국정밀공학회지 제20권 제10호 (2003년 10월)
Journal of the Korean Society of Precision Engineering Vol. 20, No. 10, October 2003.

전 영역 그레이코드 유전자 알고리듬의 효율성 증대에 관한 연구

이원창#, 성활경*

A Study on Computational Efficiency Enhancement by Using Full Gray Code Genetic Algorithm

Won Chang Lee#, Hwal Gyeong Seong*

ABSTRACT

Genetic algorithm (GA), which has a powerful searching ability and is comparatively easy to use and also to apply, is in the spotlight in the field of the optimization for mechanical systems these days. However, it also contains some problems of slow convergence and low efficiency caused by a huge amount of repetitive computation. To improve the processing efficiency of repetitive computation, some papers have proposed paralleled GA these days. There are some cases that mention the use of gray code or suggest using gray code partially in GA to raise its slow convergence. Gray code is an encoding of numbers so that adjacent numbers have a single digit differing by 1. A binary gray code with n digits corresponds to a hamiltonian path on an n-dimensional hypercube (including direction reversals). The term gray code is open used to refer to a reflected code, or more specifically still, the binary reflected gray code. However, according to proposed reports, gray code GA has lower convergence about 10~20% comparing with binary code GA without presenting any results. This study proposes new Full gray code GA (FGGA) applying a gray code throughout all basic operation fields of GA , which has a good data processing ability to improve the slow convergence of binary code GA.

Key Words : GA(유전자 알고리듬), Binary code(2진코드), Gray code(비가중 코드일종), FGGA(Full gray code GA: 전영역 그레이 코드 유전자 알고리듬)

1. 서론

기계 구조물의 최적해를 구하는 문제에서 널리 사용되고 있는 설계 민감도 이론[1]을 대체할 이론으로 유전자 알고리듬[2]이 각광받고 있다. 유전자 알고리듬은 Zeroth-order methods[3] 최적화 기법중 하나로 Zeroth-order methods는 설계 문제의 최적화시 목적 함수의 설계 변수에 대한 도함수를 사용하는 것이 아니라 목적 함수값을 바로 이용하는 방법으로 알고리듬이 비교적 간단하며 목적 함수가 연속적이며 미분 가능해야 할 필요가 없음으로 광범위한 설계문제에 쉽게 적

접수일: 2002년 11월 25일; 게재승인일: 2003년 8월 14일
교신저자: 창원대학교 기계공학과 대학원
E-mail: rainman@sccr.changwon.ac.kr, Tel: (055) 263-4956
* 창원대학교 기계공학과

용할 수 있다. 또한 설계치를 가지는 다수개의 유전자 개체를 이용하므로 전역최적해를 구할 확률이 높다.

일반적으로 유전자 알고리듬은 2진코드를 이용하여 예상 최적 설계 모델인 개체를 다량으로 생성하고 주어지는 넓은 탐색공간에서 각 개체를 해석, 평가후 우수한 개체를 선택하고 교배와 돌연변이등의 조작을 통하여 적합도가 우수한 개체로 진화하여 최적 설계값을 가지는 설계 모델을 찾아 간다. 이때, 수 많은 개체를 해석하기 위해서는 막대한 반복적인 계산량을 요구하는 저효율성[4]과 최적 수렴값 부근에서 최적치를 바로 찾지 못하고 상당한 시간을 최적치 부근의 값들을 반복적으로 답습하는 비효율성[5]을 가지고 있다. 이러한 문제는 설계변수가 많아질수록 커지며 심한 경우는 최적치를 찾지 못하는 경우도 발생한다.

최근에 이러한 유전자 알고리듬이 가지는 단점을 극복하고자 하는 노력이 활발해 지고 있다. 먼저 막대한 계산량을 요구함으로 발생하는 저효율성은 유전자 알고리듬의 병렬화 처리기법[4]으로 극복할 수 있는 방향이 제시 되고 있다.

일반적으로 유전자 알고리듬의 기본적인 조작을 위해 유전자 생성,교배,돌연변이 조작을 가하는데 있어 2진코드(Binary code)를 사용하고 있다. 2진코드는 자리에 따른 특정값을 가지는 방식으로 비트(bit) 변화에 따른 실제 설계변수의 연속적인 변화를 반영하는데 있어 효율이 떨어짐으로써 국소해에 수렴한 후 탈출이 어렵고 탈출하기까지의 시간도 많이 소모 되는 비효율성을 가지고 있다. 이러한 비효율성을 피하기 위하여 디코딩(decoding)이 필요없는 Continuous Parameter GA[5]나 자료처리의 연속성이 우수한 그레이코드(Gray code)를 이용하는 것[6]이 제안되기도 하였다. 하지만, 어디에서 어떤 부분에 이용할 것인가하는 문제와 적용하였을때의 효용성에 대해서는 아무런 결과가 제시되고 있지 않다. 또한 2진코드를 기반으로 부분적으로 그레이코드를 사용을 제안[5]한 사람들은 통상 약 10~20% 정도 느린 수렴 속도를 보인다고 말하고 있을뿐 이 또한 정확한 적용 사례와 결과를 제시하지 않고 있다.

본 연구에서는 유전자 알고리듬의 유전자를 연속적인 변화가 결여된 2진코드를 사용하므로써 발생되는 수렴의 비효율성을 개선하고자 연속적인 변화를 나타내는데 우수한 그레이코드를 도입하고 기존 연구에서 부분적으로 도입하여 수렴속도가 더욱 느려진다는 의견을 보완하기 위하여 유전자 알고리듬의 최적화 전반에 걸쳐 그레이 코드를 적용한 Fully gray code GA(FGGA)를 제안하며 FGGA를 도입하여 유전자 알고리듬의 수렴 효율을 높이고자 한다.

2. 그레이 코드 유전자 알고리듬

유전자 알고리듬은 자연계의 적자 생존법칙을 응용한 탐색 알고리듬으로써 초기에 적당한 크기의 모집단을 생성시킨 후 모집단에 속한 각 개체에 대하여 주어진 환경에 대한 적합도(fitness)를 평가함으로써 개체의 진화형태와 모집단의 진화방향이 결정되며 적합도가 우수한 개체를 기준으로 진화를 거듭하여 최적치를 찾는 알고리듬이다.[7]

2.1 2진코드(Binary code)

2진 코드[8]는 비트로 표현되면 n 비트는 2^n개의 요소를 표현할 수 있다. 2^n개의 요소에 코드를 부여하는 방법은 2^n개의 요소에 0부터 차례대로 2^n-1까지 코드를 부여함으로써 가능하다. 그러나, 요소의 개수가 2의 멱수의 배수가 되지 않으면 비트 조합의 일부는 할당되지 아니할 수 있다. 예를 들면 BCD(Binary coded decimal)코드는 10개의 숫자를 표시하기 위해 4비트가 필요하지만 이중 6개의 코드는 사용하지 않고 있다.

2.2 그레이 코드(Gray code)

그레이 코드[8]는 일정한 자리값이 없는 코드로 한 수에서 다음 수로 진행하는데 있어 인접 코드간 오직 하나의 비트만이 변화하도록 비트를 부여한 코드이다. 이러한 특성으로 인해 음성이나 영상의 경우에 발생되는 아날로그 신호를 컴퓨터가 처리하기 위한 디지털신호로 변환하거나 I/O(input/output) 장치를 제어하는 코드로 사용되어져 왔다.

예를 들면 I/O 장치로부터 입력되는 신호의 크기가 7에서 8로 변할 때 BC(binary code)에서는 0111 -> 1000 로 네 개의 비트가 한 번에 모두 변한다. 이런 경우 제일 좌측의 비트가 변화하는 시간이 다른 비트보다 느리다면 0000으로 인식될 가능성이 높다. 따라서, 이러한 경우 그레이 코드를 사용하면 유리하다. 즉, 그레이 코드에서는 같은 변화

가 0100 -> 1100 으로 한 비트의 변화만으로 가능하기 때문이다. 이러한 연속적으로 변화하는 성질을 이용하면 유전자 알고리듬을 이용한 기계 구조물의 최적 설계 문제에서 인접한 두 설계치의 경계부분에 인접한 그레이 코드를 할당하여 두 부분을 구분 짓는 경계선에서 탐지를 행하면 2진코드를 쓸 경우 보다 훨씬 모호성이 감소한다.

그레이 코드를 얻기 위해서는 어느 비트 조합으로 부터도 시작할 수 있다. 원하는 형태로 0을 1로, 또는 1을 0으로 오직 한 비트만 바꾸어서 그 다음 비트 조합을 얻으면서 진행 시켜 가면 된다. 단, 이렇게 만든 코드 중에서 2개 이상의 같은 수가 있어서는 안된다.

2.3 그레이 코드 적용예

그레이 코드를 사용한 유전자 알고리듬[5]은 유전자 생성, 교차, 돌연변이 등의 기본적인 조작은 2진코드를 사용한 유전자 알고리듬과 동일하다. 하지만, 처음 생성하는 비트형 유전자를 그레이 코드로 생성 시키고 이를 실제 설계변수값으로 바꿀때 생성된 그레이 코드를 이용하기 때문에 2진 코드형 유전자의 설계변수 변환값과 차이가 발생한다. 그리고, 해석후 평가된 평가값을 기준으로 교차에 참여할 개체를 결정하는 부분도 기존의 유전자 알고리듬과 동일하지만 평가된 값을 기준으로 교차를 실시할 때 그레이 코드를 사용하기 때문에 설계변수값의 연속적인 탐색이 용이해 지는 것이다. 게다가 돌연변이에 의한 유전자 비트 변이시 오직 하나의 비트를 바꾸어서 그 다음값을 표시하는 그레이 코드의 특성상 기존의 2진코드에서의 돌연변이 조작보다 더욱 용이하게 다양한 탐색이 가능하다.

간단하게 3비트를 사용한 2진코드와 그레이 코드를 이용하여 3에서 4를 탐색하기 위한 가능성을 고려해보자.[5] Table 1과 같이 2진 코드에서는 011에서 100로 총 3비트 전부가 동시에 바뀌어야만 4를 탐색할 수가 있다. 하지만 그레이 코드에서는 3을 나타내는 010에서 4를 나타내기 위해서는 110으로 처음의 1개 비트만을 바꾸어서 4를 탐색해 낼 수 있다.

이는 여러 개의 설계변수들이 3과 4처럼 연속적인 변화를 할 때 교차나 돌연변이 조작을 가하여 새로운 유전자를 생성하여 다음 세대를 구성할 때 보다 다양하고 연속적인 탐색값을 가지게 하는 것이 가능하다.

Table 1 Binary and gray code representation of integers 1 to 7

Integer	Binary Code	Gray Code
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

2.4 전영역 그레이코드 유전자 알고리듬 (FGGA)

전영역 그레이 코드 유전자 알고리듬(FGGA)는 개체를 구성하는 유전자를 그레이 코드를 이용하여 생성하며 디코딩 및 복제, 돌연변이등의 유전자 알고리듬의 모든 조작 전반에 걸쳐 오직 그레이 코드만을 이용하도록 구성된 유전자알고리듬이다. 이러한 점이 기존에 그레이 코드를 제안 했던 유전자 알고리듬[5]과의 차이점이다.

3. 전 영역 그레이코드 유전자 알고리듬 (FGGA)에 의한 설계 최적화

작성한 유전자 알고리듬을 3개의 설계 변수를 가지는 비교적 간단한 이상적인 코일 스프링 최적화 문제와 좀 더 실제적이며 복잡한 7개의 설계 변수를 가지는 3차원 25요소 트러스 구조물의 최적화 문제에 적용하고 2진코드를 사용한 최적화 결과와 비교 검토하였다.

3.1 스프링의 설계 최적화

최적화 문제로 Fig. 1과 같이 인장력을 받고 있는 코일 스프링을 고려해 보자. 목적 함수로는 코일 스프링의 총 질량이 사용되었으며 구속조건은 하중 P가 작용할 때 스프링의 변형이 δ 보다 작아야 하며 재료의 파괴를 막기위해 철사의 전단응력은 τ_{max} 보다 크면 안된다.

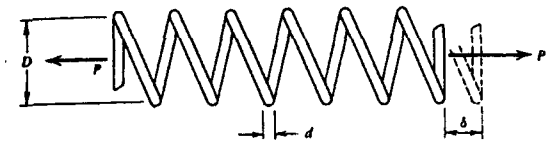

Fig. 1 A coil spring

또한 동하중의 적용을 위해 스프링의 surge frequency는 ω Hz 보다 작아야 하며, 마지막으로 장착공간을 고려하여 D_a 보다 스프링의 외경이 크면 안된다.[6] 즉,

$$\text{Minimize } f(\vec{x}) = \frac{(N+Q)\pi^2 Dd^2\rho g}{4} \quad (1)$$

Subject to

$$g_1 = \delta - \frac{8PD^3N}{d^4G} \le 0 \quad (2)$$

$$g_2 = \frac{8PD}{\pi d^3}\left[\frac{4D-d}{4D-4d} + \frac{0.615d}{D}\right] - \tau_{\max} \le 0 \quad (3)$$

$$g_3 \equiv \omega - \frac{d}{2\pi D^2 N}\sqrt{\frac{G}{2\rho}} \le 0 \quad (4)$$

$$g_4 \equiv D + d - D_a \le 0 \quad (5)$$

(단, $D^L \le D \le D^U$ $d^L \le d \le d^U$ $N^L \le N \le N^U$)

여기서,

$$P = K\delta \quad (6)$$

$$K = \frac{d^4G}{8D^3N} \quad (7)$$

$$k = \frac{4c-1}{4c-4} + \frac{0.615}{c} \quad (8)$$

$$\tau = \frac{8kPD}{\pi d^3} \quad (9)$$

$$\omega_f = \frac{d}{2\pi D^2 N}\sqrt{\frac{G}{2\rho}} \quad (10)$$

이며 사용된 기호의 표기는 Table 2[6]와 같다.

코일 스프링의 최적화 문제는 유전자 알고리듬을 적용시키기 위하여 1차적으로 unconstrained optimization problem 으로 변환 시킨다.

$$F' = f + r\left[\sum_{j=1}^{4}(g_j^2)\right] \quad (11)$$

Table 2 Notation for the coil spring

Notation	Explanation
d	wire diameter
D	mean coil diameter
N	number of active coils
Q	number of inactive end coils
E	Young's modulus
G	shear modulus
P	applied load
K	spring constant
δ	deflection along spring axis
k	Wahl stress concentration factor
τ	shear stress
$\tau_{\max}$	allowable shear stress
ω_f	natural frequency of surge wave in hertz(Hz)
ω	allowable surge frequency of the spring
ρ	mass density of spring material
g	gravitational acceleration
c	D / d
D_a	allowable spring outer diameter

여기서, r은 벌칙 paramater이고, g_j 함수는 아래와 같이 정의된다.

$$\{g_j\} = \begin{cases} 0, & g_j \le 0 \\ g_j, & g_j > 0 \end{cases} \quad (12)$$

(단, j=1,2,3,4)

F'은 최적화가 될 수록 작은 값을 취하게 되나, 유전자 알고리듬에서 판단의 기준이 되는 적응도(fitness)는 최적화 될수록 크게 된다. 따라서, 다음과 같이 적응도를 정의할 수 있다.

$$F = \frac{C_1}{C_2 + F'} \quad (13)$$

단, C_1 및 C_2는 적응도의 크기를 조절하기 위해 사용되는 상수이며 본 연구에서 C_1으로는 1이 사용되었고 C_2로는 10^{-6} 이 사용되었다. 하중 및 물성(material property)으로는 Table 3[6]이 사용되었다.

Table 3 Design data for a coil spring

notation and data	explanation
$P=68.95$ kPa	spring load
$Q=2.0$ times	number of inactive coils
$G=79.23$ GPa	shear modulus
$\delta=0.0127$ m	boundary for deformation
$\tau_{max}=551.6$ MPa	allowable shear stress
$\omega=100.0Hz$	bound for natural frequency
$D_a=0.0381$ m	packaging condition

코일 스프링은 Table 4와 같이 3개의 설계 변수를 정의하고 10비트(bit)로 배열(string)시켜 총 30비트의 배열을 1개의 개체로 하는 2진 코드를 이용한 유전자 알고리듬 문제와 FGGA로 표현한 문제로 각 각 구성하였다. 이 경우 각 변수에서 선택될 설계변수는 Table 4와 같다.

Table 4 Strings assigned to design variable and upper, lower bound

Design variable	Lower bound	Upper bound	String (bit)
D(m)	0.00635	0.03302	10
d(m)	0.00127	0.00508	10
N(times)	2.0	15.0	10
Total			30 bits

초기 모집단의 크기는 난수 발생기(random number generator)에 의해 생성되며 모집단의 크기는 30으로 하였다. 유전자 연산을 위해 교차확률(crossover probability)은 0.85로서 2점 교차를 사용하였으며 돌연변이 확률(Mutation probability)은 0.1로 하여 돌연변이 유전자가 생성되도록 하였다. 위의 확률값들은 통상적으로 사용되어지는 범위 안에서 몇 번의 예비 테스트를 통하여 선택된 값들이다. 이 조건은 다음에 나오는 25요소 3차원 트러스 구조물에서도 적용된다.

코일 스프링의 최적화 문제에서 600세대를 연산 시켜 나온 결과중 적응도 함수와 구조물의 무게를 구한 결과는 Fig. 2 및 Fig. 3과 같다. 2진 코드를 이용한 유전자 알고리듬은 228세대에서 가장 우수한 유전자의 적응도는 78.145를 기록했으며 600세대까지 진행 시켰을때 가장 우수한 유전자의 적응도는 78.471로 큰 변화가 없음을 알 수 있다. 하지만, 그레이 코드를 적용한 유전자 알고리듬의 결과는 139세대에서 가장 우수한 유전자의 적응도는 99.061이었으며 600세대까지 진행 시킨 결과 가장 우수한 유전자의 적응도는 102.635까지 높아지는 것을 알 수가 있다. 목적 함수인 구조물의 총무게는 Fig. 3과 같다.

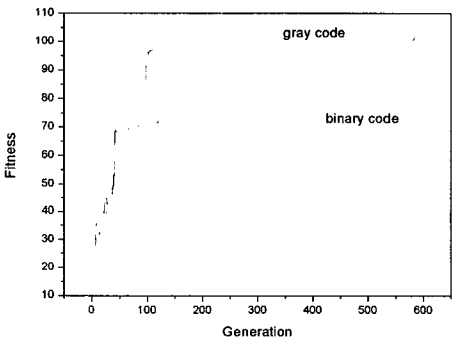

Fig. 2 Generaion-history of "most fitness" design for a coil spring

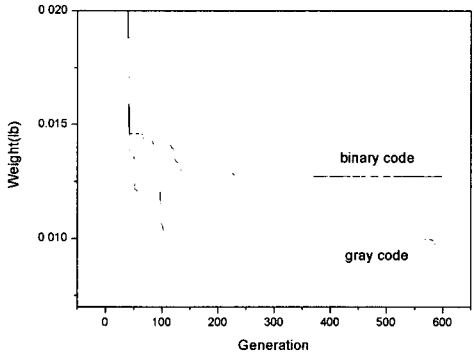

Fig. 3 Generaion-history of "most weight" design for a coil spring

최적화에 도달 했을때의 2진코드와 그레이 코드를 사용한 유전자 알고리듬의 설계 변수의 최적치는 Table 5와 같다.

Table 5 Comparison of topology results after 600 generations

Design variable	Optimum topology	
	Binary code GA	Gray code GA
d (m)	1.7526×10^{-3}	1.4986×10^{-3}
D (m)	2.3901×10^{-2}	1.4046×10^{-2}
N (times)	2.0	5.164
Weight(kg)	5.783×10^{-3}	4.421×10^{-3}

3.2 트러스 구조물 최적화

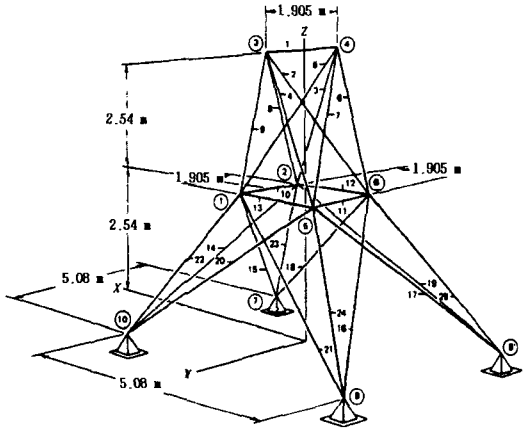

Fig. 4 Twenty-five member transmission tower mode

25요소 트러스 구조물(twenty-five member transmission tower)에 대한 예제는 Fig. 4[3]의 구조를 모델로 하였으며 목적함수로는 구조물의 총질량이 사용되었다. 제한조건식은 구조물에 정하중이 작용될 때의 변위량(displacement)과 응력(stress)으로 하였다. 목적함수식과 구속조건식 다음과 같다.[6]

Minimize

$$f(\vec{x}) = \sum_i \rho A_i L_i \tag{14}$$

Subject to

$$\begin{aligned} g_{j1}(\vec{x}) &= |u_j| - \overline{u_j} \le 0 \\ g_{j2}(\vec{x}) &= |v_j| - \overline{v_j} \le 0 \\ g_{j3}(\vec{x}) &= |w_j| - \overline{w_j} \le 0 \end{aligned} \tag{15}$$

$$g_i(\vec{x}) = |\sigma_i| - \overline{\sigma_i} \le 0 \tag{16}$$

이다.

이때 u_j, v_j, w_j는 j 번째 절점에서의 각 방향 변위이며 $\overline{u_j}, \overline{v_j}, \overline{w_j}$ 는 각 방향 변위의 상한치이다. 그리고, σ_i는 i 번째 부재에서의 응력을 $\overline{\sigma_i}$는 i 번째 부재의 응력 상한치이다. 또한 ρ는 재질의 밀도이고, A_i 및 L_i 는 i번째 부재의 단면적과 길이이다.

위의 최적화 문제는 유전자 알고리듬을 적용시키기 위하여 1차적으로 *unconstrained optimization* problem 으로 변환 시킨다.

$$F' = f + r\left[\sum_{j=1}^{n}(g_{j1}^2 + g_{j2}^2 + g_{j3}^2) + \sum_{i=1}^{m}(g_i^2)\right] \tag{17}$$

여기서, r은 벌칙 paramater이고, $\{g\}$함수는 아래와 같이 정의된다.

$$\{g\} = \begin{cases} 0, & g \le 0 \\ g, & g > 0 \end{cases} \tag{18}$$

F'은 최적화가 될 수록 작은 값을 취하게 되나, 유전자 알고리듬에서 판단의 기준이 되는 적응도(fitness)는 최적화 될수록 크게 된다. 따라서,다음과 같이 적응도가 정의된다.

$$F = \frac{C_1}{C_2 + F'} \tag{19}$$

여기서 C_1, C_2는 적응도의 크기를 조절하기 위해 사용되는 0보다 큰 상수이며 본 연구에서 C_1으로는 10^5이 사용되었고 C_2로는 10^{-4}이 사용되었다. 하중 및 물성(material property)으로는 Table 6[6]이 사용되었다.

25개의 트러스 부재는 Table 7과 같이 7개의 그룹을 형성하고 각 그룹을 11비트(bit)로 배열(string)시켜 총 77비트의 배열을 1개의 개체로 하는 2진코드를 이용한 유전자 알고리듬 문제와 FGGA로 표현한 문제로 각 각 구성하였다. 이 경우 각 그룹에서 선택될 설계변수는 Table 8[6]에서와 같이 0.06451×10^{-3}부터 3.2258×10^{-3}사이의 연속적인 값으로부터 선택 된다.

Table 6 Design data for 25-member transmission tower

Load condition (kN)	Node	Direction of load		
		X	Y	Z
step 1	1	2.225	0	0
	2	2.225	0	0
	3	4.450	44.5	-22.25
	4	0	44.5	-22.25
step 2	3	0	89.0	-22.25
	4	0	-89.0	-22.25
Modulus of elasticity = 68.95 GPa Material density = 27.155 kN/m^3 Upper limit on cross-sectional areas = None Lower limit on cross-sectional areas = 6.4516×10^{-6} m^2 Displacement limits at all nodes and in all directions = 0.00889 m				

Table 7 Strings assigned to design group

Design group	Design variables	string (bit)
1	x_1	11
2	x_2, x_3, x_4, x_5	11
3	x_6, x_7, x_8, x_9	11
4	x_{10}, x_{11}, x_{12}, x_{13}	11
5	x_{14}, x_{15}, x_{16}, x_{17}	11
6	x_{18}, x_{19}, x_{20}, x_{21}	11
7	x_{22}, x_{23}, x_{24}, x_{25}	11
Total		77 bits

특히 초기 모집단의 크기는 난수 발생기(random number generator)에 의해 생성되며 모집단의 크기는 200으로 하였다.

유전자 연산을 위해 교차확률(crossover probability)은 0.85 로서 2점 교차를 사용하였으며 돌연변이 확률(Mutation probability)은 0.1로 하여 돌연변이 유전자가 생성되도록 하였다. 위의 확률값들은 통상적으로 사용되어지는 범위 안에서 몇 번의 예비 테스트를 통하여 선택된 값들이다.

Table 8 Selectable sizing variables(cross-sectional area) for each topology

Topology (x)	Cross-sectional area, (×10^{-3} m^2)						
	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7
1	0.06451 ~ 3.2258						
string (bit)	11	11	11	11	11	11	11

25요소 3차원 트러스 구조물의 최적화를 위해서는 400 세대를 연산을 하였으며 2진코드를 사용한 유전자 알고리듬과 FGGA를 사용하였을때의 적합도와 목적함수를 비교해 보면 Fig. 5 및 Fig. 6과 같다. 코일 스프링문제와 마찬 가지로 FGGA가 보다 나은 결과를 보여 주고 있음을 알 수 있다.

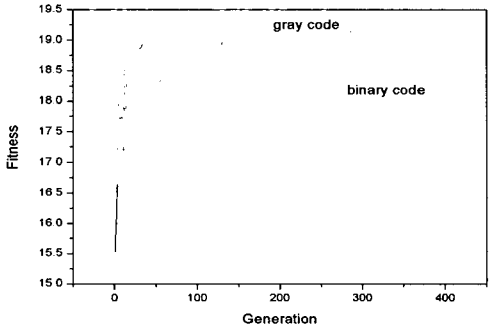

Fig. 5 Generaion-history of "most fitness" design for twenty-five member transmission tower

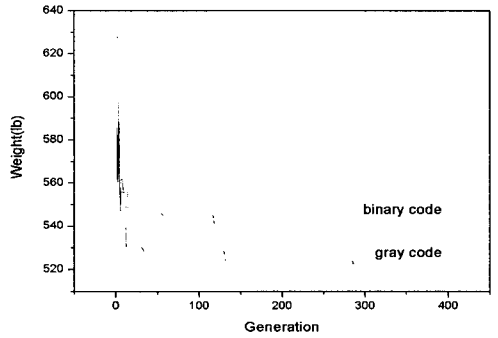

Fig. 6 Generaion-history of "most weight" design for twenty-five member transmission tower

최적치에 도달했을때의 설계 변수는 Table 9와 같다. 2진코드를 사용하였을 경우에는 적응도는 18.411에 거쳤으며 총 무게도 245.68 kg에 수렴을 하였다. 그러나, FGGA를 사용하였을 경우에는 적응도는 19.148에 총무게도 237.09 kg으로 더 우수한 전역탐색 능력을 보여 주었다.

Table 9에 다른 일반적인 최적화 기법과 수렴결과를 비교하기 위하여 참고문헌상의 결과를 Ref[6]로 2진 코드 GA 및 FGGA의 수렴결과와 같이 표시하였다.

Table 9 Comparison of topology results after 400 generations

Design group	Optimum topology(m^2)		
	Ref[6]	Binary code GA	FGGA
1	6.4526×10^{-6}	1.755×10^{-3}	0.5081×10^{-3}
2	1.3210×10^{-3}	1.848×10^{-3}	1.670×10^{-3}
3	1.9332×10^{-3}	0.9594×10^{-3}	1.121×10^{-3}
4	6.4516×10^{-6}	0.2203×10^{-3}	0.5002×10^{-3}
5	0.44213×10^{-3}	0.4750×10^{-3}	0.5348×10^{-3}
6	1.04626×10^{-3}	0.1385×10^{-3}	0.007999×10^{-3}
7	1.72335×10^{-3}	2.840×10^{-3}	2.739×10^{-3}
Weight(kg)	247.45	245.68	237.09

4. 결론

본 연구에서는 우수한 전역 검색 기능과 정확한 결과를 가지는 유전자 알고리듬의 효율성을 높이기 위해 기존에 널리 이용되고 있던 2진코드를 이용한 유전자 알고리듬을 연속적인 변화의 추적이 비교적 용이한 그레이 코드를 유전자 알고리듬의 전 영역에 도입한 FGGA로 대체하여 설계 변수가 3개인 비교적 간단한 이상적인 문제와 설계 변수가 7개인 좀 더 복잡한 문제에 적용 시켜본 결과 다음과 같은 결론을 얻을 수 있었다.

1. 기존에 사용되던 2진코드를 이용한 유전자 알고리듬을 전영역 그레이코드 유전자 알고리듬(FGGA)으로 대체함으로써 모든 구속조건을 만족시키면서 기존 방법으로 탐색이 불가능한 부분의 탐색 능력이 대폭 향상됨으로 동일한 시간으로 2진코드 유전자 알고리듬을 사용하여 구한 최적 설계치보다 통상 3~30%의 더 우수한 목적함수값을 얻을 수 있었다.

2. 초기의 30~40 세대 이하에서 개선되는 양상은 2진코드 유전자 알고리듬이 우수 하지만 그 이후에서는 FGGA에 의한 탐색이 더욱 뛰어 난 것을 알 수 있었으며 동일한 시간에 탐색한 결과로 2진코드 유전자 알고리듬이 찾아내는 시간과 비교할 때 후반부에서는 50~100세대이상의 연산 시간을 벌 수 있는 컴퓨팅 효율을 나타낼 수 있으며 무엇보다도 같은 시간에서 2진 코드 유전자 알고리듬으로 찾을 수 없는 영역의 전역 탐색값을 찾을 수 있었다.

본 연구에서는 유전자 알고리듬의 decoding(디코딩)과 교차, 돌연변이 부분에 연속적인 값의 변화를 추적하는 것이 뛰어난 그레이 코드를 도입하여 유전자 알고리듬의 수렴 효율성을 높일 수 있다는 것을 알 수 있었으며, 제안된 FGGA의 탐색 능력은 설계 변수의 개수에 상관없이 뛰어난 수렴 효과를 보여 주고 있다. 이후 초기 세대에서 뛰어난 탐색 능력을 보여주는 2진코드를 이용한 유전자 알고리듬과 중반 이후 세대에서부터 뛰어난 탐색 능력을 보여 주는 FGGA를 조합하여 더욱 뛰어난 탐색 능력과 향상된 수렴 효율성을 가지는 복합적인 GA (유전자 알고리듬)를 개발하고자 한다.

참고문헌

1. Haug, E. J., Choi, K. K. and Komkov, V., "Design Sensitivity Analysis of Structural Systems," Academic press inc., pp. 25-49, 1986.
2. Goldberg, D. E., "Genetic Algorithms in Search Optimization and Machine Learning," Addison-Wesley Publishing Company Inc., pp. 7-10, 1989.
3. Chong, T. H., Lee, J. S., "A Design Method of Gear Trains using a Genetic Algorithm," J. of the KSPE, Vol. 1, No. 1, pp. 62-63, 2000.
4. Lee, W. CH., Ju, J. H. and Seong, H. G., "A Study for Improvement Effect of Paralleled Genetic Algorithm by using Clustering Computer System," J. of the KSPE, Vol. 20, No. 4, pp. 189-196, 2003.
5. Haupt, R. L., Haupt, S. E., "Practical Genetic Algorithm," John Wiley & Sons, pp. 49-65, pp. 88-91, 1998.
6. Haug, E. J., Arora, J. S., "Applied Optimal Design," John Wiley & Sons, pp. 14-16, pp. 245-248, 1979.
7. Baek, W. T., Seong, H. G., "The Optimization of Sizing and Topology Design for Drilling Machine by Genetic Algorithm," J. of the KSPE, Vol. 14, No. 12, pp. 25-27, 1997.
8. Choi, J. H., Jung, J. O., "The Theory of Computer Architecture," Daelim, pp. 126-128, 2000.