멀티모달 함수의 최적화를 위한 멘델 연산 유전자 알고리즘

A Genetic Algorithm with a Mendel Operator for Multimodal Function Optimization

송 인 수, 심 재 완, 탁 민 제
(In-Soo Song, Jae-Wan Shim, and Min-Jea Tahk)

Abstract : In this paper, a new genetic algorithm is proposed for solving multimodal function optimization problems that are not easily solved by conventional genetic algorithm(GA)s. This algorithm finds one of local optima first and another optima at the next iteration. By repeating this process, we can locate all the local solutions instead of one local solution as in conventional GAs. To avoid converging to the same optimum again, we devise a new genetic operator, called a Mendel operator which simulates the Mendel's genetic law. The proposed algorithm remembers the optima obtained so far, compels individuals to move away from them, and finds a new optimum.

Keywords : mendel operator, genetic algorithm, multimodal function optimization

I. 서론

일반적인 유전자 알고리즘은 주어진 목적함수의 단 하나의 최적값을 찾기 위해 고안되었다. 따라서 비슷한 적합치(fitness)를 가지는 여러 개의 최적점이 존재하는 문제에서는 확률적인 우연에 따라 정해지는 단 하나의 해만을 얻게 된다. 이 경우 얻게 되는 해 이외의 최적점도 최종해 결정 이전에 충분히 고려되어야 하지만, 일반적인 유전자 알고리즘의 반복 수행으로는 모든 최적점을 찾는다는 것을 보장할 수 없다. 따라서 멀티모달 함수의 최적화를 고려한 유전자 알고리즘이 필요하다. 멀티모달 함수 최적화 기법의 유용성은 유전자 알고리즘의 대표적인 문제점으로 지적되는 디셉션(deception)[1][2] 에서도 나타난다. 일반적으로 유전자 알고리즘은 전역 최적화에 있어서 비교적 탁월한 성능을 보이지만 전역 최적점보다 강력한 지역 최적점이 존재하는 일부 특정 지형의 함수에 대해서는 그 지역 최적점으로만 반복해서 수렴하는 경우가 있다. 이 경우 멀티모달 함수 최적화 기법을 적용하면 전역 최적점을 보다 효율적으로 찾을 수 있다.

이러한 목적에 따라 현재까지 진화 알고리즘에 있어 많은 멀티모달 함수 최적화 기법이 개발되었는데, 주요한 접근 방법의 하나는 개체의 서식지에 따른 종화(niche and speciation) 방법이었다. 이 방법은 개체들이 그 위치에 따라 각각 다른 종으로 분화됨으로써 여러 개의 최적점으로 동시에 수렴하게 된다. Cavicchio의 preselection scheme[3]은 이 개념을 처음으로 유전자 알고리즘에 도입한 방법으로 꼽힌다.

이 방법에서 각 자손은 자신의 적합치가 자신의 부모보다 월등할 때 부모를 대체하게 된다. 이 방법은 후에 De Jong의 crowding method[4]로 확장되는데, 이 기법에서는 개체군의 일부만이 자손을 번식하게 하고 이 자손들이 그들과 가장 닮은 개체들을 대체하는 방식으로 진화가 이루어진다. 이런 종화의 방법 중 가장 유명하고 널리 쓰이는 것은 Goldberg에 의한 sharing method[5]이다. 이 방법에서는 특정 거리 내에 있는 개체들끼리 서로 적합치를 공유하게 되며, 각 개체는 이 공유에 따라 나누어진 적합치에 따라 진화하여 여러 개의 최적점을 동시에 찾게 된다. 그러나 이 방법은 문제에 대해 의존적인 파라미터를 가지고 있고 그 값의 선택에 따라 수렴 성능이 달라지는 약점이 있다.

종화를 구현하기 위한 다중 개체군 구조를 형성하는 시도도 많은 연구가 이루어졌다. MRSC_GA(Minimal Representation Size Criterion GA)[6]에서는 최적점에 대한 개체들의 분포를 가우시안(Gaussian) 분포로 가정하고, 전체 개체들의 분포를 몇 개의 가우시안으로 보는 것이 최적인가에 대한 연산을 통해 개체군 수를 결정한다. 이 때 결정된 개체군 수에 따라 상호 작용이 없는 개체군들이 병렬적으로 진화하게 되며 하나의 개체군마다 하나의 최적점을 찾게 된다. fGA(Forking GA)[7]에서는 각 개체들의 수렴 상태를 감시하면서 개체군을 분할하는 구조를 사용하였다.

이러한 서식지에 따른 종화 방법은 다음의 몇 가지 단점을 가지고 있다. 첫째, 이들 방법은 대개 최적점의 개수에 비례하는 많은 수의 개체수를 필요로 한다. 둘째, 각 최적점으로 개체수를 할당하는 과정에 있어서 특정 최적점에 대해서 개체수가 부족할 경우 이로 인해 전체의 수렴 과정이 크게 지연될 수 있다. 셋째, 이런 종화 구현을 위한 전략들은 때때로 심각한 계산 부담을 야기할 수가 있다. 또한 많은 경우에 문제 의존적인 파라미터를 가지기도 한다.

이런 단점들을 피하기 위해서, 이들 방법과 근본적으로 다른 sequential niche method[8]가 제안되었다. 이 기법에서는 하나의 최적점이 찾아지면 적합치 계산 함수가 갱신되어 다음 계산에서는 이 최적점이 다시 찾아지지 못하도록 한다. 그러나 이 방법도 역시 파라미터의 문제 의존성을 해결하지 못하였다.

이 논문에서는 멘델 연산을 이용하는 새로운 유전자 알

접수일자 : 2000. 4. 25., 수정완료 : 2000. 8. 11.
송인수 : JC Entertainment
심재완, 탁민제 : KAIST

고리즘이 제안되었다. 이 알고리즘은 sequential niche technique과 마찬가지로 매 반복마다 하나의 지역 최적점들을 차례로 찾도록 되어 있다. 그 동안 찾은 최적점들에 대한 정보가 멘델 연산의 속성치에 의해 기록되어 기존에 찾았던 최적점들을 피해 새로운 최적점을 찾는 것이다.

II. 멘델 연산

유전자 알고리즘의 많은 유전 연산자들은 자연계의 유전 현상들을 수치적으로 모사하여 고안되었으며 대개 개체의 변이들을 생성시키는 데 그 목적이 있다. 멘델 연산자도 마찬가지로 자연계의 멘델 유전 법칙을 모사함으로써 고안되었다. 멘델의 콩 실험에서 열성 인자를 가진 형질은 우성 인자를 가진 형질에 비해서 같은 형질을 가지는 자손의 번식에 있어서 불리함을 볼 수 있다. 이를 바탕으로 멘델 연산은 특정 염색체와 비슷한 스키마를 갖는 염색체를 진화 과정에서 열성처리하고 배제하기 위해 개발되었다.

이 연산을 정의하기 위해서 모든 염색체의 각 비트에는 하나씩 속성치가 부가되었다. 이 속성치는 그 비트의 형질을 구분하기 위한 것이며, R, H, D 중 하나의 문자로 표기하였다. 여기서 R은 순종의 열성을, H는 잡종을, D는 순종의 우성을 가리키며 (1)과 같이 해당되는 비트의 오른쪽 괄호 속에 표기하는 것으로 하였다.

$$0(D)\ 1(H)\ 0(D)\ 1(R)\ 1(R) \tag{1}$$

위 염색체에서 첫번째 유전자와 그것의 속성치는 0(D)로 표기되었는데, 이는 0이 이 염색체의 첫번째 비트 자리에서는 우성 형질임을 알려준다. 그러나 실제 자연계의 유전학에서는 어떤 유전자도 단독으로는 우성이나 열성의 속성을 띠지 못한다. 우성이나 열성의 분류는 개개의 유전자 자체에 행해지는 것이 아니라 여러 유전자들의 조합으로 나타나는 형질에 대해 행해지는 것이기 때문이다. 따라서 위에 사용된 표기법을 이해하기 위해서는 염색체의 각 비트는 하나의 유전자가 아닌 하나의 형질로 대응시키고 염색체는 이런 형질들의 조합으로 이해해야 한다. 즉 앞으로 사용하는 염색체 코드는 모두 유전자형이 아닌 표현형으로 사용하는 것으로 한다. 또한 혼동을 피하기 위해 "유전자"라는 표현 대신 "비트"라는 표현을 사용하기로 한다.

실제 유전자 알고리즘의 적용에 있어서는 대략 수십 개 이상의 염색체들이 사용되는 경우가 많으므로 이들 사이의 속성치 결정에 있어서 모순이 있는지 검사해 볼 필요가 있다. x_{ij} 를 i번째 염색체의 j번째 비트라고 하고, a_{ij}를 이 비트에 해당되는 속성치라고 하면, 모든 염색체의 속성치와 비트는 (2)를 만족해야 한다.

표 1에는 유명한 멘델의 콩 실험이 앞에서 말한 표기법에 의해 표현되어 있다. 이 콩들은 오로지 하나의 형질—콩의 모양(둥글거나 주름진)—만이 고려되었기 때문에 하나의 비트를 가지는 염색체로 표현될 수 있다. 여기서 0은 둥근 콩을, 1은 주름진 콩을 뜻하게 된다. 이 표의 첫번째 경우는 순종 우성의 콩과 순종 열성의 콩의 교배 결과를 보여주고, 두번째 경우는 첫번째 경우의 교배로 인해 생긴 자손들끼리의 교배 결과를 보여 준다.

속성치 할당 규칙

- if $j=l$ and $a_{ij}=a_{kl}$

  $x_{ij}=x_{kl}$

- if $x_{ij}=y$ and $a_{ij}=D$ or H, for all k

$$\begin{cases} x_{kj}=y \text{ when } a_{kj}=D \text{ or } H \\ x_{kj}=NOT(y)=1-y \text{ when } a_{kj}=R \end{cases} \tag{2}$$

- if $x_{ij}=y$ and $a_{ij}=R$, for all k

$$\begin{cases} x_{kj}=y \text{ when } a_{kj}=R \\ x_{kj}=NOT(y)=1-y \text{ when } a_{kj}=D \text{ or } H \end{cases}$$

표 1의 두 경우는 모두 멘델 연산의 한 예로 볼 수 있다. 표에서 나타난 경우 외에도 하나의 비트를 가진 염색체의 멘델 연산은 이와 같은 방법으로 실제 자연계의 대응되는 교배를 모사함으로써 정의될 수 있다. 만약 0(D)와 0(H)로 연산이 이루어진다면 이 연산의 결과는 순종의 둥근 콩과 잡종의 둥근 콩의 교배 결과로 해석할 수 있다. 멘델 유전 법칙에 따라 이 교배의 결과는 반은 순종의 둥근 콩이 나오며 반은 잡종의 둥근 콩이 나옴을 쉽게 알 수 있다. 이 결과를 멘델 연산의 표기법으로 표현하면 0(D)-50%와 0(H)-50%가 되는 것이다. 이런 식으로 하나의 비트를 가지는 염색체의 멘델 연산은 표 2에서 보는 것처럼 정의할 수 있다. 이 표에서는 0이 우성이고 1이 열성인 경우에 대해 표시되었으며, 그 반대 경우는 0과 1을 바꾸면 얻을 수 있다.

표 1. 멘델의 콩 실험.
Table 1. Mendel's pea experiments.

	Mendel's experiment	*Notation*
Parents	Smooth - 50% (Homo, Dominant) Wrinkled - 50% (Homo, Recessive)	0 (D) - 50% 1 (R) - 50%
Offsprings	Smooth - 100% (Hetero)	0 (H) - 100%
Parents	Smooth - 100% (Hetero)	0 (H) - 100%
Offsprings	Smooth - 25% (Homo, Dominant) Smooth - 50% (Hetero) Wrinkled - 25% (Homo, Recessive)	0 (D) - 25% 0 (H) - 50% 1 (R) - 25%

표 2. 단일 비트 염색체에 대한 멘델 연산.
Table 2. Mendel operations for 1-bit chromosome.

	0(D)	0(H)	1(R)
0(D)	0(D)(100%)	0(D)(50%) 0(H)(50%)	0(H)(100%)
0(H)	0(D)(50%) 0(H)(50%)	0(D)(25%) 0(H)(50%) 1(R)(25%)	0(H)(50%) 1(R)(50%)
1(R)	0(H)(100%)	0(H)(50%) 1(R)(50%)	1(R)(100%)

이 멘델 연산을 여러 개의 비트를 가지는 염색체의 경우까지 확장시키기 위해서는 멘델의 독립 유전의 법칙을 적용한다. 이 법칙에 따르면 여러 개의 형질을 가진 콩들의 교배 결과는 각각의 형질들에 대한 독립적인 교배의 조합으로 생각할 수 있다. 따라서 이 경우의 멘델 연산은 다음과 같이 정의된다. 우선 염색체의 위치가 같은 모든 비트쌍에 대하여 표 2에서 정의한 것과 같은 연산 법칙을 각각 독립적으로 적용시킨다. 그리고 이 결과를 모두 모아서 새로운 염색체를 구성한다. 만약 또 하나의 염색체가 필요할 경우엔 이 과정을 다시 처음부터 반복하면 된다. 이 연산은 교배의 경우처럼 두 개의 부모를 필요로 하기 때문에 개체군의 크기를 연산 후에도 유지하기 위해서는 두 개의 자식을 생성하여야 한다. 따라서 멘델 연산이 이루어지는 예를 들어보면 (3)과 같다.

- Parents:
  0(*D*)1(*R*)0(*H*)1(*D*)1(*D*)1(*H*)0(*R*)1(*H*)0(*H*)1(*D*)
  1(*R*)1(*R*)0(*D*)0(*R*)1(*H*)0(*R*)1(*H*)1(*H*)0(*H*)0(*R*)
- Offsprings:
  0(*H*)1(*R*)0(*H*)1(*H*)1(*H*)0(*R*)0(*R*)1(*D*)0(*D*)1(*H*)
  0(*H*)1(*R*)0(*D*)1(*H*)1(*H*)1(*H*)1(*H*)0(*R*)1(*R*)1(*H*) (3)

III. 멘델 연산의 효과

멘델 연산을 활용하기에 앞서 염색체 각 비트의 속성치를 결정하기 위한 기준이 되는 염색체를 먼저 결정해야 한다. 예를 들어 1(R)1(H)1(D)라는 3개의 비트로 구성된 염색체를 생각해 보자. 괄호 속에 표기된 R, H, D는 1이 첫번째 비트에서는 열성이고 두번째와 세번째 비트 자리에서는 우성임을 말해 주고 있다. 그렇다면 원래의 111이라는 염색체를 1(R)1 (H)1(D)로 표기하기 위해서는 어떤 기준이 필요하다는 것을 알 수 있다. 이 논문에서는 이러한 목적으로 기준이 되는 염색체를 "적대 염색체(outcast chromosome)"라고 부르기로 한다. 적대 염색체는 모든 비트가 각각의 자리에서 열성인 비트로 구성된다. 예를 들어 1(R)1(H)1(D)의 경우에서는 100이 적대 염색체가 된다. 적대 염색체는 정의상 모든 비트가 열성으로 R의 속성을 띠게 되므로 따로 속성을 표기할 필요는 없다.

멘델 연산이 유전자 알고리즘에서 작용하는 바를 쉽게 보기 위해서 예제를 하나 다루어 보았다. 이 문제에서는 교배, 돌연변이 등의 멘델 연산을 제외한 유전 연산을 일체 배제하였으며 선택 과정도 넣지 않았다. 따라서 목적 함수도 필요하지 않으며 오로지 유전자 알고리즘의 루프 내에서는 멘델 연산만이 수행되도록 하였다. 이 문제는 하나의 파라미터 x만을 가지는 문제로 실수값으로 변환된 값의 범위는 0에서 1까지로 하였다. 적대 염색체는 0101100101010110으로 16비트의 그레이 코드가 사용되었으며 변환된 실수값은 0.4312199이다. 우선 초기 개체군은 x_{ij}이 i번째 염색체의 j번째 비트를 가리키고, a_{ij}가 해당되는 속성치를 가리킨다고 할 때 (4)와 같이 결정된다.

그림 1에서 개체들은 멘델 연산에 의해서 적대 염색체로 지정된 점을 피해 새롭게 분포된다는 것을 볼 수 있다. 이로써 알 수 있듯이 멘델 연산은 개체들로 하여금 특정점 근처로 모여드는 것을 막을 수 있다. 이런 특성은 현재의 개체들이 주어진 강력한 지역 최적점으로부터의 탈출 가능성을 보다 높여준다. 예를 들어 어떤 지역 최적점 x_1^*이 일반적인 유전자 알고리즘에서 찾아졌다고 가정하자. 이것은 곧 x_1^*의 베이신(basin)이 탐색 영역내에서 넓은 부분을 차지하고 있으며 적합치 역시 이 지역 최적점이 전역 최적점인가의 여부와 관계없이 상당히 큰 값을 가지고 있다는 것을 뜻한다. 따라서 단순히 일반적인 유전자 알고리즘을 반복적으로 수행한다면 이 지역 최적점만이 반복적으로 발견되는 것이 예상 가능하다. 이를 고려해 볼 때 멀티모달 함수에 있어서 다른 최적점을 찾기 위해서는 이미 찾은 최적점으로 다시 한번 수렴하는 것을 막는 것이 우선시되어야 한다. 이는 기존에 찾은 최적점을 적대 염색체로 지정하고 멘델 연산을 수행함으로써 달성할 수 있다. 멘델 연산은 계속적으로 현재의 개체들을 적대 염색체로부터 멀어지게 하므로 개체들은 과거에 찾은 최적점으로는 다시 수렴할 수 없게 된다. 여기에서 주목할 만한 점은 이러한 탈출이 적대 염색체 근처의 적합치 함수의 지형적 특성이나 적합치 함수값에 영향을 받지 않는다는 것이다. 오로지 각 염색

$$
\begin{aligned}
&\cdot\ x_{ij} = 0 \text{ or } 1 \text{ (Randomly determined)} \\
&\cdot\ a_{ij} = D \text{ when } x_{ij} \neq \tau_j \\
&\quad\ a_{ij} = R \text{ when } x_{ij} = \tau_j
\end{aligned} \tag{4}
$$

여기서~ tau_{j}는~ 적대~ 염색체의~j번째 ~비트의 ~값이다.

이렇게 생성된 초기 개체군에 대하여 5 세대에 걸쳐멘델 연산이 수행된 결과가 그림 1에 도시되어 있다.

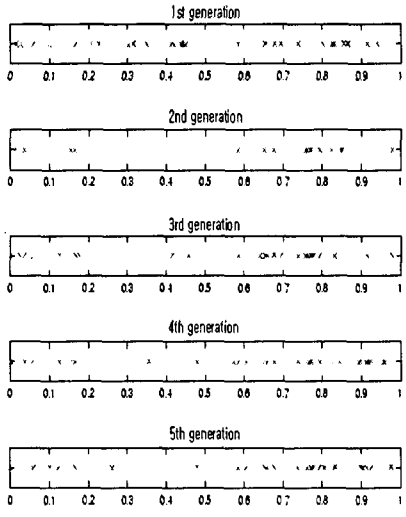


그림 1. 멘델 연산에 의한 개체 분포 상황 (적대 염색체 = 0.43121).

Fig. 1. Population distribution (the outcast chromosome= 0.43121).

체가 적대 염색체로부터 떨어진 유전자적 거리만이 이 연산에 영향을 주게 된다. 이런 이유로 멘델 연산은 적합치를 조정하는 다른 멀티모달 최적화 기법과 비교하여 문제에 대한 강인성이 상대적으로 뛰어나다. 또한 일반적인 이진코드 대신에 그레이 코드가 사용되었다는 것에도 주목할 필요가 있다. 이진코드와 비교해서 상대적으로 그레이 코드는 코드 자체는 유사하면서도 디코딩된 값은 무척이나 다른 경우가 상당히 드물다. 따라서 그레이 코드를 사용함으로써 멘델 연산은 이전의 최적점이 어느 정도의 오차를 가지고 결정된 경우에 대해서도 안전하게 작용할 수 있다.

이번에는 같은 높이의 봉우리들을 가지는 정현파 곡선 문제에 대해서 멘델 연산을 적용하여 보기로 한다. 이 문제에서는 멘델 연산 외에도 교배나 돌연변이 같은 일반적으로 사용되는 유전 연산을 포함시켰으며 선택 과정도 수행하였다. 여기서 교배나 돌연변이를 수행할 때는 해당하는 비트 뿐만이 아니라 그 속성치에 대해서도 이 연산이 적용되도록 해야 한다. 예를 들어 어떤 비트에 돌연변이가 일어나서 그 비트가 바뀌어 버리면 이에 해당하는 속성치도 열성이면 우성이나 잡종으로, 우성이나 잡종이면 열성으로 바뀌어야 한다. 이 문제의 목적 함수는 $f(x)=\sin 4\pi x$ 로 파라미터 x의 범위는 0에서 2까지로 정했으며, 0.125, 0.625, 1.125, 1.625에서 4개의

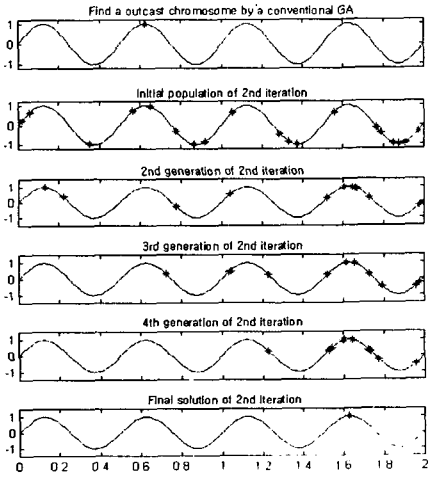


그림 2. 정현파 곡선 문제에 대한 멘델 연산 적용 결과.
Fig. 2. GA simulation results with the Mendel operation on equal peaks.

같은 높이의 봉우리를 가진다. 30비트의 그레이 코드를 사용하여 멘델 연산을 적용한 결과가 그림 2에 나타나 있다.

그림 2의 첫번째 그래프는 이 예제에 대해 일반적인 유전자 알고리즘을 적용하였을 때 얻은 하나의 지역 최적점이다. 이 최적점을 적대 염색체로 지정한 후에 멘델 연산을 적용한 결과가 두번째 그래프부터 나와 있다. 여러 번의 반복 수행 결과 그림에서처럼 적대 염색체로 지정한 최적점이 배제되어 다른 최적점으로 수렴함을 알 수 있다.

Ⅳ. 여러 개의 적대 염색체에 대한 멘델 연산

앞 절에서는 하나의 최적점이 찾아진 경우 그 최적점을 피해 다른 최적점을 찾기 위한 멘델 연산이 제안되었다. 그러나 대부분의 멀티모달 함수 최적화 문제에 있어서는 최적점은 두 개 이상으로 존재하기 때문에 두 개 이상의 최적점이 찾아진 경우에 대한 멘델 연산의 정의가 필요하게 된다. 이러한 경우에는 그림 3에서와 같이 적대 염색체가 여러 개가 되므로 자연히 하나의 비트에 대하여 여러 개의 속성치가 부과 된다.

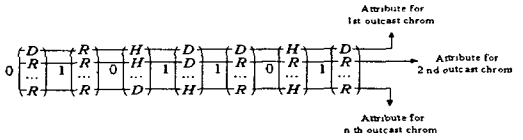


그림 3. 여러 개의 적대 염색체를 가지는 염색체의 구조.
Fig. 3. An example of a chromosome with multiple outcast chromosomes.

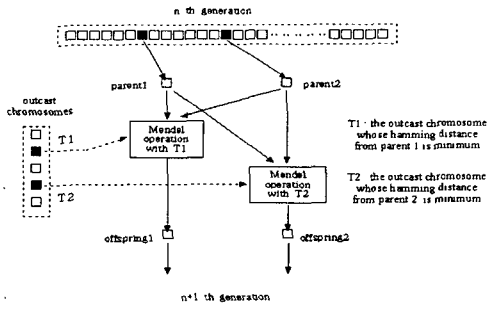


그림 4. 여러 개의 적대 염색체에 대한 멘델 연산.
Fig. 4. Applying the Mendel operation for multiple outcast chromosomes.

적대 염색체가 존재할 경우의 멘델 연산에 대해 설명하고 있다. 이를 세부적으로 설명하면 먼저 멘델 연산을 위해 두 개의 염색체가 개체군에서 선택된다. 그러면 그들은 그림 4에서 보이는 대로 두번의 멘델 연산을 거치게 된다. 하나는 T_1을 적대 염색체로 생각하는 멘델 연산이며, 나머지 하나는 T_2를 적대 염색체로 생각하는 멘델 연산이다. 여기에서 T_1과 T_2는 서로 아무런 상관관계 없이 다음과 같은 독립적인 전략에 따라 결정된다.

P_1과 P_2는 멘델 연산을 위해 선택된 두 개의 개체를 나타내고, $\tau_1, \tau_2, \tau_3, \ldots, \tau_m$는 이미 찾아낸 m개의 최적점들을 의미하며, $H(a,b)$는 a, b 두 염색체 사이의 hamming distance를 표현한다고 하면, T_1과 T_2가 선택되는 방법은 (5)와 같다.

$$\begin{aligned} &\cdot\ T_1=\tau_i,\ \text{where } \tau_i \text{ is selected to minimize } H(P_1,\tau_i)\ (i=1,2,\ldots,m)\\ &\cdot\ T_2=\tau_j,\ \text{where } \tau_j \text{ is selected to minimize } H(P_2,\tau_j)\ (j=1,2,\ldots,m) \end{aligned} \tag{5}$$

$H(P_1,\tau_i)$를 최소화시키는 τ_i를 찾는다는 것은 결국 기존에 찾은 최적점들, 즉 적대 염색체들 중에서 P_1과 가장 가까운 적대 염색체를 고른다는 것을 의미한다. 이런 전략을 사용하여 비록 적대 염색체가 여러 개 있는 경우에 대해서도 하나의 자손 생성을 위해 해당하는 부모로부터 가장 가까운 하나의 적대 염색체만을 사용하게 된다. 이것은 모든 개체들이 각각 자신으로부터 가장 가까운 적대 염색체를 회피하는 것만으로도 모든 적대 염색체를 탐색 영역에서부터 배제하는 것이 가능하다는 것에 기초한 기법이다. 이 멘델 연산의 결과로 T_1, T_2의 속성치는 직접적인 연산의 결과에 의해 바뀌게 되고 나머지 속성치들은 2절에서 얘기한 속성치 할당 규칙에 따라 수정된다. 이 기법을 앞 절에서 예로 들었던 여러 개의 같은 높이의 봉우리를 가진 정현파 곡선 문제에 적용한 결과가 그림 5에 도시되어 있다. 이 그림의 첫 번째 그래프는 이미 세 개의 서로 다른 최적점을 찾은 것을 보여준다. 두 번째 그래프부터는 이 세 개의 최적점들을 모두 적대 최적점으로 지정하고 멘델 연산을 적용하여 결국 나머지 하나의 최적점을 찾게 됨을 보여 주고 있다.

이상의 멘델 연산을 이용한 멀티모달 최적화 기법의 모든 단계를 요약하면 다음과 같다.

단계	요약	설명
1	지역 최적점 찾기	일반적인 유전자 알고리즘 혹은 여타의 최적화 기법을 이용하여 하나의 지역 최적점을 찾는다. 찾아진 해를 멘델 연산을 적용하기 위해 그레이 코드로 변환한다.
2	적대 염색체 지정	찾아진 지역 최적점들을 모두 적대 염색체로 지정한다.
3	개체군 초기화	지정한 적대 염색체들에 대한 멘델 속성치를 가지는 개체군을 초기화한다. 이때 속성치 할당 규칙이 지켜지도록 주의한다.
4	개체군의 수렴 여부 확인	개체군의 수렴 여부를 체크한다. 만약 수렴되었다고 판단되었을 때는 8의 단계로 간다. 허용 최대 세대수를 넘기도록 수렴하지 못하는 경우에는 계산을 종료한다.
5	유전 연산 수행	교배나 돌연변이 등의 유전 연산을 수행한다.
6	멘델 연산 수행	멘델 연산을 수행한다. 각 염색체에 대해서 이진 코드의 차가 가장 적은 적대 염색체를 찾아 이에 대한 멘델 연산을 수행한다.
7	다음 세대의 부모 선택	선택을 통해 다음 세대의 부모를 추려내고 4의 단계로 간다.
8	최고의 적합치를 가지는 개체 선택	현 개체군 중에서 최고의 적합치를 가지는 개체를 찾는다. 이 개체는 주어진 멀티모달 함수의 또 다른 최적해가 된다. 2의 단계로 간다.

V. 예제

이 절에서는 몇 가지의 멀티모달 함수에 대해 멘델 연산을 이용한 유전자 알고리즘으로 최적해를 구하는 과정을 제시하였다.

1. Himmelblau 함수

$$\cdot \max F(x,y) = 200 - (x^2+y-11)^2 - (x+y^2-7)^2 \quad (6)$$

$$\text{where } -6 \le x \le 6,\ -6 \le y \le 6$$

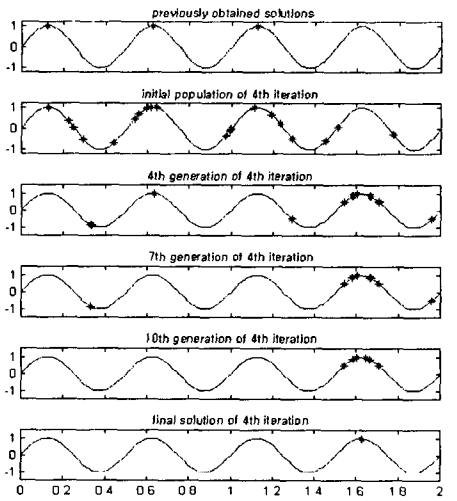


그림 5. 멘델 연산을 이용한 새로운 최적점 탐색.

Fig. 5. Finding a new optimum by applying the Mendel operation.

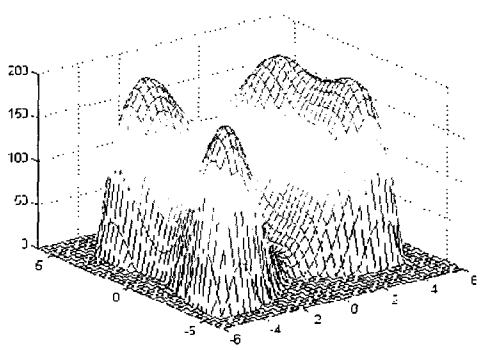


그림 6. Himmelblau 함수의 형상.

Fig. 6. The landscape of the modified Himmelblau's function.

표 3. Himmelblau 함수의 최적해.

Table 3. The solution of the modified Himmelblau's function.

	optimum(x,y)	function value
1st iteration	(-3.7793,-3.2832)	200.0
2nd iteration	(-2.8051,3.1313)	200.0
3rd iteration	(3.5864,-1.8486)	200.0
4th iteration	(2.9997,2.0002)	200.0
5th iteration	no more solution	

(6)은 최소화 문제의 예시로 만들어진 Himmelblau 함수를 최대화 문제로 바꾼 것이다[6]. 네 개의 같은 높이의 봉우리를 가지고 있으며 그 값은 모두 200이다. 멘델 연산을 이용한 유전자 알고리즘을 적용하여 계산한 결과가 표 3에 나타나 있다. 네 번의 반복 계산을 통해 네 개의 다른 최적해를 찾을 수 있었으며 다섯 번째 반복에서는 개체들이 수렴하는데 실패하여 계산이 중지되었음

을 볼 수 있다. 그림 7에는 이 계산 과정의 4번째 반복에서 해를 찾는 개체들의 모습을 보여 주고 있다. 3개의 서로 다른 최적해('T' 로 표시)의 베이신을 탐색 영역에서 차츰 배제하며 마지막 하나의 최적해를 찾아가는 과정을 볼 수 있다.

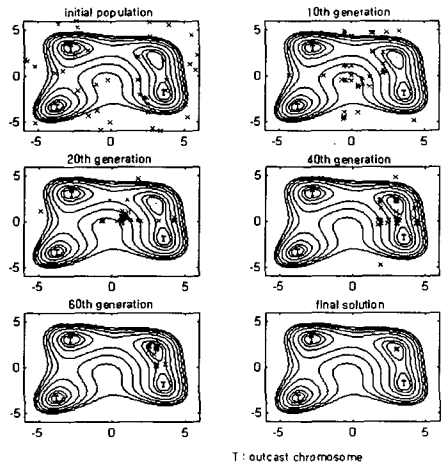


그림 7. 표 3의 4번째 반복에서의 개체 수렴 상황.

Fig. 7. Converging feature of population at 4th iteration.

제안한 기법과의 성능 비교를 위해서 같은 문제에 대해 공유 기법(sharing method)을 이용하여 계산해 보았다. 공유 함수(sharing function)는 다음의 식을 사용하였다.

$$Sh(d)=\begin{cases}1-\left(\dfrac{d}{\sigma_{share}}\right)^{\alpha}, & \text{if } d<\sigma_{share}\\ 0, & \text{otherwise}\end{cases} \tag{7}$$

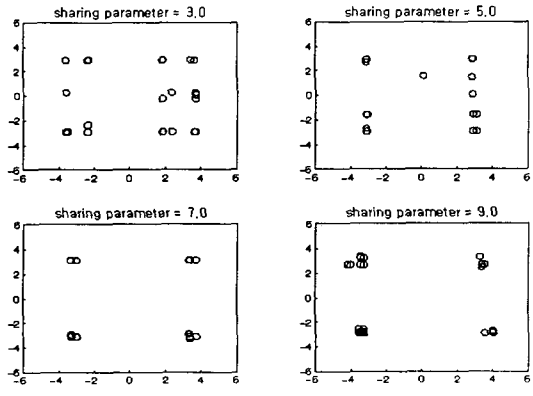


그림 8. 500 세대의 개체군 분포도 ($\alpha=1.0$).

Fig. 8. Population distribution at the 500th generation ($\alpha=1.0$).

공유 파라미터(sharing parameter) σ_{share}는 공유(sharing)의 정도를 제어하는 파라미터이며 그 최적값은 주어진 멀티모달 함수의 봉우리들간의 거리의 기대치에 따라 좌우된다. 또한 지수 파라미터 α는 공유 함수의 convexity의 정도를 결정한다. 이 두 값의 최적값을 결정하는 방법은 명쾌하게 정의되어 있지 않으며 대개의 경우 사용자의 시행착오를 통해 결정된다. 그림 8과 그림 9는 이 문제에 대해서 이 두 파라미터가 끼치는 영향을 보여 주고 있다. 공유는 각 개체의 적합치를 서로 나누어 가짐으로써 구현되기 때문에 잘못 정해진 제어 파라미터에 따라서는 그림에서처럼 실제로는 봉우리가 존재하지 않는 곳으로 개체들이 몰려들 수도 있다. 이런 시행착오의 과정을 거쳐 $\sigma_{share}=6.5$, $\alpha=0.9$로 결정했으며 이 값을 적용한 결과를 표 3에 나타내었다.

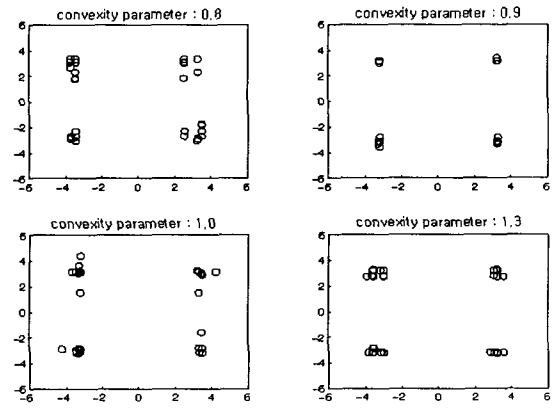


그림 9. 100 세대의 개체군 분포도($\sigma_{share}=6.5$).

Fig. 9. Population distribution at the 100th generation($\sigma_{share}=6.5$).

2. Goldstein-Price 함수

$$\begin{aligned}&\cdot \max F(x,y)\\ &=200-[1+(x+y+1)^2\\ &\quad *(19-14x+3y^2-14x+6xy+3y^2)]\\ &\quad *[30+(2x-3y)^2(18-32x+12x^2\\ &\qquad +48y-36xy+27y^2)]\\ &\text{where}\quad -2\le x\le 2,\ -2\le y\le 2\end{aligned} \tag{8}$$

표 4. Goldstein-Price 함수의 최적해들.

Table 4. The solutions of modified Goldstein-Price function.

	optimizer(x,y)	function value
1st iteration	(-0.5999, -0.4002)	170.0000
2nd iteration	(-0.0000, -1.0000)	197.0000
3rd iteration	(1.8004, 0.2290)	116.0000
4th iteration	no more solution	

(8)은 네 개의 지역 최대점을 가지며 그 크기는 모두 다르다. 표 4에 나타난 것처럼 그 중 세 개가 멘델 연산을 적용하여 찾을 수 있었고 그 중 두 번째 반복에서 찾은 해를 전역 최적해로 결정하였다. 지역 최적점 중 (1.2, 0.8)은 그 값이 너무 작고 베이신도 너무 협소하여 찾아내지 못하였다. 참고로 이 점에서의 함수값은 -640이다.

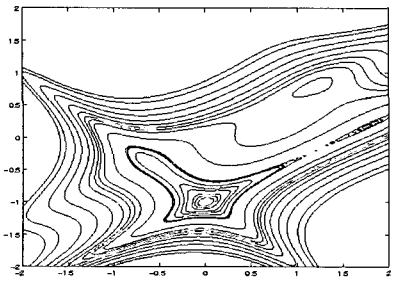


그림 10. Goldstein-Price 함수의 등고선도.

Fig. 10. Level curves of the modified Goldstein-Price function.

3. 구속 조건이 있는 Himmelblau 함수

앞에서 보여준 두 개의 예제는 모두 구속조건이 없는 멀티모달 함수 최적화 문제였다. 이 기법을 구속조건이 있는 경우로 확장시키기 위해서 진화 알고리즘에서 잘 알려진 구속 조건 처리 기법들을 적용하였다. 하나는 behavioral memory method[10][11]이고, 다른 하나는 the method of superiority of feasible points[11][12]이다. 이 예제에서는 behavioral memory method를 적용하여 구속 조건이 있는 Himmezxcvlblau 함수 최적화 문제를 다루어 보았다.

$$\text{max } F(x,y)=200-(x^2+y-11)^2-(x+y^2-7)^2$$

$$\begin{aligned} \text{subject to} \quad & x+y-1\le 0 \\ & -x+y-6\le 0 \\ & (x+4)^2-y-5\le 0 \end{aligned} \qquad (9)$$

$$\text{where} \quad -6\le x\le 6,\ -6\le y\le 6$$

Himmelblau 함수의 등고선과 구속 조건이 그림 11에 도

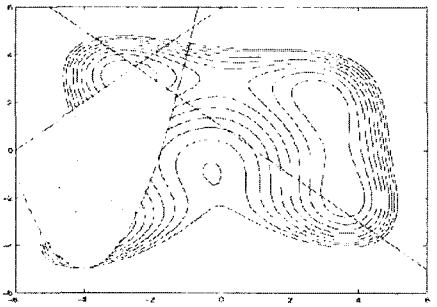


그림 11. 구속 조건이 있는 Himmelblau 함수의 등고선.

Fig. 11. Level curves of the modified Himmelblau's function and its constraints.

시되어 있다. 그늘진 부분이 구속조건을 만족하는 영역이다. 그림에서 보는 것처럼 네 개의 봉우리 중에 두 개는 구속 조건을 만족시키며 두 개는 이를 만족시키지 않는다.

Behavioral memory method는 주어진 문제의 구속 조건들을 각 단계마다 하나씩 순서대로 처리하며, 각 단계마다 정해진 flip threshold 만큼의 개체들이 구속조건을 충족시키면 다음 단계로 넘어 가게 된다. 이렇게 각 단계별로 각 구속 조건을 처리하며 최적해를 찾는 과정이 이 문제에 대하여 그림 12에 도시되어 있다. 멘델 연산을 적용한 멀티모달 최적화 결과는 표 3에 주어져 있다. 네 개의 해 가운데 구속 조건을 만족하는 (-3.7793,-3.2832)과 (-2.8051, 3.1313)만이 찾아졌으며 나머지 구속 조건을 위반하는 해들로는 수렴하지 않았다. 이 방법에서 중요한 것은 구속 조건들을 처리하는 순서와 적합한 flip threshold 이다. 그러나 이들을 결정하는 전략에 대해서는 명쾌한 이론이 세워져 있지 않은 상태이다. 또한 만일 구속 조건을 만족하는 지역이 하나의 연결된 지역이 아니라 여러 개로 나뉘어진 지역으로 되어 있다면 sharing method 같은 부가적인 장치가 없는 일반적인 유전자 알고리즘으로는 만족할 만한 성능을 얻을 수 없다. 이러한 단점에도 불구하고 이 방법은 유전자 알고리즘에 적용될 때 구속 조건을 만족시키지 않는 부분의 함수 모양에 상관없이 작용한다는 점에서 멘델 연산과 접목시켜 사용할 때 큰 장점을 가지고 있다. 만일 구속 조건을 만족시키는 지역이 하나의 연결된 지역으로 생각되는 문제에 있어서는 멘델 연산을 이용한 멀티모달 최적화 기법의 구속 조건 처리 기법으로 가장 적합한 방법이 될 것이다.

표 5. 구속 조건이 있는 Himmelblau 함수의 최적해.

Table 5. The solutions of modified Goldstein-Price function with artificial constraints.

	optimizer(x,y)	function value
1st iteration	(-3.7793, -3.2832)	200.0
2nd iteration	(-2.8051, 3.1313)	200.0
3rd iteration	no more solution	

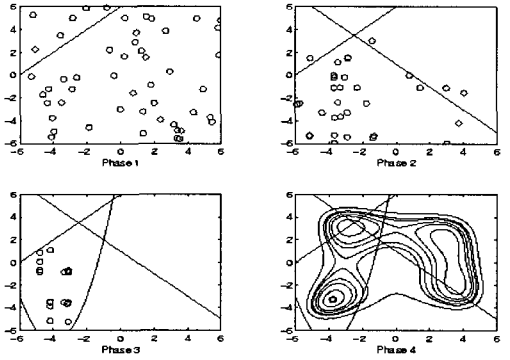


그림 12. Behavioral memory method를 적용하였을 때의 개체군 분포도.

Fig. 12. Population distribution when using behavior memory method.

4. 구속 조건이 있는 9개의 같은 봉우리를 가진 멀티모달 최적화 문제

이 예제에서는 멀티모달 함수의 구속 조건을 처리하기 위해서 method of superiority of feasible points가 적용되었다. 앞에서 사용한 behavioral memory method와는 달

리 이 방법은 여러 개로 나누어진 구속 조건을 만족하는 영역에 대해서도 유효하게 적용할 수 있다.

$$\begin{aligned} &\cdot \max F(x,y) \\ &\quad = 15\exp(0.01\cos(1.5x)+0.01\cos(1.5y)) \\ &\text{subject to} \quad 9(x+5)^2+y^2-36\le 0 \\ &\qquad\qquad -x+y^2\le 0 \\ &\qquad\qquad -x+4y-14\le 0 \\ &\text{where} \quad -6\le x\le 6,\ -6\le y\le 6 \end{aligned} \tag{10}$$

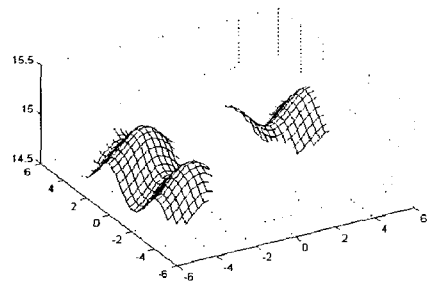


그림 13. 구속 조건이 있는 멀티모달 함수의 형상.

Fig. 13. The landscape of the artificial constrained multimodal function.

그림 13에서 보는 것처럼 구속 조건을 만족시키는 영역은 두 개의 떨어진 지역으로 나뉘어져 있기 때문에 순수한 behavioral memory method를 적용해서는 원하는 성능을 얻기 힘들다. 따라서 method of superiority of feasible points를 적용하였으며 이 방법에 따라 각 개체는 다음의 식에 의해서 적합치를 계산하였다.

$$eval(\mathbf{x}) = f(\mathbf{x}) - r\sum_{j=1}^{m} f_j(\mathbf{x}) - \theta(t,\mathbf{x}) \tag{11}$$

표 6. (10)의 최적화 결과.

Table 6. The solution of the artificial constrained multimodal function of (10).

	optimizer(x,y)	function value
1st iteration	(-4.1888, -4.1888)	15.3030
2nd iteration	(-4.1888, 0.0000)	15.3030
3rd iteration	(3.9998,0.0000)	15.2969
4th iteration	(0.0000,0.0000)	15.3030
5th iteration	no more solution	

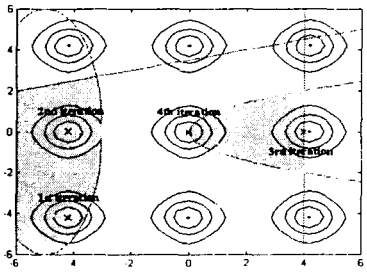


그림 14. (10)의 해.

Fig. 14. The solution of (10).

(11)은 한 가지 주목할 만한 항 $\theta(t,\mathbf{x})$를 제외하고는, 기본적으로 고전적인 벌칙 함수 방법과 같은 형태이다. $\theta(t,\mathbf{x})$는 세대수에 따라 변하는 부가적인 값으로서, 다음과 같은 한가지 법칙만 지킬 수 있으면 어떤 특정한 형태를 요구하지 않는다. 그 법칙은 다음과 같다. 어떤 개체든지 만일 구속 조건을 만족시킨다면, 구속 조건을 만족시키지 못하는 다른 어떤 개체보다도 큰 적합치를 가져야 한다는 것이다. 이 예제에서 사용한 $\theta(t,\mathbf{x})$는 다음과 같다.

$$\theta(t,\mathbf{x}) = \begin{cases} 0, & \text{if } \mathbf{x}\in F \\ \max\limits_{\mathbf{x}\in S-F}\{f(\mathbf{x}) - r\sum\limits_{j=1}^{m} f_j(\mathbf{x})\} & \\ \quad - \min\limits_{\mathbf{x}\in F}\{f(\mathbf{x})\}, & \text{otherwise} \end{cases} \tag{12}$$

(12)를 사용하여 주어진 문제를 멘델 연산을 적용하여 푼 결과가 표 6과 그림 14에 나타나 있다.

VI. 결론

본 논문에서 제안한 멘델 연산을 이용한 멀티모달 최적화 기법의 장점은 다음과 같이 요약된다. 첫째, 멀티모달 최적화 기법이면서도 한번에 단 하나의 최적해만을 찾기 때문에 종화를 통해 한번에 여러 최적해를 한꺼번에 찾는 다른 기법들에 비해 보다 작은 크기의 개체군을 사용하면서 보다 정밀한 탐색이 가능하다. 다른 기법에서는 개체군이 여러 개의 최적해를 찾기 위해 나뉘어지는 데 반하여 이 기법에서는 전체 개체군이 하나의 최적해를 찾기 때문이다. 둘째, 이 방법은 적용하기가 비교적 쉽다. 이 알고리즘은 기존의 유전자 알고리즘에 부가적인 한 과정으로 삽입함으로써 보다 나은 최적화 성능을 기대할 수 있다. 셋째, 이 알고리즘은 성능에 치명적인 영향을 주는 어떤 파라미터도 존재하지 않기 때문에 수치해를 얻을 수 있는 신뢰도가 높다. 넷째, 스키마 이론에 기초를 둔 기본 아이디어 때문에 유전자 알고리즘에 적용하는 데 있어서 아무런 문제를 일으키지 않으며, 따라서 유전자 알고리즘의 장점을 완전히 활용할 수 있다. 이런 특성 때문에 기존의 구속 조건 처리 기법을 적용하는 데 아무런 문제없이 적용할 수 있다. 적대 염색체가 여러 개 존재할 경우 멘델 연산을 어떻게 적용하는 것이 효율적인가는 앞으로의 좋은 연구과제가 될 것이다.

참고문헌

[1] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 1992.

[2] T. Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford, University Press, 1996.

[3] D. J. Cavicchio, "Adaptive search using simulated evolution," *PhD thesis*, Ann Arbor, MI: University of Michigan, 1970.

[4] K. A. De Jong. "An analysis of the behavior of a class of genetic adaptive systems," *PhD thesis*, Ann Arbor, MI: University of Michigan, 1975.

[5] D. E. Goldberg and J. Richardson. "Genetic algorithms with sharing for multimodal function

optimization," *Proceedings of the Second International Conference on Genetic Algorithms*, 1987.

[6] C. Hocaoglu and A. C. Sanderson, "Multimodal function optimization using minimal representation size clustering and its application to planning multipaths," *Evolutionary Computation*, vol. 5, 1997.

[7] S. Tsutsui, A. Ghosh, and Y. Fujimoto, "Forking genetic algorithms: GAs with search space division schemes," *Evolutionary Computation*, vol. 5, 1997.

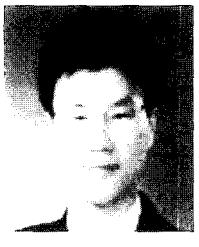
[8] D. Beasley, D. R. Bull, and R. R. Martin, "A sequential niche technique for multimodal function optimization," *Evolutionary Computation*, vol. 1, 1993.

[9] I. S. Song, H. W. Woo, and M. J. Tahk, "A genetic algorithm with a mendel operator for global optimization," *Proceedings of Congress on Evolutionary Computation*, vol. 2, 1999.

[10] M. Schoenauer and S. Xanthakis, "Constrained GA optimization," *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993.

[11] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary Computation*, vol. 4, 1996.

[12] D. Powell and M. Skolnick, "Using genetic algorihtms in engineering design optimization with non-linear constraints," *Proceedings of the Fifth Interational Conference on Genetic Algorithms*, 1993.



송 인 수

1998년 KAIST 항공우주공학과 졸업. 동대학원 석사(2000), 2000년-현재 JC Entertainment 연구원.

탁 민 제

1976년 서울대학교 항공공학과 졸업. 1983년 Univ. of Texas at Austin 석사. 1986년 Univ. of Texas at Austin 박사. 1976년-1981년 국방과학연구소 연구원. 1986년-1989년 미국 Integrated Systems, Research Scientist. 1989년-현재 KAIST 항공우주공학과 교수.

심 재 완

2000년 KAIST 항공우주공학과 졸업. 동대학원 석사과정 중퇴, 현재 Ecole de Polytechnigue 재학중.