J1	J1	J1	J1	J2	J2	J2	J2				
M3	J1	J2	J2	J2							J0												

[000011112222] Makespan=21

간트 차트를 작성하기 위해서는 다음과 같은 규칙이 적용되어야 한다.

(1) 모든 작업을 완료하는 데 필요한 시간으로 총 작업시간을 설정해야 하고, 작업 시작에 대한 하한과 작업(J,M)을 완료하는 상한시간을 설정해야 한다. 즉

$$start_{J.M} \geq 0 \quad (6)$$

$$start_{J.M} + dur_{J.M} \leq makespan$$

(2) 모든 선행 작업은(J,M)이 시작되기 전에 완료되어야 한다.

$$start_{k.n} + dur_{k.n} \leq start_{J.M} \quad (7)$$

(3) 기계 M_j 에서 수행되는 모든 작업에 대해 동일한 시간대에 중복 작업이 없어야 한다.

$$start_{J.M} + dur_{J.M} \leq start_{J.M} \quad (8)$$

또는

$$start_{k.m} + dur_{k.m} \leq start_{J.M}$$

Table 7과 Table 8에 유전자 비트열 [001122221100]에 대한 작업스케줄과 기계스케줄에 대한 간트 차트를 나타내었다. 유전자 비트열이[000011112222] 인 경우에 전체 작업을 종료하는데 21 시간이, 유전자 비트열이 [001122221100]인 경우는 18 시간이 걸리는 것을 알 수 있다. [001111002222] 유전자 비트열은 [000011112222] 유전자 비트열에 대한 돌연변이 유전자 조작을 통해 생성할 수 있다. 따라서 작업 스케줄 문제 유전 알고리즘을 적용하여 유전자 조작을 통해서 다양한 유전자 비트열을 생성하고 각각의 경우에 대한 총 소요시간을 평가하여 최적해를 구하는 것이 가능하다. 단, 유전자 조작을 수행할 때에 같은 시간대에 동일한 작업이 수행되어서는 안되고 또한 생성한 유전자 비트열은 반드시 각 작업이 4 회씩 행해져야 한다는 것에 주의하여야 한다.

Table 7 Gantt chart of the work schedule for the bitstring [001122221100]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
J0	M0	M0	M0			M1	M1	M1						M2	M2	M2	M2	M3					
J1	M3			M2	M2	M2	M2	M2	M0	M0	M1	M1											
J2	M1	M3	M3	M3						M2	M2	M2	M2	M0	M0								

[001122221100] Makespan=18

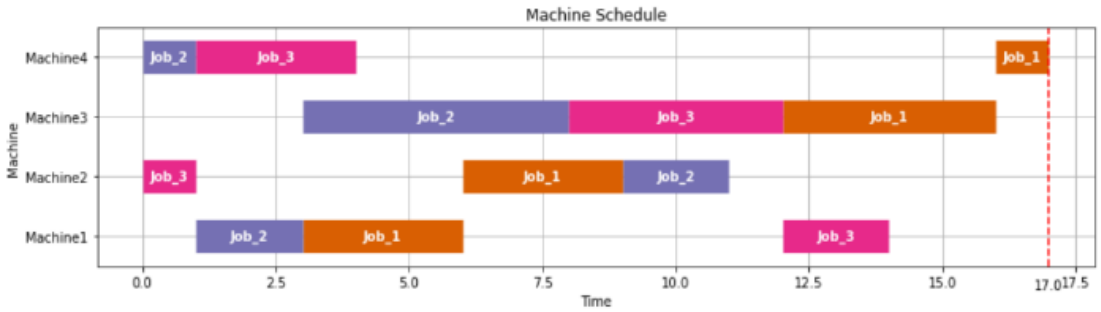
Table 8 Gantt chart of the machine schedule for the bitstring [001122221100]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
M0	J0	J0	J0	J1	J1				J2	J2													
M1	J2			J0	J0	J0								J1	J1								
M2				J2	J2	J2	J2	J1	J1	J1	J1	J0	J0	J0	J0								
M3	J1	J2	J2	J2														J0					

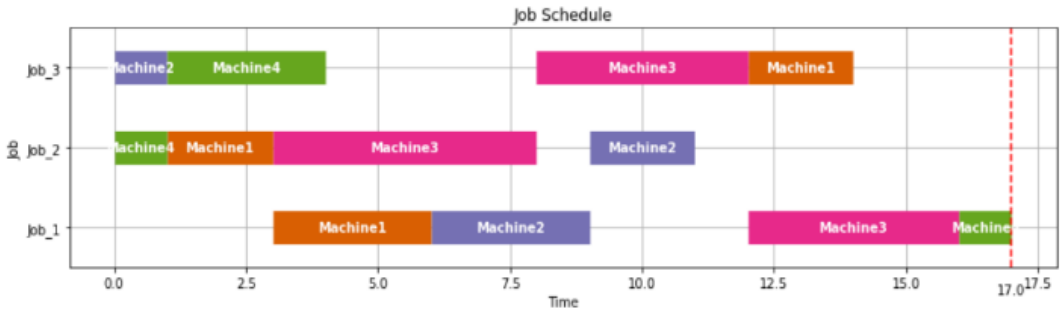
[001122221100] Makespan=18

Fig. 10 과 11 같이 유전 알고리즘 해석 결과로부터 여기서 다룬 (3 작업 x 4 기계) 문제에서 유전자 비트열이 [000011112222]인 경우에 총 작업시간은 17.0 시간, 유전자 비트열이 [001122221100]인 경우에 총 작업시간은 15.0 시간으로 단축되는 것을 알 수 있다.

여기서 사용한 작업 스케줄 문제의 유전 알고리즘은 문헌[25]을 이용하였음을 밝혀 둔다. 독자들은 이 프로그램을 적절히 수정하여 개별 생산 공정의 최적 스케줄 설계에 활용할 수 있을 것이다. (부록 3)

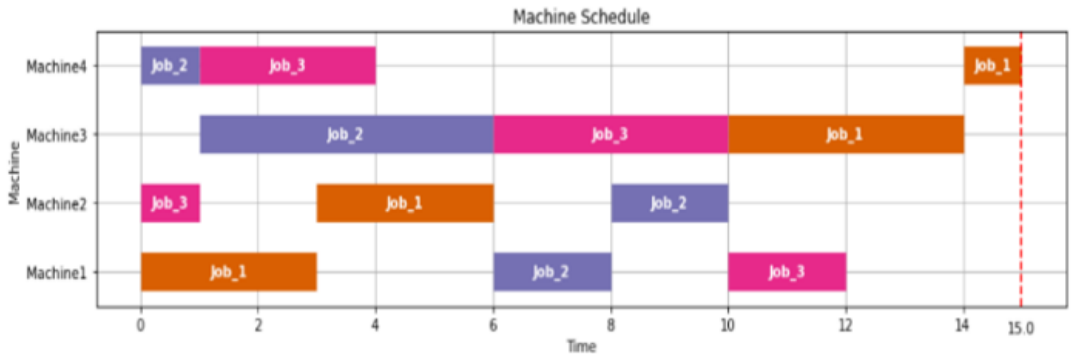


(a)

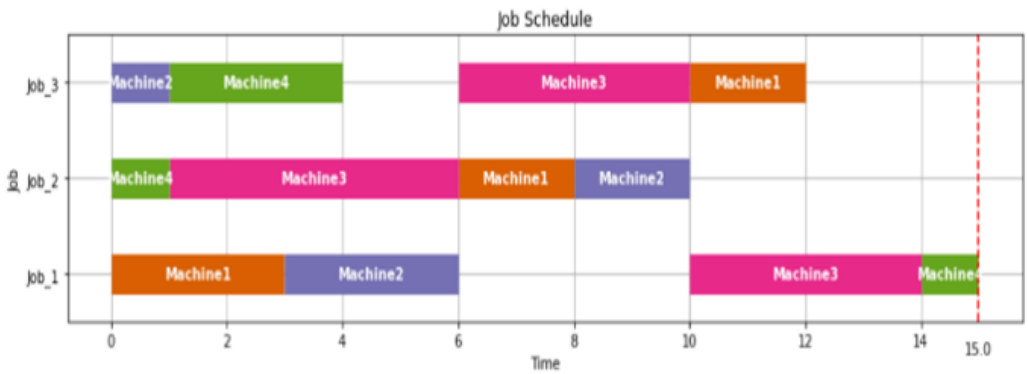


(b)

Fig. 10 (a) Machine schedule and (b) job schedule of bitstring [000011112222] by using GA



(a)



(b)

Fig. 11 (a) Machine schedule and (b) job schedule of bitstring [001122221100] by using GA

5. 결 론

본 해설에서는 유전 알고리즘에 대한 설명과 STP, 소성가공문제 및 작업스케줄 설계에의 적용을 통해 유전 알고리즘 프로그램의 응용에 대한 이해를 높이도록 하였다. 본 해설을 통해 독자들은 유전 알고리즘에 대한 최소한의 지식을 갖추어 소성가공분야의 산업현장과 연구에 도움이 되길 바란다. 본 해설에 사용한 python 프로그램을 한국소성가공학회 홈페이지의 일반자료실에 올려놓았다. 유전 알고리즘을 소성가공 연구에 활용하고자 하는 독자들은 프로그램을 각자의 문제에 맞게 수정하여 사용하길 추천한다.

본문 중에 기술한 방문판매원 문제, 점진판재 성형 문제, 작업 스케쥴 문제에 대한 유전 알고리즘 프로그램을 한국소성가공학회의 자료실에 GA-1, GA-2, GA-3 으로 게재하였다. 참고 바랍니다.

후 기

이 논문은 2021 대한민국 교육부와 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2019R1A2C1011224).

REFERENCES

[1] M. Mitchell, 1996, An Introduction to Genetic Algorithms, MIT Press.

[2] E. H. L. Aarts, A. H. Eiben, K. M. V. Hee, 1989. A general theory of genetic algorithms, In Computing Science Notes., Eindhoven University of Technology.

[3] D. G. Goldberg, 1989, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley.

[4] E. Wirsansky, 2020, Hands-On Genetic Algorithms with Python: Applying genetic algorithms to solve real-world deep learning and artificial intelligence problems, CreateSpace.

[5] J. H. Holland, 1992, Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence, A Bradford Book

[6] J. S. Chung, S. M. Hwang, 1996, Application of Genetic Algorithm to Die Shape Otimization in Extrusion, Trans. Mater. Process., Vol. 5, No. 4, PP. 269~280.

[7] W. S. So, J. H. Jung, 2006, Development of Production Scheduling Management Program using Genetic Algorithm for Polymer Production, Korean Chem. Eng. Res., Vol. 44, No. 2, April, 2006, pp. 149-159.

[8] D. H. Kim, B. W. Yoon, J. H. Lee, 2011,Size and Shape Optimization of Truss Structures using Micro Genetic Algorithm, J. Kor. Society. Steel. Constr., Vol. 23, No.4, pp.465~474.

[9] Y. H. Lee, E. T. Park, J. Kim, B. S. Kang, W. J. Song, 2019, Optimization on Weight of High Pressure Hydrogen Storage Vessel Using Genetic Algorithm, Trans. Mater. Process., Vol. 28, No. 4, PP. 203~211.

[10] D. Hermawanto, 2013, Genetic Algorithm for Solving Simple Mathematical Equality Problem, Comput. Sci.

[11] 島田拓弥, 2016, M.S., 遺伝的アルゴリズムを用いた勤務スケジュール作成, Chuo university.

[12] B. T. Zhang, 1995, Genetic Algorithm Theory and Applications, Inst. Electr. Information. Eng., Vol. 22, No. 11, pp. 1290~1300.

[13] S. Chatterjee, C. Carrera, L. A. Lynch, 1996. Genetic algorithms and traveling salesman problems, Eur. J. Oper. Res., Vol. 93, No. 3, pp. 490~510.

[14] C. Sheppard, 2018, Genetic Algorithms with Python, CreateSpace.

[15] W. J. Lee, H. Y. Kim, 2005, Genetic Algorithm Implementation in Python, Proc. Kor. Information. Process. Society Conf., Vol. 12, No. 1, PP. 473~476.

[16] https://pythonhealthcare.org/2018/10/01/94-genetic-algorithms-a-simple-genetic-algorithm/

[17] H. Iseki, K. Kato, S. Sakamoto, 1992, Flexible and Incremental Sheet Metal Bulging Using a Path-Controlled Spherical Roller. Trans. Jpn. Soc. Mech. Eng. Vol. 58, pp. 3147~3155.

[18] T. McAnulty, J. Jeswiet, M. Doolan, 2017 Formability in single point incremental forming: A comparative analysis of the state of the art. CIRP. J. Manuf. Sci. Technol, Vol.16, pp. 43~54.

[19] S. Gatea, H. Ou, G. McCartney, 2016, Review on the influence of process parameters in incremental sheet forming, Int. J. Adv. Manuf. Technol., Vol. 87, pp. 479~499.

[20] S. Yang, Y. S. Kim, 2020, Optimization of Process Parameters of Incremental Sheet Forming of Al3004 Sheet Using Genetic Algorithm-BP Neural Network, J. Kor. Acad.-ind. Coop. Society., Vol. 21, No. 1 pp. 560~567.

[21] A. Lawrynowicz, 2011, Genetic algorithms for solving scheduling problems in manufacturing systems, Found. Manag., Vol. 3, No. 2, pp. 7~25.

[22] B. J. Park, H. R. Chol, H. S. Kim, 2003, A hybrid genetic algorithm for the job shop scheduling problems, Comput. Ind. Eng., Vol. 45, No. 4, pp. 597~613.

[23] T. C. E.Cheng, B. M. T. Lin, 2009, Johnson's rule, composite jobs and the relocation problem, Eur. J. Oper. Res., Vol. 192, No. 3, pp. 1008~1013.

[24] J. M. Wilson, 2003, Gantt charts: A centenary appreciation, Eur. J. Oper. Res., Vol. 149, No. 2, pp. 430~437.

[25]https://colab.research.google.com/github/jckantor/ND-Pyomo-Cookbook/blob/master/docs/04.03-Job-Shop-Scheduling.ipynb#scrollTo=0k5vVxKIshoF