분할구조 기반의 다기능 연산 유전자 알고리즘 프로세서의 구현

論 文
52D-5-6

Implementation of GA Processor with Multiple Operators, Based on Subpopulation Architecture

趙 珉 錫* · 鄭 德 鎭**
(Min-Sok Cho · Duck-Jin Chung)

Abstract - In this paper, we proposed a hardware-oriented Genetic Algorithm Processor(GAP) based on subpopulation architecture for high-performance convergence and reducing computation time. The proposed architecture was applied to enhancing population diversity for correspondence to premature convergence. In addition, the crossover operator selection and linear ranking subpop selection were newly employed for efficient exploration. As stochastic search space selection through linear ranking and suitable genetic operator selection with respect to the convergence state of each subpopulation was used, the elapsed time of searching optimal solution was shortened. In the experiments, the computation speed was increased by over 10% compared to survival-based GA and Modified-tournament GA. Especially, increased by over 20% in the multi-modal function. The proposed Subpop GA processor was implemented on FPGA device APEX EP20K600EBC652-3 of AGENT 2000 design kit.

Key Words : 유전자 알고리즘, GAP, subpopulation, 진화형 하드웨어, steady-state model

1. 서 론

유전자 알고리즘(Genetic Algorithm)은 개체집단이 다음 세대의 집단을 형성하는 과정에서 적자생존을 확률적으로 알고리즘화 한 것으로 다윈의 생물진화의 원리인 자연도태의 유전적인 메커니즘에 기초한 탐색알고리즘이며, 그 목적은 세대가 지날수록 주어진 환경에 잘 적응한 개체가 발생한다는 가설에 입각하여 최적개체를 찾는 것이다. 유전자 알고리즘은 두 부모의 유전자로부터 그들 자손의 유전자를 형성하는 유성생식과 자연환경에서 일어나는 진화원리를 흉내 내고 있다. 유전자 알고리즘은 문제에 대한 가능한 해들을 정해진 형태의 자료 구조로 표현한 다음 이들을 선택, 교차, 돌연변이로 이루어진 재생산과정을 반복 수행함으로써 최적의 해를 생성하는 알고리즘이다. 그러나 이러한 유전자 알고리즘은 반복적인 진화과정과 적응과정을 거치게 되며 양질의 해를 얻는 데는 많은 연산시간을 필요하게 된다는 문제점을 가지고 있다. 특히, 실시간 응용 및 환경의 변화에 민감한 응용분야, 빠른 연산처리속도가 필요한 분야, stand-alone으로 동작해야 하는 분야 등에서는 하나의 전용칩으로 인공생명의 여러 모델을 하드웨어로 집적화 하여 구현하는 것은 필수 불가결하다.[1,2] 본 연구에서는 유전자 알고리즘에 있어서의 심각한 문제인 연산 시간을 줄이기 위해 분할 구조를 적용하여 개체의 다양성을 높여 조기 수렴에 보다 효과적으로 대응할 수 있고 다기능 유전연산자를 사용하여 최적해로의 수렴 속도를 효과적으로 향상시킬 수 있는 하드웨어 지향의 수정된 유전자 알고리즘의 제안하고, 제안한 알고리즘의 하드웨어로 구현하고자 한다.

본 연구에서는 하드웨어에 적합한 분할 구조 유전자 알고리즘 프로세서를 위한 별도의 알고리즘을 제안하였다. 고속의 연산처리를 요구하는 시스템 및 실시간 처리가 요구되는 복잡한 최적화 문제들에 적용시키기 위한 목적으로 수렴 속도의 감소와 정확성의 증가를 위한 알고리즘을 고안하였다. 이 알고리즘의 타당성을 검증하기 위해 Dejong Function[3]과 같은 수학적 최적화 문제 및 Set Covering Problem(SCP)[5]과 같은 NP 문제에 적용하여 기존의 속도 지향의 GA인 Survival-based GA[6] 및 Modified tournament GA[7]를 MATLAB을 이용한 시뮬레이션을 통해 비교하여 성능의 우수성을 입증하였다. 또한, 제안된 알고리즘을 VHDL을 이용하여 FPGA device인 APEX EP20K600EBC652-3가 탑재된 AGENT 2000 Design kit 상에서 하드웨어 구현을 통해 소프트웨어 GA의 단점인 연산시간을 크게 줄여 실제 응용 시스템에 적용할 수 있는 가능성을 확인하였다.

다음 2장에서는 제안된 분할구조 유전자 알고리즘의 동작, 특성 및 구조를 설명하고, 3장에서는 제안된 유전자 알고리즘 프로세서의 내부 구조 및 구현 방법, 제안된 프로세서의 성능검증을 위해 사용된 적합도 함수와 구현된 Prototype의 결과 및 분석에 대해 설명하고, 마지막으로 결론을 맺고자 한다.

* 正 會 員 : 仁荷大 電子材料工學科 碩士
** 正 會 員 : 仁荷大 情報通信工學部 敎授 · 工博
接受日字 : 2002年 12月 18日
最終完了 : 2003年 3月 7日

2. 분할구조 유전자 알고리즘

유전자 알고리즘은 많은 반복적인 연산을 수행해서 여러 세대에 걸쳐서 개체를 진화시켜야 하므로 시간이 많이 걸린다는 단점을 가지고 있다. 그리하여 우수한 탐색 능력에도 불구하고 실시간 응용을 위한 문제에 대해서는 보다 효율적으로 최적 해를 탐색할 수 있는 알고리즘의 고안 및 하드웨어 기반의 유전자 알고리즘 프로세서의 필요가 불가결하다.[6,7]

최적해의 탐색에 있어서 수렴 속도에만 중점을 두어 알고리즘을 개발할 경우 특히, 다봉 문제(Multimodal function)에서 국부 최소점(Local minima)에 빠지는 문제가 심각하게 대두될 수 있다. 그러한 경우 개체군은 동일한 개체들이 개체군 전체에 확산 된 경우로 나타낼 수 있으며 더 이상 교차 유전연산자는 탐색에 있어서 의미가 없어진다. 동일한 개체간의 교차를 통한 유전자 조작은 교차라는 유전연산자가 개체간의 동일한 유전자좌 사이에 유전형질만을 교환하는 성질을 갖고 있기 때문이다. 돌연변이 유전연산자에만 의지하여 국부 최소점을 탈출한다는 것은 일반적으로 유전자 알고리즘에서 사용하는 돌연변이율을 고려했을 경우 수많은 반복연산이 필요하게 된다. 그렇다고 너무 큰 돌연변이율을 사용하면 개체의 유전형질을 파괴하여 랜덤 탐색과 같은 성질을 갖게 된다. 이러한 관점에서 유전자 알고리즘의 최적 해의 탐색이 이루어지는 과정 동안 보다 효율적인 진화가 필요하며 이에 본 연구에서는 하드웨어 구현까지 고려한 몇 가지 방법을 고안하여 기존의 유전자 알고리즘의 수정하였다.

2.1 제안한 분할구조 개체군

유전자 알고리즘은 진화 형태에 따라 크게 두 가지로 구분이 된다. 한 세대의 진화 과정이 이루어지는 동안 개체군내의 모든 개체가 선택에 참여해 일제히 자손을 만들어 다음 세대의 진화를 위한 새로운 개체군을 형성하는 Generational model과 한 세대의 진화 과정동안 만들어진 우성의 자손 염색체가 열성의 부모염색체를 대신하여 개체군내에 포함되어 곧바로 다음세대의 진화과정에 참여할 수 있는 Steady-state model로 나뉘어 진다. 전자의 경우 개체군 크기의 두 배만큼의 기억 공간이 필요하기 때문에 하드웨어 기반의 유전자 알고리즘의 구현을 고려한다면 후자가 보다 용이하다.[8,9]

유전자 알고리즘의 가장 큰 특징인 전역 탐색이라는 관점에서 봤을 때 개체군내의 개체의 다양성은 중요한 문제 중 하나이다. 기존의 여러 연구들에서 개체의 다양성을 증가시킬 목적으로 개체의 분화(Species)에 대한 방법들이 제시되어 왔었다. 이러한 연구들은 대부분 Generational model 유전자 알고리즘에 국한되어 왔으며 개체 분화 과정도 개체군내의 모든 개체간의 유사성 비교를 해밍 거리(Hamming distance)를 계산하는 방법을 사용하기 때문에 추가적인 연산시간이 증가되는 단점을 가지고 있다. 개체군의 크기가 p 일 때 각 개체는 자신을 제외한 나머지 개체들과의 유사성 비교가 필요하므로 $O(p^2)$ 의 추가적인 연산시간을 필요하게 된다.[10,11]

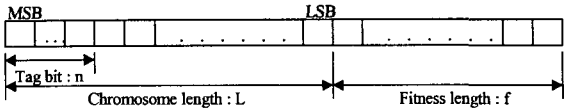


그림 1 제안한 분할구조에서의 개체 및 적합도

Fig. 1 Chromosome & fitness of subpopulation architecture

이에 본 연구에서는 하드웨어 기반의 유전자 알고리즘을 목적으로 개체군에 사용되는 기억 공간은 절약할 수 있는 Steady-state model[8] 형태에 추가적인 연산이 필요하지 않고 개체의 다양성을 증가시킬 수 있는 형태의 분할구조 개체군을 적용하였다. 그림 1은 제안한 분할구조 개체군내에서의 개체와 적합도의 구조를 보여주고 있다. 여기서 Tag bit은 개체군을 몇 개의 분할구조로 나누는지를 결정하는 요소가 되며 각 개체의 상위 유전자좌에 포함되어 있다. Tag bit의 개수가 n 이면 2^n 개로 분할된 개체군을 표현할 수 있으며 전체 개체군의 크기가 p 라면 각 부 개체군의 크기는 $p/2^n$ 이 된다. Tag bit의 개수가 0 이면 개체군이 분할되지 않은 단일 개체군 형태가 된다. 그림 2는 2개의 Tag bits를 사용하여 개체군을 4개의 분할구조로 나눈 형태를 보여주고 있다.[11]

00.....	a 영역
01.....	b 영역
10.....	c 영역
11.....	d 영역

그림 2 분할구조 개체군의 형태

Fig. 2 Subpopulation architecture
(Number of tag bits = 2)

그림 2 와 같은 구조는 전체 해의 탐색 영역을 4개의 탐색 영역으로 각 부 개체군내의 개체들끼리 제한된 결합에 의해 탐색을 수행하므로 단일 구조의 개체군에 비해 개체들의 다양성을 증가시켜 개체들이 하나의 최적의 값만 존중하여 진화하는 현상을 막을 수 있다. 각 부 개체군 영역은 유전자 알고리즘의 개체군이 초기화 과정에서 동일한 크기로 형성되며 각 부 개체군의 최적의 개체들의 적합도의 크기에 따른 확률적 우선순위에 의해 각 세대마다 진화를 수행할 부 개체군을 선택하는 방법을 적용하였다. 이것은 진화가 이루어지는 시점에서 볼 때 가장 최적의 개체를 보유하고 있는 부 개체군에 보다 진화에 우선순위를 둠으로 해서 최적의 해를 찾는 시간을 줄이기 위해서이다.

2.2 개발과 탐색

유전자 알고리즘에서 개체군의 크기는 최적해의 수렴시간

및 수렴성을 결정하는 중요한 파라미터 중 하나이다. 개체군의 크기가 너무 크면 개체의 다양성은 증가하여 최적해를 찾는 정확성은 커질 수 있으나 개체들 간의 좋은 형질을 교환하는 경우의 수가 증가하므로 최적해로의 도달 시간은 길어질 수 있다. 반면에, 개체군의 크기가 너무 작으면 개체들 간의 좋은 형질을 교환하는 경우의 수가 감소하여 최적해의 근접 시간은 단축되지만 해의 다양성 부족으로 최적해에 이르지 못하고 국부 최소점(Local minima)에 빠질 수 있는 확률이 커지게 된다. 기존의 탐색 결과를 최대한 이용하려는 경향 즉 Exploitation과, 새로운 영역을 탐색하려는 성질 즉 Exploration으로 생각한다면 큰 개체군의 경우에는 Exploration이, 작은 개체군은 Exploitation이 우세하다고 표현할 수 있다.

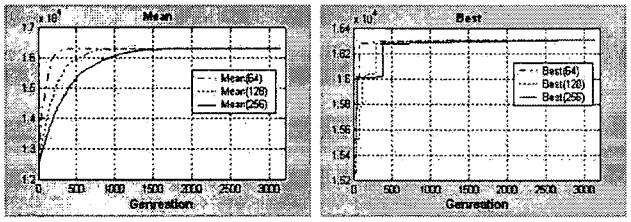


그림 3 개체군의 크기에 따른 평균값 및 최적값

Fig. 3 Mean & best fitness of each population size

개체군의 크기에 따른 실험을 통해 이러한 성질을 볼 수 있는데 그림 3 에서 1000세데 개체군의 평균값 및 최적값을 보면 개체군의 크기 64일 때가 가장 빠르게 진화하는 것을 확인할 수 있다. 이는 개체군의 크기가 작을수록 Exploitation이 우세하기 때문이다.

표 1 개체군의 크기에 따른 수렴세대 (100회 평균)

Table 1 Convergent generation of each subpopulation size

Pop size	xover type	Average (Gene.)	Num of test (Gene.<2000)	Num of test (Gene.>5000)
64	uniform	1607.2	78	10
	2-point	1868.5	70	13
	1-point	1935.1	72	11
128	uniform	1872.8	64	3
	2-point	2211.5	55	7
	1-point	1895.1	63	2
256	uniform	2953.3	7	2
	2-point	2978.4	8	3
	1-point	3029.3	7	3

표 1 은 똑같은 실험을 교차연산자의 종류에 따라 100회 실험한 결과이다. 개체군의 크기가 작을수록 최적해를 2000세대 미만에서 찾는 실험의 횟수는 많아 평균 세대수는 감소할 수 있으나 5000세대를 넘어도 수렴하지 못하는 실험의 횟수가 더 많았다는 것은 Exploitation은 우세하지만 Exploration이 충분히 이루어지지 못했기 때문이다. 최적해의 탐색이 보다 안정적이기 위해서는 개체군의 크기가 어느 정도 이상이어야 한다는 것을 의미한다. 그렇다고 개체군의 크기를 무한정 늘린다는 것보다는 자원 낭비이며 유전자 알고리즘의 근본적인 메커니즘을 벗어나는 것이므로 두 성질이 균형을 이룰 수 있도록 개체군의 크기가 설정되어야 한다. 위 두 실험의 조건은 개체길이는 32bits이고 적합도 함수는 (32X63) SCP를 이용하였다.

본 논문에서는 앞에서 제시한 분할구조를 통해 탐색 영역 공간을 나누어 Exploration을 높이고 유전 연산은 제한된 영역 내에서 수행함으로 해서 동시에 Exploitation을 높일 수 있는 개체군의 구조를 제안하였다.

2.3 선형순위 선택법

각 부 개체군의 최적의 개체들의 적합도의 크기에 따른 확률적 우선순위에 의해 각 세대마다 진화를 수행할 부 개체군을 선택하는 방법에는 선형 순위 선택법을 적용하였고 식 (1)의 방법을 사용하였다.

$$P_s = \frac{1}{N}\left[n_{max} - (n_{max} - n_{min})\frac{r-1}{N-1} \right] \qquad (1)$$

$$(n_{min} = 2 - n_{max} , \ 1 < n_{max} < 2)$$

이것은 유전자 알고리즘에서 Baker가 개체를 선택하는 방법으로 사용하였던 것으로 단일 개체군에서 개체를 선택하는 확률을 개체의 순위에 선형적으로 표현한 방법이다.[12] N 은 개체군의 크기, r 은 개체군내의 개체의 순위를 뜻한다. 보통 이 선택법을 사용한 유전자 알고리즘의 경우 최적해에 가까운 값에는 빠르게 진화할 수 있으나 진화가 어느 정도 진행된 시점에서는 적합도 값이 비슷한 값 사이에도 순위에 입각한 확률을 통해 선택을 수행하므로 선택압력이 증가한다는 단점을 지니고 있다.[13] 하지만, 본 연구에서는 이를 부 개체군을 선택하는 방법으로 사용하였고 부 개체군이 선택되면 다른 선택 방법에 의하여 진화할 개체를 선택하였다. 그러므로 N 은 부 개체군의 개수, r 은 각 부 개체군내의 최적 개체들 간의 적합도 사이에 순위로 사용될 수 있다. n_{max}/N 는 최고 순위(r=1)의 부 개체군의 선택확률이고 n_{min}/N 는 최저 순위(r=N)의 부 개체군의 선택확률이 된다. 그림 4 는 각 부 개체군의 개수가 2, 4, 8의 경우에서 각 부체군의 순위(r)를 매개변수로 에 대한 선택확률의 관계를 표현한 것이다.

본 연구에서는 n_{max} 에 대하여 사용자가 8 가지(1.1, 1.2, 1.3, …, 1.9)의 경우에 대하여 부 개체군의 선택에 대한 압력을 조정할 수 있도록 설계하였으며 하드웨어 기반의 유전자 알고리즘에서는 이 선택에 3 bits를 할당하게 된다.

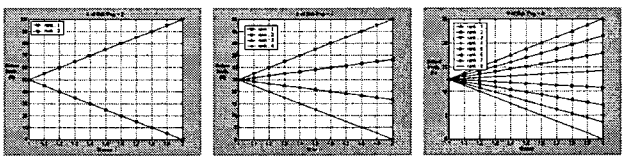


그림 4 부 개체군 순위 선택확률

Fig. 4 Probability of selecting each subpopulation

2.4 유전자 연산자의 특성

유전자 알고리즘에서 사용되는 기본적인 연산자는 교차와 돌연변이이다. 교차는 부모의 형질이 자손에게 적절히 계승되어야 하며 여러 가지 변형이 있으나 단순 교차(Simple crossover), 복수점 교차(Multi-point crossover), 균일 교차(Uniform crossover)등이 주로 이용된다. 돌연변이 연산자는 염색체의 각 유전자좌에 대하여 돌연변이 확률 P_m 에 따라 대립 유전자의 값으로 변경하는 연산자로 잃어버린 유전 형질을 복구하기 위한 2 차적인 연산자로 사용되며, 돌연변이를 너무 큰 변이확률로 설정하면 스키마(schema)가 전부 파괴되어 무작위 탐색으로 변하고 돌연변이가 없는 경우 해의 질의 한계가 드러나기 때문에 적절한 돌연변이 확률의 선택이 중요하다. 여기서 말하는 스키마는 유전자 알고리즘의 수학적 배경을 제시한다는 점에서 매우 중요하며 우성의 성질이 강한 개체들의 정해진 스트링 위치에 같은 비트 값을 가진 모든 가능한 스트링들의 부분집합으로 정의 내릴 수 있다. 유전자 알고리즘에서 스키마는 탐색을 유도하는데 해의 구성부로 작용하여 중요한 역할을 수행하므로 유전자 연산자에 의한 영향을 비교해 본다면 유전자 연산자의 특성을 알아낼 수 있다.[4]

N : 개체군의 크기, l : 개체길이,
$\delta(H)$: 스키마 H의 길이, $O(H)$: 스키마 H의 차수,
$m(H,k)$: k 세대에서 스키마 H의 빈도,
P_c : 교차 확률, P_m : 돌연변이 확률,
$f_{sum}(k)$: k 세대에서 개체군 전체 적합도의 합,
$f_{avg}(H,k)$: k 세대에서 스키마 H와 일치하는 평균 적합도
에서 재생산 및 유전자 연산자에 의한 영향은 아래와 같다.

2.4.1 재생산에 의한 영향

유전자 알고리즘에서 여러 선택 방법이 있지만 본 연구에 사용한 tournament 선택에 의한 재생산 영향은 H 와 일치하는 개체의 선택확률이 $f_{avg}(H,k)/N$ 이므로 재생산 수행 후 예상되는 빈도는

$$m_1(H,k+1) = m(H,k)\cdot N\cdot \frac{f_{avg}(H,k)}{N} = m(H,k)\cdot f_{avg}(H,k) \quad (2)$$

이 되며 tournament 선택이 적합도를 존중한 선택이 아니기 때문에 $f_{sum}(k)$ 와는 무관한 관계를 나타낸다.

2.4.2 교차 연산에 의한 영향

일반적으로 교차점은 1 과 l-1 사이에서 균등하게 발생하므로 교차점이 스키마 내에서 발생할 확률은 $\delta(H)/(l-1)$ 이 되므로 파괴 확률(Disruption probability) P_{dist} 은

$$P_{dist} = P_c\frac{\delta(H)}{l-1} \quad (3)$$

이 된다.

결과적으로 생존 확률(Survival probability) P_{surv} 은

$$P_{surv} = 1 - P_{dist} \geq 1 - P_c\frac{\delta(H)}{l-1} \quad (4)$$

이므로 재생산과 교차 연산이 이루어진 후 빈도는

$$m_2(H,k+1) \geq m_1(H,k+1)\left[1 - P_c\frac{\delta(H)}{l-1}\right] = m(H,k)\cdot f_{avg}(H,k)\left[1 - P_c\frac{\delta(H)}{l-1}\right] \quad (5)$$

이 된다. 이 식은 진화가 진행됨에 따라 길이가 짧은 스키마의 수는 증가함을 보여준다.

2.4.3 돌연변이 연산에 의한 영향

개체군내의 모든 개체들은 같은 돌연변이 확률로 변경시키기 때문에 만일 스키마의 특이점 중 하나가 돌연변이 되면 이로 인해 스키마가 파괴될 것이다. 스트링 내 각 비트가 변경될 확률이 P_m 이므로 비트의 생존 확률은 $1-P_m$ 이 된다. 스키마 H 에 포함된 특이점의 수(차수)가 $O(H)$이므로 스키마의 생존확률은

$$P_{surv} = (1 - P_m)^{O(H)} \simeq 1 - P_m\cdot O(H) \quad (6)$$

이 된다. 일반적으로 $P_m \ll 1$ 이므로 Taylor 급수를 얻고 고차항을 무시하면 생존확률은 식 (6)과 같이 근사화된다. 따라서 재생산, 교차, 돌연변이의 효과를 결합한 예측은

$$m(H,k+1) \geq m_2(H,k+1)[1 - P_m\cdot O(H)] = m_1(H,k+1)\left[1 - P_c\frac{\delta(H)}{l-1}\right][1 - P_m\cdot O(H)] = m(H,k)\cdot f_{avg}(H,k)\left[1 - P_c\frac{\delta(H)}{l-1}\right][1 - P_m\cdot O(H)] \quad (7)$$

가 되며 $P_c\frac{\delta(H)}{l-1}\cdot P_m\cdot O(H) \ll 1$ 이므로 더욱 간략화되어 최종적으로 (8) 과 같은 식으로 정리된다.

$$m(H,k+1) \geq m(H,k)\cdot f_{avg}(H,k)\left[1 - P_c\frac{\delta(H)}{l-1} - P_m\cdot O(H)\right] \quad (8)$$

식 (8) 은 스키마 H 의 정의길이와 차수의 함수로 표시되는데 진화가 진행됨에 따라 정의길이가 짧고 낮은 차수의 스키마는 개체군 내에서 여전히 그 수가 증가하게 된다는 것을 의미하다. 이러한 스키마 이론을 바탕으로 어느 한 해로의 집중이 되는 진화과정을 반복하여 수행하면 정의길이가 짧고 낮은 차수의 스키마들이 결합하여 보다 정의길이가 길고 높은 차수의 새로운 스키마를 발생시켜 그것을 존중하면서 탐색의 과정이 이루어진다.

이러한 관점에서 각 부 개체군들의 수렴 상태에 따른 적절한 유전자 연산자의 사용이 필요하다. 기존의 진화과정으로 높은 적합도의 개체를 보유하고 있는 부 개체군의 경우 스키마가 덜 파괴되면서 Exploitation 위주의 유전자 연산자의 사용이 이루어져야 하고 다른 부 개체군보다 낮은 적합도의 개체를 보유하고 있는 부 개체군의 경우는 어딘가에 존재할 가능성이 있는 보다 높은 적합도의 탐색을 위해 Exploration 위주의 유전자 연산자의 사용이 이루어져야 한다.

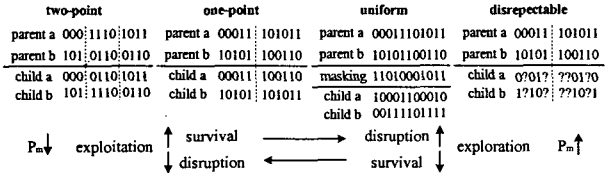


그림 5 유전자 연산자의 특성

Fig. 5 Properties of genetic operators

그림 5 에서는 교차 연산자의 종류와 돌연변이율에 따라 Exploitation 및 Exploration의 두 성질과 관련한 특성을 보여주고 있다. 이것은 스키마 이론 및 기존의 유전자 알고리즘 학자들의 연구 결과를 바탕으로 한 내용이다. 본 논문에서는 이러한 특성을 이용하여 개체군이 부 개체군으로 나누어져 각기 제한된 영역을 탐색의 경우에 각 부 개체군의 최적 값들의 적합도 순위에 따라 다른 교차 연산자를 사용하여 진화를 수행하는 가변적인 유전자 연산자의 선택을 분할구조에 적용하였다. 즉, 부 개체군의 개수가 4 개인 경우 가장 우수한 적합도를 보유한 부 개체군이 진화에 참여할 경우 two-point crossover를 사용하며 가장 낮은 적합도를 보유한 부 개체군은 disrespectable crossover[14]를 사용하게 된다. 부 개체군의 개수가 8 개인 경우는 사용자가 선택하는 2가지 형태의 돌연변이율과도 쌍을 맞추어 사용할 수 있도록 설계하였다.

2.5 제안된 유전자 알고리즘

표 2 에는 본 연구에서 제안한 알고리즘의 의사코드를 보여주고 있다. 개체의 다양성을 높이기 위한 분할구조 개체군의 형성, 각 부 개체군의 진화상태에 따른 탐색할 부 개체군의 선형 선택법 및 그에 따른 유전자 연산자의 결정에 관한 부분이 모두 포함되어 있다. 개체의 선택은 전 세대의 우성 부모를 다시 진화에 참여시키는 modified tournament 방식[7]을 취하였고 이주 모델을 기반으로 하는 병렬화기법[15]을 적용하였다.

표 2 제안된 분할구조 유전자 알고리즘의 의사코드

Table 2 Pseudo-code of Subpopulation GA

```
Procedure Subpop GA( )
Begin
-- Initialize Population and Best and Sub_best
   Sub_num = 2^tag;   Subpop_size = N_pop/ Sub_num;
   For i = 0 to Sub_num -1 do
      For j = 0 to Subpop_size do
         Chrom = tag(i) & Random(Nchrom-tag);
         Fitness = Fitness (Chrom);
         Population(ixSubpop_size+j)=Chrom&Fitness;
         If  Sub_best_fit(i) < Fitness then
            Sub_best(i) = Chrom & Fitness;
         End if;
      End For;
      If  Best_fit < Sub_best_fit(i)  then
         Best = Sub_best(i);
      End if;
   End For;
-- Start Genetic Algorithm
   While evalution(Best_chrom) do
      Rank = Ranking(Sub_best, Sub_num);
      Sel_pop = Linear_rank(Rank, Nmax);
-- Select chromosomes and decide the parent_A,B
      Space = Sel_pop x Subpop_size;
      Chrom_A = select(Population(Space+RNG(N_subpop_size));
      Chrom_B = Select(Population(Space+RNG(N_subpop_size));
      Parent_A = Max. of Fit (chrom_A, chrom_B);
      Parent_B = previous Parent_A(Sel_pop);
      Worse_addr = Min. of Fit (chrom_A_addr, chrom_A_addr);
      Worse_fit = Min. of Fit (chrom_A_fit, chrom_B_fit);
      Worse_data = Worse_addr concat Worse_fit;
--Crossover and Mutation and evaluate a fitness
      Offspring(A,B)
      = Crossover(Parent_A,Parent_B,Rank(Sel_pop));
      Offspring(A,B)
      = Mutation(offspring(A,B),Rank(Sel_pop));
      Offspring_fit(A,B) = Fitness(offspring(A,B));
-- Update a population after accept data from other GA
      New_chrom = Max. of Fitness(offspring_fit(A,B));
      Migr_out = Max. of Fitness (New_chrom, Sel_pop);
      Migr_chrom_in = Migrator(other GA, Sel_pop);
      If New_chrom_fit > Worse_fit then
         If New_chrom_fit > Migr_chrom_in_fit
         then Population(Worse_adrs) = New_chrom;
         Else Population(Worse_adrs) = Migr_chrom_in;
         End if;
      Else
         If Worse_fit > Migr_chrom_in_fit
         then Population(Worse_adrs) = Worse_data;
         Else Population(worse_adrs) = Migr_chrom_in;
         End if;
      End if;
-- Update Best & Sub_best
      If New_chrom_fit > Sub_best(Sel_pop)
      then Sub_best(Sel_pop) = New_chrom;
      End if;
      If New_chrom_fit > Best_chrom_fit
      then Best_chrom = New_chrom;
      End if;
   End While;
End Procedure;
```

2.6 적합도 함수와 실험결과

목적함수(object function) 즉 최적화(최대화 또는 최소화)하고자 하는 함수는 각 개체의 적합도를 평가하는 기반이다. 그러나 목적함수 값의 범위는 문제마다 다르기 때문에 보통 정해진 구간 사이의 양수 값을 갖도록 표준화된 값을 적합도로 사용한다. 이러한 적합도 함수는 복잡도에 따라 유전자 알고리즘의 성능을 평가하는 기준이 되기도 한다. 해의 공간이 크고 극값의 개수가 많을수록 적합도 함수는 복잡해지며 그에 따라 탐색의 시간도 길어진다. 유전자 알고리즘은 미지의 공간에서 최적해를 찾는 것이 목적이므로 복잡도가 높은 적합도 함수에 대해서도 충분히 고려가 되어야 한다. 그러나 적합도 함수의 복잡도가 낮은 경우에는 알고리즘의 성능평가에서 상대적인 차이를 관측하기가 어렵다.

본 연구에서 제안한 분할구조 유전자 알고리즘의 경우 적합도 함수의 복잡도가 높을 경우에 더욱 성능의 차이가 현저히 나타난다. 실험에 적용한 적합도 함수는 무수히 많은 다봉 문제(Multimodal function)를 선택하여 성능을 측정하였고 실제 이용한 적합도 함수는 아래와 같다.

2.5.1 수학적 최적화 문제

첫 번째 적합도 함수는 식 (9) 와 같은 최적화 함수에 대해서 $0 \le x \le 12.1$ 와 $4.1 \le y \le 5.8$ 의 범위에서 성능을 검증하였다.[3]

$$f(x, y) = 21.5 + x\sin(4\pi x) + y\sin(20\pi y) \qquad (9)$$

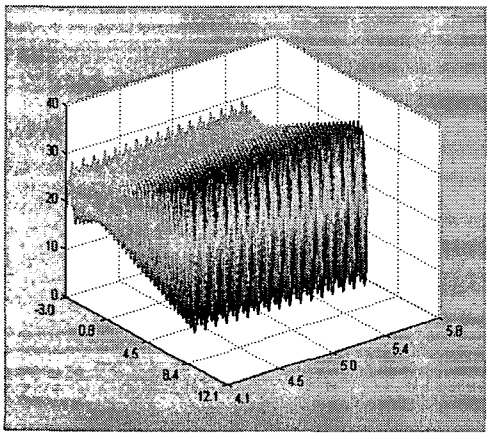


그림 6 수학적 최적화 함수
Fig. 6 Mathematical func. (Dejong)

실제 적합도 함수의 조망(Landscape)은 그림 6 과 같은 형태이며 많은 국부적 수렴지점을 지닌 매우 다루기 힘든 특징을 가지고 있기 때문에 최적의 해를 찾아내기 어렵다. 정밀도는 x 와 y 각각 10^{-4}로 x 에 대해 18 bits를 y 에 대해서는 13 bits의 개체길이를 할당하여 전체 실수 해 공간을 31bits로 표현하여 실험하였다.

2.5.2 SCP (Set covering Problem:32x63)

두 번째 적합도 함수는 잘 알려진 조합의 최적화문제로서 일반적으로 문제 푸는 방법이 존재하지 않는 NP 문제의 일종인 non-unicost set covering problem[5]이다. Set covering problem은 zero-one 행렬의 rows를 최소의 비용의 columns를 이용하여 covering 하는 문제이다. 이는 주로 승무원 스케줄링, 비상 설비의 위치 결정, 조립 라인 평형 및 boolean 표현 간략화 등과 같은 많은 실제적인 응용문제에 적용된다.

함수의 복잡도를 높이기 위하여 128(0~127)개의 minterm 중 임의로 63개의 minterm을 뽑은 식 (10) 을 사용하여 32X63 행렬에 대한 SCP 함수를 제작하였다.

$$\begin{aligned} f(a,b,c,d,e,f,e) = \sum m(&0,2,4,5,6,7,10,12,17,\\ &18,20,21,24,25,26,27,28,34,36,\\ &37,38,39,40,42,43,44,49,50,52,\\ &54,56,57,60,65,67,68,69,70,71,\\ &73,75,81,84,85,87,88,90,91,92,\\ &97,101,102,103,104,108,112,113,\\ &116,120,121,124) \end{aligned} \qquad (10)$$

여기서 rows 는 17개의 EPI(Essential prime implicant)와 15개의 PI(Prime implicant)가 그림 7 과 같이 뒤섞어 배치되어 있고 columns은 식에서 선택된 minterm으로 구성되어 있으며 최소의 Boolean function으로 모든 columns을 covering 할 수 있는 17개의 EPI를 찾는 문제이다. 개체 공간은 이며 적합도는 columns를 covering하는 개수(6bits)와 그에 따른 cost의 합의 반전값을 결합하여 표현하였다.

Row					Optimal Sol.(hex)	Cost
31-28	00-0--0	0-001—	0--0-10	-0-01-0	A	8
cost	4					
27-24	-0001—	0-—100	-—001-1	-0-010-	7	12
cost	4					
23-20	—0011-0	--1-100	—11-00-	-1-1-00	B	12
cost	4					
19-16	0010-0-	0-0-010	001--00	-0-0101	4	5
cost	5					
15-12	001-00-	--00110	100-0-1	-010-01	A	10
cost	5					
11-8	--10001	1--0001	0-1100-	-01101-	5	10
cost	5					
7-4	-0110-0	-11-001	10-01-1	10-0-01	6	10
cost	5					
3-0	111—00	-11100-	010101-	111000-	A	11
cost	5		6			
					Total Cost	78 (4E)
					INV(Total Cost)	B1

: EPEPI

그림 7 32 x 63 SCP 구성도
Fig. 7 32X63 SCP (Set Covering Problem)

2.5.3 실험 및 결과

앞에서 제시한 두 다봉 문제에 관하여 본 논문에서 제안한 알고리즘의 성능을 수행하였다. 실험조건은 개체군의 크기는 256, 돌연변이율은 5.0%, 분할구조의 선형 선택 조건을 결정하는 파라미터인 N_{max} 는 2개의 분할구조에서는 1.4, 4개의 분할구조에서는 1.6을 사용하였으며 각각 분할구조를

1, 2, 4개로 하였을 경우에 대하여 표 3 에서 수렴세대를 100회 실험의 평균값을 비교하였다. 분할구조가 2개에서는 각각 Exploitation과 Exploration 성질이 강한 교차 연산자를 쌍을 지었고 분할구조가 4개에서는 모든 교차 연산자를 사용하였다.

표 3 단일개체군과 분할개체군 성능비교

Table 3 Performance comparison between single population and subpopulation

Num_{subpop} (Num_{subpop})	Xover type	32×63 SCP		Math. function	
		Generation (AVG.)	Num_{best}< 2000 gene.	Generation (AVG.)	Num_{best}< 2000 gene.
0(1)	1-point	3029.3	7	2889.7	16
	2-point	2987.7	9	2816.6	21
	Uniform	2953.3	7	2789.5	19
	Direspectable	4967.5	0	4873.2	0
1(2)	1-point/Uniform	2498.3	29	2322.4	31
	2-point/Uniform	2401.2	26	2291.2	30
	1-point/Dires.	2576.6	30	2346.1	28
	2-point/Dires.	2522.7	28	2301.1	25
2(4)	All	2356.2	34	2109.8	36

분할구조가 1개일 때는 기존의 속도 지향의 무작위 개체 선택의 선택 압력을 수정 보완한 modified-tournament 알고리즘과 같은 동작을 수행하게 되며 이를 분할구조가 나누어 졌을 경우와 비교하면 평균 10~20% 정도의 수렴세대의 단축을 얻을 수 있었다. 뿐만 아니라 평균 수렴세대보다 적은 2000세대 이전에서 최적해를 찾는 실험 횟수 면에서도 분할구조가 단일 구조에 비하여 향상된 결과를 관측할 수 있었다. 이것은 실시간 응용분야에서 보다 큰 장점이 될 수 있는 부분이다. 다만, 적합도 함수의 복잡도가 낮은 문제에 관해서는 이 정도 결과의 차이를 관측하기는 어려울 수 있지만, 유전자 알고리즘의 범용성을 고려한다면 복잡도가 높거나 낮은 모든 문제를 고려해야만 한다.

본 연구에서 제안한 분할 구조 알고리즘은 단일 개체군을 사용하는 modified-tournament 알고리즘에 비해 각 부 개체군들의 최적해간 순위를 결정하는 연산과 그 순위에 따른 탐색을 수행할 부 개체군을 선택하는 부분에서 추가적인 연산시간이 사용될 수 있으나 제안한 알고리즘은 하드웨어의 구현까지 고려되어 수정 및 고안되었기 때문에 pipeline 구조로 이를 보완 하여 modified-tournament 알고리즘 프로세서[7]에 비해 추가적인 연산이 발생하지 않도록 설계하였다.

3. 분할구조 유전자 알고리즘 프로세서

3.1 제안한 분할구조 GAP의 구조

본 연구에서 제안한 분할구조 유전자 알고리즘은 하드웨어 지향의 알고리즘으로 연산 속도의 향상 및 효율의 증진을 위해 하드웨어로 구현하기 위하여 앞장에서 알고리즘을 제안하고 성능을 검증하였다.

유전자 알고리즘 프로세서의 효율적인 구현은 하드웨어의 복잡성과 성능을 동시에 고려한 구조여야 한다. 따라서 본 연구에서는 이를 위해 프로세서의 내부의 모든 동작은 중앙 제어 모듈에 의해 각 모듈들이 파이프라인 적으로 동작되어 지도록 설계하였다. 그림 8 는 제안된 분할구조 유전자 알고리즘 프로세서의 내부 블록 다이어그램을 나타낸 것이다. 전체 내부 구조는 주요 11개의 부분 블록으로 구성되어 중앙 제어 모듈에 의해 기능적인 세부모듈이 동작한다. 각 블록들은 적용되는 알고리즘에 따라 연산하는데 소요되는 지연시간이 다르므로 전체 프로세서의 동시 실행성에 중요한 영향을 미치는 요소가 된다.

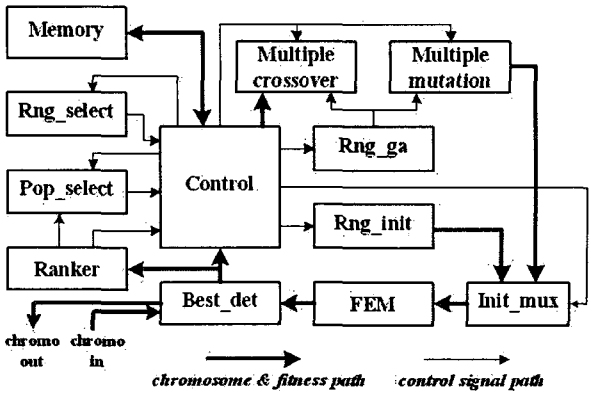


그림 8 제안한 분할구조 GAP 블록다이어그램

Fig. 8 Block diagram of subpopulation GA processor

3.2 병렬처리와 pipeline 구조의 적용

유전자 알고리즘은 기본적으로 선택, 교차, 돌연변이, 적합도 평가라는 과정이 반복되면서 개체를 진화시키는 형태를 취하고 있다. 여기서 선택, 교차, 돌연변이를 수행하기 위해서는 별도의 난수가 필요하게 되며 이를 하드웨어에서는 독립적인 난수 발생기 모듈에 의해 사전에 준비하여 연산시간을 단축할 수 있다. 각 모듈은 알고리즘의 수행 상태에 따라 중앙 제어 모듈의 제어 신호에 따라 하나 혹은 두개가 독립적으로 실행되어질 수 있도록 pipeline 구조를 이용하여 설계하였다. 또한 parallel modified-tournament GA의 장점인 개체군 내의 우성 개체들의 이주 기법(coarse-grained parallelism)을 통해 개체들을 다른 서브 프로세서와 서로 교환하는 방법으로 분할구조를 사용함으로 인하여 발생할 수 있는 개체들의 부족 현상을 보완하여 성능을 보다 높일 수 있도록 하였다.[7,15]

3.3 각 모듈의 구조 및 특징

A. Control module

: 효율적인 pipeline동작을 위해 총 11개의 하위 module을 제어하는 control signal을 발생시키며 processor의 중앙 제어 기능을 수행을 담당하며 그 기능들은 다음과 같다.

-Memory interface 기능

-하위 module들의 명령 기능

-부 개체군의 수위에 입각한 GA operator 선택 기능

B. Rng_init module

: CA(Cellular automata)를 사용하여 개체의 초기군을 생성하는 기능을 수행하는 부분으로 사용자가 원하는 형태의 분할 구조로 초기화가 가능하다.

C. Rng_select module

: CA를 사용하여 GA operating에 참여 할 개체군 내의 개체를 선택할 수 있는 memory address를 control module에 할당한다.

D. Rng_ga module

: CA를 사용하여 GA operating에 사용하게 될 random value를 Crossover 및 Mutation module 에 전송하는 기능을 수행한다.

E. Best_det Module

: 개체군이 초기화가 진행 중일 때는 새로 생성된 두 개체 중 우성 개체를 Control module로 전송하며 탐색과정이 진행 중일 때는 survival-based 유전자 알고리즘을 기반으로 현재의 개체보다 더 적합한 자손만이 생존하도록 결정하는 부분으로서 모든 개체 중 가장 적합한 개체와 현재의 두 부모 개체 중 적합도가 나쁜 개체 사이의 차이에 따른 개체 생존 조건 변화 및 방법의 변화를 꾀할 수 있도록 설계되었으며 개체의 update 과정과 함께 우성 개체의 이주 과정을 수행한다. 또한 두 과정 모두에서 개체군의 최적의 해를 갱신하는 기능도 수행한다.

F. Ranker Module

: Best_det module에서 전송 받은 각각의 부 개체군들의 최적값에 준하여 순위를 정하여 그것을 바탕으로 탐색에 적합한 유전자 연산자를 Control module로 전송하는 기능과 Pop_select에 선형 선택법에 필요한 순위를 전송한다.

G. Pop_select

: Ranker module에서 전송 받은 각각의 부 개체군의 순위값과 사용자가 선택한 에 준하여 CA에 의한 random value를 통해 탐색을 수행할 부 개체군을 선형 선택법[12]으로 결정하여 Control module로 전송하는 기능을 수행한다.

H. Mutiple_crossover module

: 4가지(1-point, 2-point, uniform, disrespect) 형태 교차 연산 중에 현재 진화가 이루어지는 개체군의 순위에 합당하도록 Control module의 제어에 의해 교차 연산을 수행한다.

I. Multiple_mutation module

: 현재 진화가 이루어지는 개체군의 순위에 합당한 돌연변이 연산을 Control module의 제어에 의해 수행한다.

J. Init_mux module

: 개체군이 초기화가 진행 중일 때는 Rng_init module로부터 새로 생성된 개체를 탐색과정이 진행 중일 때는 GA operating을 수행한 개체를 Multiple_mutation module로부터 받아 평가 모듈에 전송하는 기능을 수행한다.

K. FEM module

: 입력되는 개체의 평가를 수행하는 부분으로 내부에는 32row, 63column의 minimal cost set covering problem이 포함되어 있으며 사용자의 선택에 따라 다른 적합도 함수에 적용할 경우 외부 평가 함수로의 개체 전송도 가능하도록 설계되어 있다.

L. Memory

: 탐색 영역의 대표 값들인 개체군을 형성하고 있는 곳으로 개체 32bits와 적합도 16bits로 너비 48bits를 구성하며 개체군의 크기는 256을 사용하여 256X48 크기의 internal memory를 사용한다.

3.4 Prototype

제안된 분할구조 유전자 알고리즘 프로세서의 prototype은 그림 9 에서 보이는 것처럼 APEX EP20K600EBC652-3 FPGA가 장착된 AGENT 2000 Design kit에서 구현되었다. 사용된 FPGA는 약 600,000개의 게이트, 정확히 말하면 24,320개의 로직 요소와 RAM과 ROM을 위한 2,048 bits를 가지는 152개의 내부 메모리 블록(Embedded Array Blocks)으로 구성되어 있다.

제안된 프로세서는 VHDL을 이용하여 FPGA의 24,320개의 로직 요소 중 48%을 이용하여 구현되었으며 최대 동작 주파수는 35.8 MHz의 클럭 속도에서 동작한다.

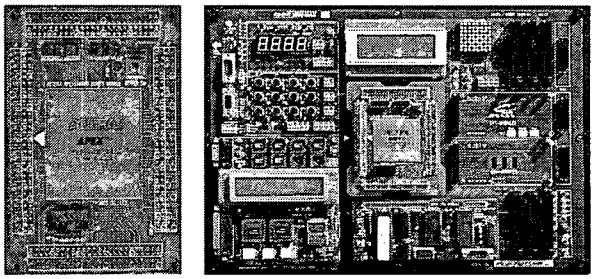
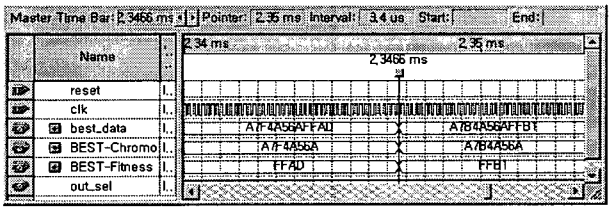


그림 9 분할구조 유전자 알고리즘 프로세서의 원형

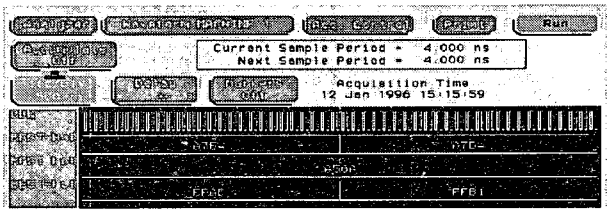
Fig. 9 Prototype of subpopulation GA processor

3.5 실험 및 측정

앞에서 제안한 알고리즘의 성능 검증을 위해 Altera사의 Quatus2.0 툴을 이용하여 설계 및 시뮬레이션을 수행하였다.



(a) Timing simulation result of VHDL coding



(b) Measured result of FPGA (HP1660A)

그림 10 시뮬레이션 결과와 측정결과 비교

Fig. 10 Comparison simulation and measured result

그림 10 의 (a)는 타이밍 시뮬레이션을 수행한 결과이며 (b)는 APEX EP20K600EBC652-3 FPGA에 합성한 디자인에 대해 실제 시뮬레이션과 같은 조건에서 논리 분석기 (HP1660A)로 파형을 측정한 결과이다. 실험 조건은 클럭 속도는 20MHz, 부 개체군의 수를 4개, 돌연변이율을 4.7 % 로 하였다.

4. 결 론

지금까지 본 논문에서는 다양하고 종합적인 분야에 실제 적용을 위해 새로운 하드웨어 지향의 분할구조 유전자 알고리즘을 제안하여 소프트웨어로 성능을 검증하고 VHDL을 이용하여 하드웨어로 구현하였다.

제안된 알고리즘은 이산세대 모델에 비해 개체군의 저장을 위해 사용되는 메모리를 반으로 줄일 수 있는 정상상태 모델 내에서 보다 복잡한 적용함수에 대해 수렴 세대 및 시간을 단축할 수 있는 방법을 제안하여 설계되었다. 개체군을 초기화시에 전체 개체군을 균일한 크기의 부 개체군으로 나누어 각 부 개체군의 수렴 상태를 순위에 입각한 확률적 선형 선택법을 사용하여 진화를 수행할 부 개체군을 선택하며 그에 적합한 유전 연산자를 사용하여 개체의 진화를 수행함으로 해서 최적 해에 접근하는 시간을 구조적으로 단축할 수 있는 방법을 사용하였다. 개체를 갱신하는 방식은 새로 생성된 자손의 적합도가 열성 형질을 지닌 부모 개체의 적합도보다 향상되었을 때 개체군의 갱신이 결정되는 개체 생존 방법을 사용하였고 이주 모델에 근간한 병렬 구조도 적용하였다. 여러 적합도 함수의 시뮬레이션 결과 기존의 속도 중심의 단일 개체군 구조의 Modified Tournament GA 보다 평균 10% 이상의 수렴 세대를 단축할 수 있었으며, 특히 다봉 함수에 대해서는 20% 이상의 성능 향상을 볼 수 있었다.

실시간 문제 적용을 위한 VHDL에 의한 FPGA 구현에 있어서는 하드웨어에서 취할 수 있는 장점인 pipeline 및 parallel 방식을 적용하여 설계되었다.

연구된 유전자 알고리즘 프로세서는 차후의 진화형 하드웨어의 중앙연산처리 장치 및 실시간 처리가 요구되는 최적화 문제, Robot Control, 음성 및 문자인식, 컴퓨터 비전 분야 등의 다양한 문제의 최적의 해를 얻기 위한 도구로써 수많은 응용 분야를 갖는다.

감사의 글

본 연구는 2002년도 산업자원부 차세대 신기술 개발사업 수퍼지능칩 및 응용기술개발 과제의 지원에 의하여 이루어진 연구로서, 관계부처에 감사드립니다.

참 고 문 헌

[1] Michalewicz, Genetic Algorithm + Data Structures= Evolution Programs, Springer-Verlag, 1995.

[2] D. E. Goldberg, Genetic Algorithm in search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[3] K. Dejong, "An analysis of behavior of a class of genetic adaptive system", Ph.D Thesis, University of Michigan, 1975.

[4] J. H. Holland, "Adaptation in Natural and Artificial Systems", Univ. of Michigan Press, Ann Arbor, 1975.

[5] Sandip Sen, "Minimal cost Set Covering using probabilistic methods", Proc. 1993 ACM/SIGAPP Symposium on Applied Computing, pp.157-164, 1993.

[6] Barry Shackleford, Etsuko Okushi et al., "A High-performance Hardware Implementation of a Suvival-based Genetic Algorithm", ICONIP'97, pp 686-691, Nov, 1997.

[7] J. J. KIM, "Implementation of a High-Performance Genetic Algorithm Processor for Hardware Optimization", IEICE TRANSACTIONS on Electronics, Vol.E85-C, No.1, pp. 195-203, January 2002.

[8] Frank Vavak, Terence C. Fogarty, "A Comparative Study of Steady State and Generational Genetic Algorithms", Evolutionary Computing, AISB Workshop, pp 297-304, 1996.

[9] D. Whitley and J. Kauth, "GENITOR: A different Genetic Algorithm", In Proceeding of Rocky Mountain Conference on Artificial Intelligence, 1988.

[10] W. M. Spears, Evolutionary Algorithms(The Role of Mutation and Recombination), Springer, 2000.

[11] W. M. Spears, "Simple Subpopulation Schemes.", Proceedings of the Evolutionary Programming Conference, pp. 296-307, 1994.

[12] J. E. Baker, "Adaptive Selection Methods for Genetic Algorithms", Proceedings of an International Conference on Genetic Algorithms and their Application, pp. 101-111, 1985.

[13] D. Whitley, "The GENITOR algorithm and selection pressure", ICGA'89, 1989.

[14] R.A. Watson and J.B. Pollack , "Recombination Without Respect: Schema Combination and Disruption in Genetic Algorithm Crossover", Proceedings of the 2000 Genetic and Evolutionary Computation Conference, 2000.

[15] H. Muhlenbein, M. Schomisch and J. Born, "The Parallel Genetic Algorithm as function optimizer", Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, 1991.

저 자 소 개

조 민 석(趙 珉 錫)

2001년 인하대 전자재료공학과 졸업. 2003 동 대학원 전자재료공학과 석사.

Tel: 032-874-1663, Fax:032-864-1664

E-mail: minsok75@dreamwiz.com

정 덕 진(鄭 德 鎭)

1948년 2월 8일생. 1970년 서울대 전지공학과 졸업. 1984년 미국 Utah State University(석사)(Electrical Eng.) 1988년 미국 Utah State University(공박)(Electrical Eng.) 1989년~현재 인하대 정보통신공학부 교수

Tel: 032-874-1663, Fax:032-864-1664

E-mail: djchung@inha.ac.kr