SMGA:종족의 분할과 병합을 이용한 효율적인 공진화 알고리즘

도영아° , 박성진, 김명원
숭실대학교 컴퓨터학과
npng1@nownuri.net, comart, mkim@computing.soongsil.ac.kr

SMGA: An Efficient Coevolutionary Algorithm based on Species Splitting and Merging

Younga Do, Soungjin Park, Myungwon Kim
School of Computing, Soongsil University

요 약

진화 알고리즘은 자원 관리, 스케줄링, 퍼지 논리 제어기의 설계 등의 다양한 문제들에 적용되는, 일반적이고 효율적인 최적화 방법이다. 그러나 이러한 진화 알고리즘의 문제점은 탐색해야할 변수의 증가에 따른 차원의 증가로 인하여 탐색공간이 기하급수적으로 늘어난다는 것이다. 이러한 문제점을 해결하기 위해 Potter와 DeJong은 개개의 종족을 독립적으로 진화시킴으로써 탐색공간을 대폭 줄인, 협력 공진화 알고리즘을 제안하였다. 그러나 이것 또한 변수 의존성이 강한 문제들에 대해서는 비효율적인 탐색을 하는 문제점이 있다.

본 논문에서는 종족의 분할과 병합을 이용한 효율적인 공진화 알고리즘을 제안한다. 이 알고리즘은 최적화 하려는 변수들이 서로 의존성이 없는 경우에는 종족의 분할을 통하여 탐색공간의 축소의 이점을 얻고, 최적화 하려는 변수들이 서로 의존성이 있는 경우에는 종족의 병합을 통하여 전역탐색을 하도록 한다. 제안하는 알고리즘을 상품재고 제어 문제(ICP)로 실험하여 현존하는 어떤 공진화 알고리즘보다도 효율적인 결과를 보여준다.

1. 서론

자원 관리, 스케줄링, 패턴 인식 등의 다양한 최적화 문제들을 효율적으로 풀기 위해 현재까지 많은 진화 알고리즘들이 개발되어 왔으나, 이러한 진화 알고리즘들의 공통적인 문제점은 탐색 공간의 확대에 대해 전반적으로 탐색 시간이 오래 걸린다는 것이다. 따라서 최근 진화 알고리즘에 대한 연구는 탐색공간의 확대에 대한 진화속도의 향상에 초점이 맞추어져 왔고, 실제로 많은 성과가 있었다. 특히, 1994년 Potter와 DeJong이 제안한 협력 공진화(cooperative coevolution)방법은 최적화해야 할 각각의 변수에 대하여 종족(species)개념을 도입함으로써, 탐색 공간이 충분히 클 때에도 빠른 수렴속도를 보인다[1]. 특히 협력 공진화의 경우, 개념 학습(concept learning)문제[6]와, 에이전트들 간의 일 배당 문제[7] 등에 적용하여 좋은 결과를 보였다. 그러나, 이 방법도 특정 문제—탐색해야 할 변수들 간의 의존성(variable interdependency)이 강한 문제, Nash 평형점(Nash equilibrium point)[3]이 많은 문제—에 대해서는 오히려 일반적인 진화 알고리즘보다도 성능이 떨어질 수 있다. 이 문제를 해결하기 위하여 1999년 Weicker는 변수들간의 의존성이 존재하면 그 변수들을 표현하는 종족들을 병합함으로써 문제를 해결한 적응적 공진화(adaptive cooperative coevolution) 알고리즘을 제안하였다[2]. 그러나 대부분의 변수들이 서로 의존성을 갖고 있을 경우, 그 변수들을 표현하는 모든 종족이 병합됨으로써, 병합된 후에는 일반적인 진화 알고리즘과 같이 탐색 공간의 급격한 확대로 인해 진화속도가 현저히 감소한다. 본 논문에서 제안하는 알고리즘은 Weicker 알고리즘의 이러한 문제점을 개선하여 종족의 병합 후에도 지속적으로 관찰하면서 필요한 경우 병합된 종족을 다시 각각의 변수를 표현하는 다른 종족으로 분할하여 진화속도를 빠르게 하도록 한다.

본 논문은 다음과 같은 순서로 구성되어 있다. 2절에서는 기존의 진화 알고리즘들에 대하여 간단히 설명하고, 3절에서는 제안하는 알고리즘의 종족에 대한 분할과 병합의 방법을 설명하며, 4절에서는 본 논문에서 제안한 공진화 알고리즘을 벤치마크 함수에 대하여 함수 최적화 문제에 대한 실험 결과를 보이고, 실제 문제로 적용한 상품 재고 제어문제(ICP)에 대한 실험결과를 보인다. 마지막으로 5절에서는 결론을 맺고, 향후연구를 제시한다.

2. 관련연구

(1) 협력 공진화(Cooperative Coevolution)[1]

Potter와 DeJong이 개발한 알고리즘으로서 최적화해야 할 변수들을 각각 따로 진화시키면서 서로의 정보를 공유하도록 하여 탐색 공간이 큰 경우에도 빠른 진화가 가능한 장점이 있다[1]. Potter와 DeJong은 협력 공진화 알고리즘으로 CCGA1과 CCGA2를 제안하였다.

- CCGA1(Cooperative Coevolution Genetic Algorithm)

CCGA1 알고리즘은 일반적인 진화 알고리즘에서의 염색체를 부분해(partial solution)로 잘라내어 종족(species)을 만들어 탐색 공간을 줄임으로써 진화속도의 향상을 꾀하였다. 여기서 종족(species)이란 각각 전체해가 아닌 하나 또는 일부의 변수(부분해)만을 최적화시키기 위하여 만들어진 부분해 단위의 개체군을 의미한다. 즉, 일반적인 진화 알고리즘의 경우, 최적화할 모든 변수들을 하나의 염색체로 만들어 진화시키지만, 종족개념을 도입하게 되면, 각각의 종족은 일부의 변수에 대하여 염색체를 구성하기 때문에 각각 일부의 변수만을 진화시키게 됨으로써 진화속도의 향상을 기할 수 있다.

- CCGA2

CCGA1은 진화속도는 빠르나 Nash 평형점(Nash equilibrium point)이 많은 문제에 대해서는 일반적인 진화 알고리즘보다 성능이 떨어질 수 있다[4]. Nash 평형이란, 1950년 Nash가 발견한 평형상태로서, 둘 이상의 개체가 서로의 상태를 보고 실시간으로 자신의 행동전략을 결정할 때, 이런 상황이 지속되면 어느 순간에는 각각의 개체들이 나머지 개체들의 행동전략이 바뀌지 않는 이상 자신이 행동전략을 바꾸면 자신에게 피해가 돌아오므로, 자신도 행동전략을 바꾸지 않는 상태에 이르게 되는데, 바로 이러한 평형상태를 말한다[3]. 이러한 문제점을 해결하기 위해서 CCGA1에서의 염색체 평가는 다른 종족의 엘리트 염색체(종족내 최적의 염색체)들을 모아서 현재 종족의 염색체의 적합도를 계산하였던 반면에 CCGA2에서의 염색체 평가는 CCGA1과 같이 다른 종족의 엘리트 염색체로도 평가를 하지만, 다른 종족에서 무작위로 선택된 염색체들과도 평가한다. 두 가지의 평가값 중 큰 평가값을 선택하므로 둘 이상의 변수가 변화 가능하도록 하여 Nash 평형점에서 벗어나 전역 최적해를 찾도록 해준다.

(2) 적응적 공진화(ACC:Adaptive Cooperative Coevolution)[2]

K. Weicker와, N. Weicker가 CCGA2를 개선한 알고리즘으로, 몇 가지

를 제외하고는 CCGA와 같다. CCGA2에서는 종족이 하나의 변수(단일 종족)만을 진화시키도록 하였으나, 종족간의 의존성을 진화상황에 제대로 반영하지 못함으로써 그다지 좋은 성능을 보이지 못하였다. 그러므로, ACC에서의 종족은 하나 이상의 변수(복합 종족)를 진화시킬 수 있도록 하였고, 둘 이상의 종족이 서로 의존성[5]이 존재한다면, 종족들은 서로 병합되고, 병합된 종족은 일반적인 진화 알고리즘이 적용되므로 병합된 종족의 최적화하는 변수들이 동시에 변화하여 국소 최적해의 끌림 유역을 벗어날 수 있게 된다. 또한, ACC에서는 병합의 시점을 알기 위해 의존도 행렬이라는 것을 사용한다. 의존도 행렬이란 최적화해야 할 변수들에 대하여 각각의 변수들과 나머지 변수들과의 의존도를 나타내는 대칭 행렬이다.

3. SMGA (Split & Merge Genetic Algorithm)[9]

ACC의 경우는 모든 변수들이 상호 의존성을 갖고 있는 문제에 적용될 경우 모든 종족들이 병합됨으로써, 진화 중반 이후로는 공진화의 이점인 탐색 공간 축소가 이루어지지 않게 된다. ACC의 이러한 문제점을 해결하기 위하여 본 논문에서는 분할과 병합을 함께 하는 진화 알고리즘 SMGA를 제안한다.

종족이 병합되어 진화하는 어느 순간에는 두 가지 상황이 가능하다. 즉, 엘리트가 국소 최적해의 끌림 유역에 존재할 경우와, 또는 엘리트가 전역 최적해의 끌림 유역에 존재할 경우이다. 만약, 국소 최적해의 끌림 유역을 탐색하고 있다면 계속 병합되어 있는 것(복합 종족)이 효율이 좋겠지만, 이미 국소 최적해의 끌림 유역을 벗어나서 전역 최적해의 끌림 유역을 탐색 중이라면 두 종족이 병합되어 있는 것보다는 다시 분할되는 것(단일 종족)이 효율이 좋아진다. 그러나 이 같은 상황을 탐색 중 알 수 없으므로, 본 논문에서는 복합 종족이 일정시간동안 엘리트의 향상에 기여를 하지 못할 경우 분할을 하도록 한다.

본 논문에서 제안하는 SMGA는 진화 과정중 병합상태와 분할상태의 두 가지 상태가 반복된다. 병합방법은 ACC에서의 병합 방법과 유사한 방법을 사용하며, 병합 후, 병합된 종족이 주어진 세대동안 전체 엘리트 염색체(각 종족에서 엘리트만 모아놓은 염색체)의 개선에 기여를 못 할 경우 다시 분할을 하도록 한다. 분할 후에도 의존도 행렬에 의해 다시 병합될 수 있도록 하여, 분할과 병합이 반복되게 한다. 따라서 의존도가 높은 변수쌍일 경우 병합상태가 오래 지속되며, 의존도가 낮은 변수쌍일 경우 분할상태가 오래 지속되므로 효율적인 진화가 가능하다.

(1) 종족 병합

V_j, V_k를 각각 t세대의 j, k번째 종족이 최적화 할 변수들을 원소로 갖는 집합이라고 가정하자. 이 때 집합 V_k의 원소들을 표현하는 종족을 $S_{V_k}(t)$이라고 하고, $V = V_j \bigcup V_k$라 할 때, 종족 $S_{V_j}(t)$와 종족 $S_{V_k}(t)$가 병합된 새로운 종족 $S_V(t)$는 (식 1), (식 2)와 같이 결정된다.[2]

$$\text{merge}(S_{V_j}(t), S_{V_k}(t)) = S_V(t) \quad \text{(식 1)}$$

$$\left.\begin{array}{l} S_{V_j}(t) = \langle c_1^{S_{V_j}(t)}, \ldots, c_n^{S_{V_j}(t)} \rangle \\ S_{V_k}(t) = \langle c_1^{S_{V_k}(t)}, \ldots, c_n^{S_{V_k}(t)} \rangle \end{array}\right\} \rightarrow S_V(t) = \langle c_1^{S_V(t)}, \ldots, c_n^{S_V(t)} \rangle \quad \text{(식 2)}$$

여기서 새로운 종족 $S_V(t)$의 i번째 염색체 $c_i^{S_V(t)}$는,

$$c_i^{S_V(t)} = \begin{cases} c_{elite}^{S_{V_j}}(t) \circ c_{elite}^{S_{V_k}}(t) & : i = 1\text{일 때}, \\ c_{elite}^{S_{V_j}}(t) \circ c_{rand}^{S_{V_k}}(t) & : 2 \le i \le \lceil \frac{n}{3} \rceil \text{일 때}, \\ c_{rand}^{S_{V_j}}(t) \circ c_{elite}^{S_{V_k}}(t) & : \lceil \frac{n}{3} \rceil < i \le \lceil \frac{2n}{3} \rceil \text{일 때}, \\ c_{rand}^{S_{V_j}}(t) \circ c_{rand}^{S_{V_k}}(t) & : \text{그 외의 경우} \end{cases} \quad \text{(식 3)}$$

와 같이 결정된다.

결과적으로 새로이 생겨나는 종족 $S_V(t)$는 개체군 중, 종족 $S_{V_j}(t)$의 엘리트 부분해를 포함한 염색체들과, 종족 $S_{V_k}(t)$의 엘리트 부분해를 포함한 염색체들, 그리고, 국소 최적해를 벗어날 수 있기 위해 종족 $S_{V_j}(t)$와 종족 $S_{V_k}(t)$에서 무작위로 선택된 부분해들로 이루어진 염색체들로 이루어진다. 본 논문에서는 (식 3)에서와 같이, 각각의 크기를 종족 $S_V(t)$의 개체군 크기의 1/3로 정하였다.

이러한 병합의 시점은 ACC와 같이 의존도 행렬을 사용하여 행렬 내의 의존도 값이 주어진 값보다 커질 경우로 한다.

이렇게 ACC와 달리 적용한 이유는, 현재 의존도 행렬로 인해 변수 의존성이 발견되었으므로 현재 탐색점이 Nash 평형점에 존재할 확률이 높으나 ACC와 같은 방법으로 병합할 경우, 어느 경우에도 하나 이상의 종족 엘리트가 포함되게 되어 Nash 평형점 탈출 확률이 떨어지기 때문이다. 따라서 본 논문에서는 한 가지 경우를 더 추가하여 둘 이상의 변수가 동시에 변화 할 수 있도록 양쪽의 종족에서 임의로 선택한 염색체를 병합한 염색체도 개체군에 포함시켜 Nash 평형점 탈출 확률이 높도록 하였다.

종족의 병합은 탐색 공간의 축소 측면에서 봤을 때는 비효율적이나, 국소 최적해의 탈출 측면에서 봤을 때는 효율적이다. 이러한 이유는 종족의 병합을 통해 탐색 공간은 확대되나, 진화연산자의 적용을 통해 두 종족의 부분해가 동시에 변화하는 것이 가능하므로, 앞서 기술한 Nash 평형점의 탈출이 가능해지기 때문이다.

(2) 종족 분할

종족의 분할은 탐색 공간의 축소로 인한 진화속도의 향상에 목적이 있다. 종족 병합 단계에서 일정 세대 동안 엘리트의 향상이 이루어지지 않으면 국소 최적해를 탈출한 것으로 간주하고, 종족의 분할을 통해 다시 탐색 공간을 축소시킴으로써 진화속도의 향상을 이룬다. 병합된 종족을 분할할 때 병합 전의 두 종족이 각각 어떤 변수들에 대해서 진화를 했었는지 알 수 없기 때문에 모든 변수들로 나뉘어져 단일 종족들로 분할된다.

병합된 종족 $S_{V_k}(t)$가 존재하고, $V \subseteq V_k$일 때, $\text{proj}\left(c_i^{S_{V_k}(t)}, V\right)$를 병합된 개체군 $S_{V_k}(t)$ 중 i번째 염색체 $c_i^{S_{V_k}(t)}$에서, $V_k(t)$의 원소 중 V의 변수들을 표현하는 염색체의 유전자 부분만을 뽑아내는 함수라고 가정하자. 이 때 종족 k를 나타내는 변수집합 V_k의 분할(partition)을 $P = \{U_1, U_2, \ldots, U_m\}$ $\left(\bigcup_{i=1}^{m} U_i = V_k,\ U_i \bigcap U_j = \emptyset\ (i \neq j)\right)$라고 하면, 종족 $S_{V_k}(t)$에서 나눠진 i번째 종족 $S_{U_i}(t)$와, $S_{U_i}(t)$의 j번째 염색체 $c_j^{S_{U_i}(t)}$는 (식 4), (식 5)와 같이 결정된다.

$$\text{split}(S_{V_k}(t), P) = \{S_{U_1}(t), S_{U_2}(t), \ldots, S_{U_m}(t)\} \quad \text{(식 4)}$$

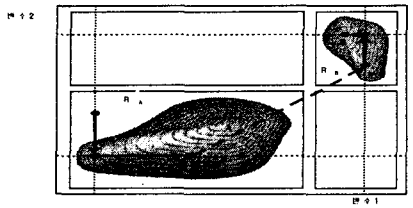
$$S_{U_i}(t) = \langle c_1^{S_{U_i}(t)}, c_2^{S_{U_i}(t)}, \ldots, c_n^{S_{U_i}(t)} \rangle$$

$$c_j^{S_{U_i}(t)} = \text{proj}\left(c_j^{S_{V_k}(t)}, U_i\right) \ : 1 \le i \le n,\ 1 \le j \le m \quad \text{(식 5)}$$

결과적으로 생겨나는 각각의 종족 $S_{U_i}(t)$는 분할되기 전의 종족 $S_{V_k}(t)$에서 염색체의 부분해(유전자)를 해당되는 종족에 맞도록 분할하여 갖게 된다. 즉, 종족의 분할은 어떤 종족을 나타내는 변수집합을 몇 개의 부분집합으로 분할하여 분할된 변수집합 각각에 해당하는 부분종족으로 분리하는 것이다. 따라서 종족의 분할은 한 종족에 대하여 종족을 나타내는 변수집합을 어떻게 분할하는가에 따라 여러 가지 방법으로 분할 할 수 있다. 본 논문에서는 종족의 분할을 최소단위의 변수집합 즉, 1개의 변수만을 포함하는 집합에 대응하는 종족(단일종족)으로 분할하는 것으로 한다. 분할의 시점은 병합된 종족들 각각을 세대마다 보면서, 그 종족이 전체 엘리트 염색체의 향상에 기여를 했는지를 체크하여 주어진 세대동안 기여하지 못하였을 경우 분할하도록 한다. 다시 말하면, 병합된 종족이 여러 세대동안 개선이 없는 경우 이를 탐색점이 진화 속도가 느린 탐색 영역에 존재한다고 가정하고, 다시 종족을 분할함으로써 탐색 공간의 축소를 통해 진화 속도가 향상되도록 한다. 본 논문에서는 해당되는 종족이 30세대동안 엘리트 염색체의 향상에 기여를 못하는 경우 분할 하도록 하였다. 실험을 통해 30세대 정도가 모든 문제에서 평균적으로 좋게 나왔으나, 10세대에서 50세대 사이의 몇 세대를 정하여도 대부분 비슷한 결과를 나타내었다.

종족 분할의 영향으로는 탐색 공간의 축소로 인한 빠른 진화의 이점과, 그로 인한 국소 최적해에 빠질 수 있는 단점이 있다. 그러나 종족 병합 단계에서 국소 최적해를 벗어났다고 가정되므로 결과적으로는 진화속도를 향상시키는 이점이 더 많다고 할 수 있다.

이와 같이 분할과 병합을 병행하면 <그림 1>과 같이 진화가 진행된다. 그림에서 극값 A와 B가 존재할 때 B가 전역 최적해 라면, <그림 1>에서 국소 최적해의 끌림 유역인 R_A영역에서 진화가 시작되어 변수 의존성을 인식하고 종족이 병합되기 전까지 CCGA1과 같이 진화가 이루어진다. 그러다가 의존성을 인식한 시점에서는 두 종족이 병합되어 전역 최적해의 끌림 유역인 R_B영역으로 유입된 후, 다시 종족이 분할됨으로써 빠른 진화가 가능해진다. 즉, CCGA의 탐색 공간의 축소를 통한 빠른 진화 속도와 ACC의 국소 최적해 탈출 능력을 동시에 가지게 되어 효율적인 진화를 이루게 된다.



<그림 1> SMGA의 진화과정

4. 실험

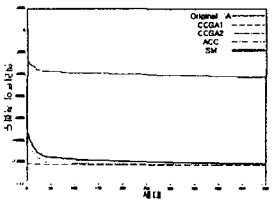
실험에서는 함수 최적화 문제에 대하여 벤치마크 함수인 Schwefel 함수와 실제 문제에 대한 실험으로는 상품 재고 제어문제(ICP)를 사용하여 SMGA의 타당성을 검증하였다.

(1) Schwefel 함수

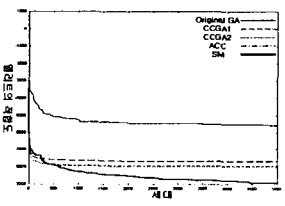
Schwefel 함수는 sin함수를 가지고 있는 항이 있고, sin함수에 의한 진동이 $\vec{x}=(0,0,\ldots,0)$를 중심으로 바깥 쪽으로 나가면서 커지는 형태를 가지고 있다. 함수의 원형은 (식 6)과 같다.

$$F(\vec{x})=\sum_{i=1}^{n}[-x_i\cdot\sin(\sqrt{|x_i|})], \quad -500\le x_i\le 500 \qquad \text{(식 6)}$$

이 함수에서 변수축을 회전하지 않았을 경우 전역 최적해는 $\vec{x}=(420.9687,420.9687,\ldots,420.9687)$일 때, $F(\vec{x})=-n\cdot 418.9829$ 이다.



<그림 2> Schwefel함수 원형에 대한 실험결과



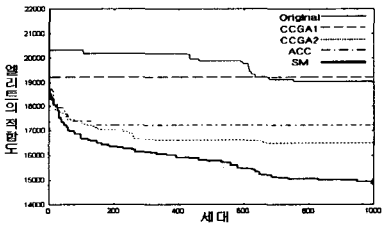
<그림 3> 변수축을 회전시킨 Schwefel함수에 대한 실험결과

실험 결과를 보면, 변수축을 회전하지 않은 경우에는 변수간에 의존성이 전혀 존재하지 않으므로, 일반적인 진화 알고리즘을 제외한 대부분의 알고리즘이 빠른 진화속도를 보인다. 특히, CCGA1의 경우는 몇 세대가 지나지 않아 전역 최적해에 수렴하고 있음을 알 수 있다. 그러나, 변수축 회전의 경우에는 대부분의 알고리즘이 국소 최적해에 빠져 진화가 멈추었지만 SMGA의 경우 병합과 분할의 반복으로 계속적인 진화를 하고 있음을 알 수 있다.

(2)ICP(Inventory Control Problem)[8]

본 논문에서는 실제 문제 적용으로 상품 재고 제어문제(ICP)[8]를 택하여 실험하였다. ICP는 상품의 공급자 측면에서 가장 적은 비용으로 가장 많은 상품을 팔려고 하는 문제이다. 상품은 창고에 저장되며, 상품이 창고에서 차지하는 부피와 저장되는 시간에 따라 창고 이용비가 달라진다. 즉 창고를 차지하는 부피가 크면 클수록, 창고에 쌓여있는 시간이 길면 길수록 그에 비례하여 재고유지 비용도 늘어난다. 그리고, 상품 생산지에서 창고까지 상품을 옮길 때도 비용이 들고, 생산지에서 창고까지 옮기는데 걸리는 시간도 고려한다. 즉 상품 요청을 하고 옮기는 동안 소비자가 와서 물건을 살수도 있는 것이다. 한번에 여러 가지 상품을 옮기게 되면 한번에 하나의 상품을 옮길 때 드는 비용보다 적어진다. 즉, 되도록 많은 상품을 한꺼번에 옮겨야 비용이 적게든다. 또한, 소비자가 상품을 사러 왔을 때 원하는 상품의 재고가 없으면 그 상품을 팔았을 때의 이윤만큼의 손해가 생기게 된다. 이와 같은 모든 상황을 고려하여 실험하였다.

10가지의 상품에 대하여 실험하였으며, 각각의 상품에 대하여 상품을 창고로 옮기는 시점(창고의 재고량)과, 한번에 옮기는 양이 변수가 된다. 즉, 전체 최적화해야 하는 변수는 각각의 상품에 대하여 2개가 존재하므로 전체 20개가 된다. 실험의 편의를 위해 모든 상품에 대하여 상품을 옮기는 비용과 옮기는데 걸리는 시간, 창고 유지비용, 창고의 크기, 상품의 이윤 등을 모두 동일하게 하였다. 실험 데이터는 10가지의 상품 각각에 대하여 한 번의 시간에 0~5개까지 팔릴 수 있도록 무작위로 생성하였고, 데이터 개수는 1000개로 하였다.



<그림 4> 상품재고 제어 문제의 실험결과

실험 결과를 보면 일반적인 진화 알고리즘은 앞의 실험과 같이 탐색공간의 확대로 인하여 효율이 좋지 않았고, CCGA1은 진화 초기부터 국소 최적해인 Nash 평형점에 빠져 진화가 되지 않음을 볼 수 있다. ACC의 경우는 150세대 정도에서 모든 종족이 병합되어 진화 속도가 현저히 감소되었으며, CCGA2는 다른 알고리즘 보다는 좋지만, 변수 의존성을 제대로 반영하지 못함으로 인해서 그다지 좋은 결과를 보이지 못하였다. 그러나 SMGA(SM)의 경우 종족의 분할과 병합을 통한 지속적인 진화로 다른 알고리즘보다 성능이 우수함을 나타내었다.

5. 결론 및 향후 연구

본 논문에서는 진화 알고리즘의 성능을 향상시키기 위하여 기존의 방법들을 개선한 새로운 방법을 제시하였고, 실제 문제 적용으로 상품 재고 제어문제(ICP)에도 응용하여 실험하여 보았다. 본 논문에서 제안한 SMGA는 변수 의존성을 고려하면서도 진화 도중 병합과 분할을 반복함으로써 ACC의 문제점을 해결하여 다른 알고리즘들 보다 좋은 성능을 보여주었다.

현재 벤치마크 함수들과 상품 재고 제어문제(ICP)에 대해서 실험을 하였으나, 제안하는 알고리즘의 타당성과 일반성을 검증하기 위해서 보다 많은 변수 의존성이 높은 여러 가지 실제적인 문제(퍼지 논리 제어기의 최적화 문제, 자원 할당 문제 등)에 적용하여 효율성을 검증해 보고자 한다. 그리고, 잦은 종족의 병합과 분할로 인해 낮아진 효율성을 높이기 위해 종족의 병합 후에도 병합 전의 종족들을 버리지 않고 동시 진화하면서, 진화 도중 각 종족의 진화 기여도를 체크하여 진화 기여도가 낮은 종족들을 없애는 방법을 적용해 보고자 한다. 이렇게 하면 가능한 종족들의 조합이 많아서 어느 종족을 생성시킬 것인지를 결정하기 어렵다는 단점이 있으나 진화 기여도를 고려하여 종족의 존망을 결정하게 되므로 효율적인 진화가 가능할 것이다.

참고 문헌

[1] Potter, M. A. and K. A. DeJong (1994). A cooperative coevolutionary approach to function optimization. In Y. Davidor and H.-P Schwefel (Eds.), *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, pp. 249-257. Springer-Verlag

[2] Weicker, K. and Weicker N. (1999). On the improvement of coevolutionary optimizers by learning variable interdependencies, *Congress on Evolutionary Computation (CEC99)*, pp. 1627-1632

[3] Nash, J. (1951). Non-cooperative games. *Annals of Mathematics 54*(2), pp. 286-295

[4] Potter, M. A. (1997). The design and analysis of a computational model of cooperative coevolution. Ph. D. thesis, George Mason University, Fairfax, Virginia.

[5] Munetomo, M. and Goldberg, D. E. (1999). Identifying Linkage Groups by Nonlinearity/Non-monotonicity Detection, In Banzhaf, W. et al. (Eds.). *Proceedings of the Genetic and Evolutionary Computation Conference 1999 (GECCO-99)*. San Fransisco, CA: Morgan Kaufmann..

[6] Potter M. A. and K. A. De Jong (1998). The coevolution of antibodies for concept learning. *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature*, pp. 530-539. Springer-Verlag.

[7] Sen S., Biswas A., Debnath S. and Puppala N. (1999) Cooperative Coevolution using Shared Memory *Genetic and Evolutionary Computing Conference (GECCO '99)* workshop on Coevolutionary Algorithms and Coevolutionary Agents.

[8] Roger Eriksson and Bjorn Olsson (1997). Cooperative Coevolution in Inventory Control Optimisation. In Smith, Steele and Albrecht (Eds.) *Proceedings of 3rd International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA97)*, Norwich, UK, April 1-4 1997.