종족의 분할과 병합을 이용한 효율적 공진화 알고리즘

(An Efficient Coevolutionary Algorithm based on Species Splitting and Merging)

박 성 진 † 김 명 원 ††
(Soung Jin Park) (Myung Won Kim)

요 약 진화 알고리즘은 자원 관리, 스케줄링, 패턴 인식 등의 다양한 문제들에 적용되는, 일반적이고 효율적인 최적화 방법이다. 그러나 이러한 진화 알고리즘의 문제점은 탐색해야할 변수가 증가할수록 그에 따른 차원의 증가로 인하여 기하급수적으로 늘어나는 탐색공간에 약하다는 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 최적화하려는 변수들이 서로 의존성이 없는 경우에는 종족의 분할을 통하여 탐색공간의 축소의 잇점을 얻고, 최적화하려는 변수들이 서로 의존성이 있는 경우에는 종족의 병합을 통하여 전역탐색을 하도록 한다. 제안하는 알고리즘을 몇 가지 벤치마크 함수 최적화 문제와, 상품 재고 제어문제로 실험하여 현존하는 어떤 공진화 알고리즘 보다도 효율적인 것을 보여준다.

Abstract Evolutionary algorithm is a general and efficient optimization method and it is applied to various problems including resource management, scheduling, and pattern recognition. However, it seriously suffers from curse of dimensionality since the search space expands exponentially as the number of variables to optimize grows. To overcome this problem coevolutionary algorithm has been proposed by Potter and DeJong and it significantly reduces search space by evolving individual species independently. However, it also suffers from inefficient search when problems have high degree of variable interdependency.

In this paper we propose an efficient coevolutionary algorithm using species splitting and merging. The algorithm takes advantage of reduced search space by splitting species when variables are independent while it conducts global search by merging species when variables are interdependent. We have experimented the proposed algorithm with several benchmarking function optimization problems and the inventory control problem, and have shown that the algorithm is more efficient than any existing coevolutionary algorithms.

1. 서 론

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 오래걸린다는 것이다. 최적화 문제들은 문제의 특성상 최적화해야 할 변수의 증가에 따라 차원의 증가로 인하여 탐색 공간이 기하급수적으로 늘어나며, 그것에 비례하여 최적해의 탐색 시간도 기하급수적으로 늘어난다. 따라서 최근의 진화 알고리즘에 대한 연구는 탐색공간의 확대에 대한 진화속도의 향상에 초점이 맞추어져 왔고, 실제로 많은 성과가 있었다. 특히,

† 정 회 원 : (주)전능메디칼소프트웨어 연구원
a-life@hanmail.net
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
mkim@comp.ssu.ac.kr
논문접수 : 2000년 5월 18일
심사완료 : 2000년 11월 27일

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[8]와, 에이전트들 간의 일 배당 문제[9] 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 분할될 수 있도록 한다. 즉, 탐색점이 국소 최적해를 벗어나 전역 최적해의 끌림 유역(basin of attraction)에 유입되었을 경우, 다시 종족을 분할하여 진화속도를 빠르게 하도록 한다.

본 논문은 다음과 같은 순서로 구성되어 있다. 2절에서는 일반적인 진화 알고리즘부터 제안하는 알고리즘까지의 선상에 있는, 기존의 진화 알고리즘들에 대하여 설명하고, 3절에서는 제안하는 알고리즘의 종족에 대한 분할과 병합의 방법을 설명하며, 4절에서는 벤치마크 함수들에 대하여 함수 최적화 문제에 대한 실험 결과를 보이고, 실제문제로 상품 재고 제어문제에 대한 실험결과를 보인다. 마지막으로 5절에서는 결론을 맺고, 향후연구를 제시한다.

2. 기존의 진화 알고리즘

2.1 진화 알고리즘(GA: Genetic Algorithm)

진화 알고리즘(Genetic Algorithm)은 1975년 존 홀랜드(John Holland)에 의해 제안된 전역적 탐색 기법으로 자연현상의 자연 도태와 진화의 메커니즘에 기반을 둔 확률적인 탐색 알고리즘으로서 특히 함수 최적화문제, 퍼지 논리 제어기의 최적화 등 최적화 문제에 효율적이다[7]. 진화 알고리즘은 문제에 대한 후보해 또는 염색체(chromosome, individual)들의 집단인 개체군(population)을 유지한다. 일반적으로 진화 알고리즘에서의 각 후보해들은 순서화된 고정 길이이며 한 속성(gene)을 나타내는 값(allele)들의 배열로 표현된다. 알고리즘은 <그림 1>과 같고, 과정을 간략히 설명하면 다음과 같다.

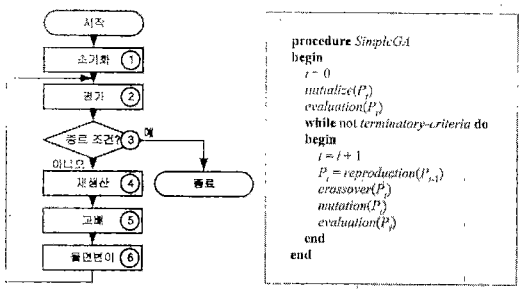

그림 1 기본적인 진화 알고리즘

우선 임의로 염색체를 발생시켜 초기 개체군(P_0)을 초기화(①: *initialize*(P_t))한 후 생성된 각 염색체들을 평가(②: *evaluation*(P_t))한다. 각 염색체의 적합도를 계산한 후, 종료조건을 만족(③: *terminatory*$-$*criteria*)하면 알고리즘을 종료하고, 그렇지 않으면, 세대를 증가($t=t+1$)한 후, 그 적합도에 근거하여 다음 세대의 부모를 결정(④: *reproduction*(P_t))하고, 부모염색체들로부터 교배(⑤: *crossover*(P_t))와 돌연변이(⑥: *mutation*(P_t))에 의해 자식 염색체들을 생산한다. 그리고, 새로이 생성된 자식 염색체들을 평가하기 위해 ②로 돌아가 반복한다. 각각의 단계를 설명하면 다음과 같다.

① 개체군 초기화(initialization): 해결하고자 하는 문제에 대한 가능한 해들을 정해진 형태의 자료구조인 염색체로 표현하는 과정이다.

② 염색체 평가(evaluation): 적합도는 문제의 해에 각 염색체가 얼마나 적합한지를 측정하는 척도로 적합도 함수를 이용하여 평가한다.

③ 종료조건(terminatory criteria): 진화를 종료하는 조건으로 완벽한 해를 발견, 오차 변화율, 최대 세대수 만큼 진화 등의 방법이 있다.

④ 재생산(reproduction): 잘 적응한 해들은 살아남고 잘 적응하지 못한 해들은 도태되도록 유도함으로써 자연선택 현상을 모델링한다. 보통 적합도에 비례하여 다음 세대에 복제되도록 하는 적합도 비례 선택(fitness proportionate selection)을 사용한다.

⑤ 교배(crossover): 두 개의 염색체를 선택하여 유전

자를 부분적으로 교환함으로써, 자손은 부모 세대로부터 유전자를 이어받으면서도 부모와는 다른 형질을 가지도록 한다.

⑥ 돌연변이(mutation): 하나의 염색체의 유전자에서 임의로 선택된 유전인자를 가능한 다른 값으로 바꿈으로써, 현재 개체군에 존재하지 않는 새로운 염색체를 생성하며 탐색의 방향이 지역적 극값으로 향할 경우 여기에서 벗어날 수 있도록 한다.

2.2 협력 공진화(Cooperative Coevolution)

Potter와 DeJong이 개발한 알고리즘으로서 최적화해야할 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 제안하였는데, 그 각각의 알고리즘은 다음과 같이 수행된다.

2.2.1 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<그림 2>.

<그림 3>의 알고리즘에서 각 단계는 <그림 1>의 일반적인 알고리즘의 단계와 동일하고, 각각의 종족에 대하여 알고리즘을 수행한다는 것이 다르다. 여기서 종족이란 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. 이러한 각각의 종족은 염색체 평가를 제외하고는 일반적인 진화 알고리즘이 적용되고, 염색체의 평가는 <그림 4>와 같이 다른 종족의 엘리트 염색체(종족내 최적의 적합도를 가진 염색체)들을 모아서 현재 종족의 염색체의 적합도를 계산하게 된다. 그런데 협력 공진화에서는 종족을 변수단위의 부분해를 최적화하도록 정하고 있다. 즉, 최적화해야 할 변수가 s개 일 때, 종족 또한 s개가 있으며 종족 k의 i번째 염색체 c_i^k의 적합도는 (식 1)과 같이 결정된다.

$$F^{CCGA1}(c_i^k) = F(c_{elite}^1, c_{elite}^2, \ldots, c_i^k, \ldots, c_{elite}^s) \quad (1)$$

여기서 c_{elite}^k는 종족 k의 엘리트 염색체를 나타내고, F는 문제 영역에서 전체 변수에 대한 적합도 함수를 나타낸다.

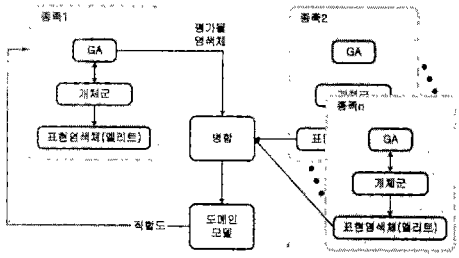

그림 2 협력 공진화에서의 염색체 형태

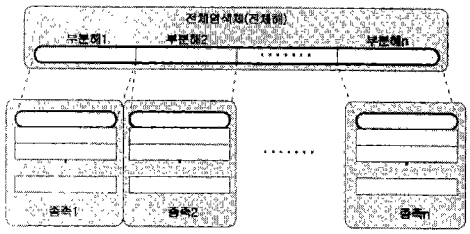

그림 3 CCGA1 알고리즘

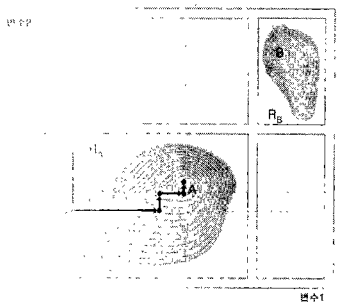

그림 4 협력 공진화에서의 염색체 평가

```
procedure CCGA1
begin
  t = 0
  for each species s do
  begin
    initialize(P^s_t)
    evaluation(P^s_t)
  end
  while not terminatory-criteria do
  begin
    t = t + 1
    for each species s do
    begin
      P^s_t = reproduction(P^s_{t-1})
      crossover(P^s_t)
      mutation(P^s_t)
      evaluation(P^s_t)
    end
  end
end
```

그림 5 협력 공진화에서의 진화과정

각각의 종족이 한 세대씩 돌아가면서 진화하기 때문에 (식 1)에 의해 한 종족(변수)이 진화할 때 나머지 종족(변수)는 엘리트로 고정되므로, 2개의 변수 최적화에 대한 진화는 <그림 5>와 같이 진행된다. 여기서 일반적인 진화 알고리즘이 탐색하는 공간은 2차원 공간 전체가 되나, 협력 공진화 알고리즘이 탐색하는 공간은 <그림 5>에서 점선들로 나타나는, 변수 축에 평행한 직선들로 축소되므로 빠른 진화가 가능하게 된다.

2.2.2 CCGA2

CCGA1은 진화속도는 빠르나 <그림 6>과 같은 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]. 이런 평형상태는 다음과 같이 해석될 수 있다. <그림 6>에서 극값은 A와 B가 존재한다. CCGA1의 측면에서 봤을 때, R_A영역은 극값 A의 끌림유역에 해당되고, R_B영역은 극값 B의 끌림유역에 해당된다. CCGA1에서는 진화과정중 다른 변수들은 고정시키고 변수 하나씩 돌아가면서 진화시키기 때문에 탐색점을 지나면서 변수 축에 수직이 되는 직선들 만을 탐색하게 된다. 따라서 한번 어떤 극값의 끌림유역에 들어가게 되면, 그 끌림유역을 벗어나기가 힘들어진다. 만약 <그림 6>에서 극값 B가 전역 최적해일 때 R_A영역에서 탐색을 시작했다면 국소 최적해인 극값 A에서 변수1과 변수2가 Nash평형을 이루게 되어, 절대로 탐색점이 전역 최적해의 끌림유역인 R_B영역으로 갈 수가 없게 된다. 이런 상황에서는 Nash 평형의 정의에 의해, 변수1과 변수2를 동시에 변화시켜야지만 탐색점이 R_B영역으로 갈 수 있다. 이럴 때, 변수1과 변수2 사이에는 변수 의존성(variable interdependency)이 존재한다고 한다. 변수 의존성의 정의를 다르게 나타내면 다음과 같다. 적합도 값이 높은 염색체가 좋은 염색체라고 가정하고, 어떤 염색체 c가 $c=(c_1, c_2, c_3, \ldots, c_s)$와 같이 구성되어 있다고 가정하고, $1 \le i, j \le s$, $i \neq j$인 i, j에 대하여, $\Delta F_i(c)$는 염색체 c의 c_i를 c_i'으로 교체했을 때의 적합도의 차이로, 그리고 $\Delta F_{ij}(c)$는 염색체 c의 c_i와 c_j를 각각 c_i'과 c_j'으로 교체했을 때의 적합도의 차이로 정의하면,

$$\Delta F_i(c) = F(.., c_i', ...) - F(.., c_i, ...) \quad (2)$$
$$\Delta F_j(c) = F(....., c_j', ..) - F(....., c_j, ..)$$
$$\Delta F_{ij}(c) = F(.., c_i', ., c_j', ..) - F(.., c_i, ., c_j, ..)$$

와 같이 ΔF가 계산된다. 이 때, 모든 경우의 c에 대하여,

$$\Delta F_{ij}(c) \le \Delta F_i(c) + \Delta F_j(c) \quad (3)$$

를 항상 만족하면 염색체 c의 구성요소인 c_i가 나타내는 변수 x_i와 c_j가 나타내는 변수 x_j에 의존성이 없다고 정의할 수 있고, 모든 경우의 c에 대하여,

$$\Delta F_{ij}(c) > \Delta F_i(c) + \Delta F_j(c) \quad (4)$$

인 경우가 하나라도 존재하면 변수 x_i와 변수 x_j에 의존성이 있다고 정의할 수 있다[5].

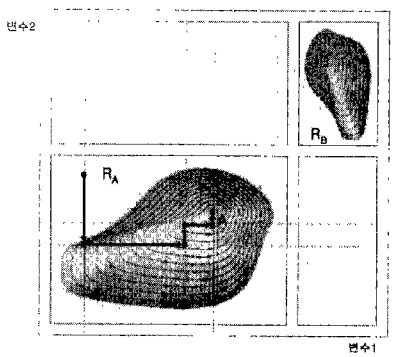

그림 6 변수간에 의존성이 존재하는 문제

다시 말하면, 함수 최적화에서 어떤 함수 F에 대한 변수 의존성이란, 함수 F의 매개변수 중 변수 x, y가 존재할 때, 최적화하기 위해 변수 x 값을 변화하는 것이 변수 y 값의 변화를 초래하는 경우가 발생하면—변수 x, y를 각각 따로 진화시키는 것이 같이 진화시키는 것보다 비효율적인 경우— 함수 F에 대해 변수 x, y에는 의존성이 존재한다고 말한다.

CCGA1의 이러한 문제점을 해결하기 위하여 Potter와 DeJong은 CCGA2를 제안하였다. CCGA2는 CCGA1과 진화방법은 유사하나 평가방법을 달리 함으로써 CCGA1과 구별된다. CCGA2에서 최적화해야할 변수가 s개 일 때, 종족 k의 i번째 염색체 c_i^k의 적합도는 (식 5)와 같다.

$$F^{CCGA2}(c_i^k) = \max\{F(c_{elite}^1, c_{elite}^2, \ldots, c_i^k, \ldots, c_{elite}^s), \quad (5)$$
$$F(c_{rand}^1, c_{rand}^2, \ldots, c_i^k, \ldots, c_{rand}^s)\}$$

여기서 c_{elite}^k는 종족 k의 엘리트 염색체를 나타내고,

c^{k}_{rand}는 종족 k에서 무작위로 선택된 염색체를 나타내며, F는 문제 영역에서 전체 변수에 대한 적합도 함수를 나타낸다. 이와 같이 염색체를 평가할 때 다른 종족에서 무작위로 선택된 염색체들과도 계산함으로써 둘 이상의 변수가 변화 가능하도록 하여 특정 끌림유역을 벗어날 수 있도록 하였다.

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

K. Weicker와, N. Weicker가 CCGA2를 개선한 알고리즘으로, 몇 가지를 제외하고는 CCGA와 같다. 먼저, CCGA에서는 종족이 하나의 변수(단일 종족)만을 진화시키도록 하였으나, ACC에서의 종족은 하나 이상의 변수(복합 종족)를 진화시킬 수 있도록 하였다. 그리고, ACC에서는 의존도 행렬이라는 것을 사용하는데, 의존도 행렬이란 최적화해야 할 변수들에 대하여 각각의 변수들과 나머지 변수들과의 의존도를 나타내는 대칭행렬이다. ACC에서의 종족의 개념은 협력 공진화에서의 종족개념을 확장하여, 최적화해야 할 전체 변수들의 집합을 I라고 하고 t세대의 종족개수가 $s(t)$개 일 때, t세대에 $k(1\le k\le s(t))$번째 종족이 최적화해야 할 변수들을 원소로 갖는 집합 $V_k(\bigcup_{k=1}^{s(t)} V_k = I,\ V_k \cap V_j = \emptyset\ (k\neq j))$의 원소들을 진화시키는 종족을 $S_{V_k}(t)$와 같이 표현하면, ACC에서의 종족 $S_{V_i}(t)$의 i번째 염색체 $c_i^{S_{V_i}(t)}$의 적합도는 다음과 같이 계산된다.

$$F^{ACC}\left(c_i^{S_{V_i}(t)}\right)=\max\left\{\begin{array}{l} F\left(c_{elite}^{S_{V_1}(t)},c_{elite}^{S_{V_2}(t)},\ldots,c_i^{S_{V_i}(t)},\ldots,c_{elite}^{S_{V_m}(t)}\right),\\ F\left(c_{elite}^{S_{V_1}(t)},\ldots,c_{rand}^{S_{V_k}(t)},\ldots,c_i^{S_{V_i}(t)},\ldots,c_{elite}^{S_{V_m}(t)}\right)\end{array}\right\} \quad (6)$$

여기서 $c_{elite}^{S_{V_k}(t)}$는 종족 $S_{V_k}(t)$의 엘리트 염색체를 나타내고, $c_{rand}^{S_{V_i}(t)}$는 종족 $S_{V_i}(t)$에서 무작위로 선택된 염색체를 나타내며, F는 문제 영역에서 전체 변수에 대한 적합도 함수를 나타낸다. max의 첫 번째 항에서는 협력 공진화 알고리즘과 같이 염색체 $c_i^{S_{V_i}(t)}$를 평가하기 위해 다른 종족들의 엘리트 염색체를 이용하여 평가하고, 두 번째 항에서는 다른 종족들은 엘리트 염색체를 유지하고, j번째 종족에서만 무작위로 선택하여 평가한다. 이와 같이 계산할 때, (식 6)의 max에서 두 번째 항이 선택되어 적합도로 될 경우, 의존도 행렬에서 종족 $S_{V_j}(t)$와 $S_{V_k}(t)$사이의 의존도를 나타내는 값을 증가하도록 하여, 이 값이 주어진 값을 초과하면, 종족 $S_{V_j}(t)$와 종족 $S_{V_k}(t)$를 병합하도록 한다. 종족들 사이에 의존성이 없다면 (식 3), (식 4)에 의해 (식 6)의 max에서 항상 첫 번째 항이 적합도로 선택이 될 것이므로 CCGA1의 알고리즘과 같이 진행되며, 둘 이상의 종족이 서로 의존성이 존재한다면, 종족들은 서로 병합되고, 병합된 종족은 일반적인 진화 알고리즘이 적용되므로 병합된 종족이 최적화하는 변수들이 동시에 변화하여 국소 최적해의 끌림유역을 벗어날 수 있게 된다.

ACC에서 종족의 병합은 다음과 같이 이루어진다.

V_j, V_k를 각각 t세대의 j, k번째 종족이 최적화할 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 다음과 같이 결정된다.

$$\mathrm{merge}(S_{V_j}(t), S_{V_k}(t)) = S_V(t) \quad (7)$$

$$\begin{array}{l} S_{V_j}(t)=\langle c_1^{S_{V_1}(t)},\ldots,c_n^{S_{V_1}(t)}\rangle \\ S_{V_k}(t)=\langle c_1^{S_{V_1}(t)},\ldots,c_n^{S_{V_1}(t)}\rangle \end{array} \rightarrow S_V(t)=\langle c_1^{S_V(t)},\ldots,c_n^{S_V(t)}\rangle \quad (8)$$

그리고, $\circ$를 종족의 개체를 병합하는 연산자라고할 때, 여기서 병합된 종족 $S_V(t)$의 i번째 염색체 $c_i^{S_V(t)}$는,

$$c_i^{S_V(t)}=\begin{cases} c_{elite}^{S_{V_1}(t)} \circ c_{elite}^{S_{V_2}(t)} & : i=1\text{일 때,} \\ c_{elite}^{S_{V_1}(t)} \circ c_{rand}^{S_{V_2}(t)} & : 2\le i\le \lceil \frac{n}{2} \rceil \text{ 일 때,} \\ c_{rand}^{S_{V_1}(t)} \circ c_{elite}^{S_{V_2}(t)} & : \text{그 외의 경우} \end{cases} \quad (9)$$

와 같이 결정된다.

다시말하면, 새로운 종족 $S_V(t)$은 $i=1$인 경우(엘리트 염색체)를 제외한 개체군의 1/2는 종족 $S_{V_j}(t)$의 엘리트 부분해와 종족 $S_{V_k}(t)$에서 무작위로 선택된 부분해로 이루어진 염색체로 구성되고, 나머지 1/2는 종족 $S_{V_j}(t)$에서 무작위로 선택된 부분해와 종족 $S_{V_k}(t)$의 엘리트 부분해로 이루어진 염색체로 구성된다.

3. SMGA: Split & Merge Genetic Algorithm

ACC의 경우는 모든 변수들이 상호 의존성을 갖고 있는 문제에 적용될 경우 모든 종족들이 병합됨으로써, 진화 중반 이후로는 공진화의 잇점인 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 끌림유역에 존재할 경우와, 또는 엘리트가 전역 최적해의 끌림유역에 존재할 경우이다. 만약, 국소 최적해의 끌림유역을 탐색하고 있다면 계속 병합되어 있는 것(복합 종족)이 효율이 좋겠지만, 이미 국소 최적해의 끌림유역을 벗어나서 전역 최적해의 끌림유역을 탐색 중 이라면 두 종족이 병합되어 있는 것 보다는 다시 분할되는 것(단일

종족)이 효율이 좋아진다. 그러나 이같은 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.

3.1 종족 병합

병합단계에서는 ACC에서의 방법과 비슷하다. 즉, $V = V_j \bigcup V_k$일 때, 종족 $S_{V_j}(t)$와 $S_{V_k}(t)$가 병합된 종족을 $S_V(t)$라 하면, 병합된 종족 $S_V(t)$의 개체군은 (식 7), (식 8)과 같이 결정된다. 여기서 새로운 종족 $S_V(t)$의 i번째 염색체 $c_i^{S_V(t)}$는,

$$c_i^{S_V(t)} = \begin{cases} c_{elite}^{S_{V_1}}(t) \circ c_{elite}^{S_{V_2}}(t) & : i=1\text{일 때}, \\ c_{elite}^{S_{V_1}}(t) \circ c_{rand}^{S_{V_2}}(t) & : 2 \le i \le \lceil \frac{n}{3} \rceil \text{일 때}, \\ c_{rand}^{S_{V_1}}(t) \circ c_{elite}^{S_{V_2}}(t) & : \lceil \frac{n}{3} \rceil < i \le \lceil \frac{2n}{3} \rceil \text{일 때}, \\ c_{rand}^{S_{V_1}}(t) \circ c_{rand}^{S_{V_2}}(t) & : \text{그 외의 경우} \end{cases} \quad (10)$$

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 10)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.

종족의 병합은 탐색 공간의 축소 측면에서 봤을 때는 비 효율적이나, 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.

3.2 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, 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 $P=\{U_1, U_2, \ldots, U_m\}$ ($\bigcup_{i=1}^{m} U_i = V_k$, $U_i \bigcap U_j = \emptyset$ $(i \neq j)$)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 다음과 같이 결정된다.

$$\text{split}(S_{V_k}(t), P) = \{S_{U_1}(t), S_{U_2}(t), \ldots, S_{U_m}(t)\} \quad (11)$$

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

$$c_j^{S_{U_i}(t)} = \text{proj}\left(c_j^{S_{V_k}(t)}, U_i\right) \ : \ 1 \le i \le n, \ 1 \le j \le m \quad (12)$$

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 여러가지 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 분할 하도록 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.

종족 분할의 영향으로는 탐색 공간의 축소로 인한 빠

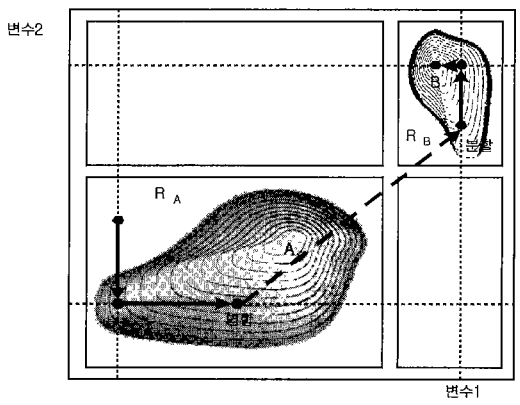

그림 7 SMGA의 진화과정

른 진화의 잇점과, 그로 인한 국소 최적해에 빠질 수 있는 단점이 있다. 그러나 종족 병합 단계에서 국소 최적해를 벗어났다고 가정되므로 결과적으로는 진화속도를 향상시키는 잇점이 더 많다고 할 수 있다.

이와 같이 분할과 병합을 병행하면 <그림 7>과 같이 진화가 진행된다. 그림에서 극값 A와 B가 존재할 때 B가 전역 최적해 라면, <그림 7>에서 국소 최적해의 끌림유역인 R_A영역에서 진화가 시작되어 변수 의존성을 인식하고 종족이 병합되기 전 까지 CCGA1과 같이 진화가 이루어 진다. 그러다가 의존성을 인식한 시점 두 종족이 병합되어 전역최적해의 끌림유역인 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.

4. 실 험

실험에서는 함수 최적화 문제에 대하여, 벤치마크 함수(benchmark function)로 많이 사용되는 Ackley 함수, Rosenbrock 함수, Schwefel 함수들에 대하여 실험하였으며, 기존의 방법들과 비교하기 위하여 Rosenbrock함수를 제외한 각 벤치마크 함수들에서 변수 의존성이 존재하는 경우와 존재하지 않는 경우에 대하여 실험하였다. 즉, 각각 함수 원형의 경우와 변수 의존성을 위하여 축 회전(coordinate rotation)[6]을 한 경우에 대하여 실험하였다. 그리고, 실제 문제에 대한 실험으로 상품 재고 제어문제(inventory control problem)를 사용하였다.

벤치마크 함수들에 대한 실험에서는 모두 <표 1>과 같은 매개변수를 사용하여 실험하였다.

표 1 실험에 사용된 매개변수

매개변수	값
개체군 크기	100
변수당 표현 bit수	16
교배확률	0.6
돌연변이확률	1/염색체길이
선택방법	적합도비례선택
교배방법	2점교배
실험회수	10회

그래프로 나타낸 각 알고리즘의 성능은 10회의 실험에서 얻은 값을 평균하여 나타낸 것이다.

4.1 Ackley 함수

Ackley함수는 자체만으로는 변수간에 의존성이 존재하지 않는다. 따라서 축 회전(coordinate rotation)을 하여 변수들 간에 의존성이 생기게 하였으며 최적화하는 변수를 30개로 하여 탐색 공간을 확장 하였다.

Ackley 함수의 원형은 (식 13)과 같다.

$$F(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e, \quad -20 \le x_i \le 20 \tag{13}$$

이 함수에서 전역 최적해는 $\vec{x}=\{0,0,\ldots,0\}$일 때, $F(\vec{x})=0$ 이다.

실험 결과를 보면, 변수축 회전을 하지 않은 경우에는 기존의 진화 알고리즘을 제외하고는 대부분 빨리 진화되어 전역 최적해에 가까워 진다. 그러나 ACC의 경우 진화 초기에 몇 개의 종족이 병합되어 중간부터는 진화 속도가 감소하는 것을 볼 수 있다.

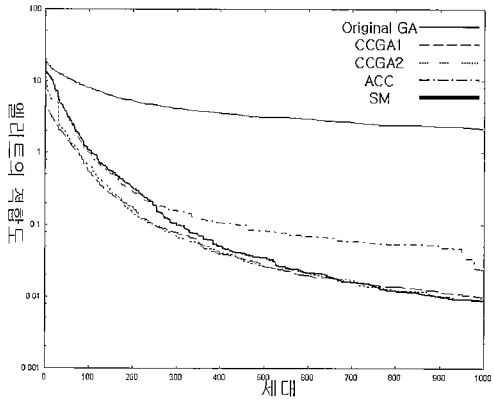

그림 8 Ackley함수 원형에 대한 실험결과

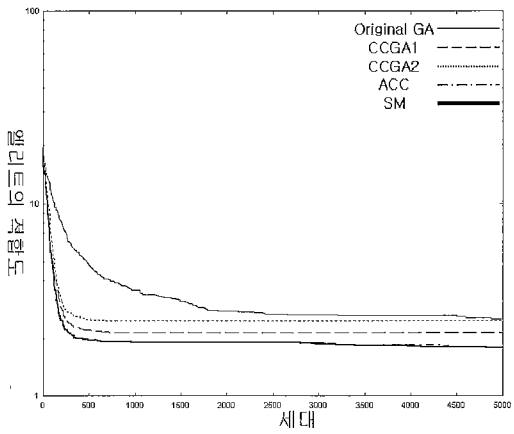

그림 9 변수축을 회전시킨 Ackley함수에 대한 실험결과

변수축 회전의 경우 일반적인 진화 알고리즘은 변수축을 회전하기 전과 거의 차이가 없는 진화양상을 보이지만, CCGA1, CCGA2, ACC, SMGA 알고리즘들은 회전 전 보다 훨씬 효율이 떨어진 것을 알 수 있다. CCGA1은 국소 최적해에서 Nash 평형이 이루어지고 있으며 CCGA2는 CCGA1보다는 좋지만 역시 국소 최적해에 빠져 있다. 그리고, ACC와 SMGA는 종족의 병합이 있기 전 까지는 유사하게 진행되다가 종족의 병합 후에는 약간의 차이를 보인다.

4.2 Rosenbrock 함수

Rosenbrock 함수는 좌표축 회전이 없어도 자체적으로 약간의 변수 의존성을 가지고 있는 함수이다. 함수의 형태는 (식 14)와 같다.

$$F(\vec{x}) = \sum_{i=1}^{n/2}[100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1}^2)^2], \quad -2.048 \le x_i \le 2.048 \tag{14}$$

이 함수에서 변수축을 회전하지 않았을 경우 전역 최적해는 $\vec{x} = \{1, 1, \ldots, 1\}$일 때, $F(\vec{x}) = 0$이다. 이 함수의 특이한 점은 $0 < i \le n/2$일 경우 변수 x_{2i}와 x_{2i-1}들 사이에만 변수 의존성이 존재한다는 것이다.

실험 결과를 보면 변수축을 회전하지 않은 경우는 CCGA1은 Ackley함수의 경우와 마찬가지로 국소 최적해에서 Nash평형을 이루고 있고, CCGA2의 경우는 CCGA1의 경우보다는 좋지만, 정확히 어떤 변수와 어떤 변수가 서로 의존성이 있는지 알지 못함으로써 빠른 진화가 이루어지지 않고 있다. 이에 반해 ACC와 SMGA는 의존성이 있는, 변수 x_{2i}와 x_{2i-1}를 표현하는

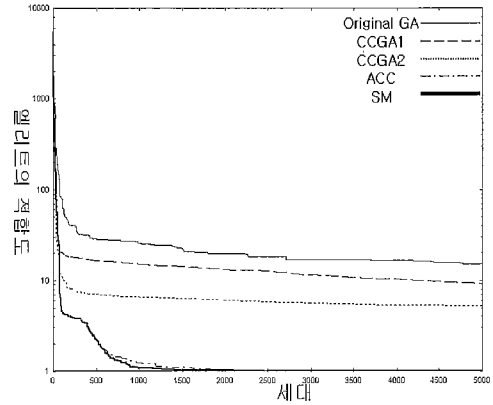

그림 10 Rosenbrock함수 원형에 대한 실험 결과

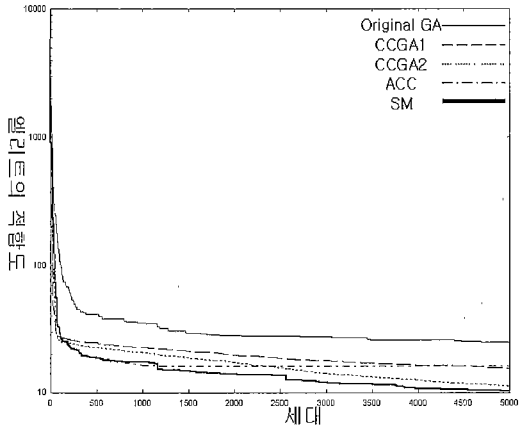

그림 11 변수축을 회전시킨 Rosenbrock함수의 실험결과

종족쌍을 병합함으로써 빠른 진화를 보이고 있다. 변수축을 회전한 경우에는 정확히 어떤 변수와 어떤 변수가 의존성을 가지는지 확실하지 않고, 대부분의 변수들이 서로 의존성을 가지게 되므로, 모든 알고리즘들이 빠른 진화를 하지 못한다. 특히 ACC의 경우 초기에 모든 종족들이 병합되어 버림으로써 1000세대 이후로는 거의 진화가 이루어지지 않고 있다. 이에 반해 SMGA는 CCGA2와 같이 지속적인 진화를 하는 것을 볼 수 있다.

4.3 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 15)과 같다.

$$F(\vec{x}) = \sum_{i=1}^{n}[-x_i \cdot \sin(\sqrt{|x_i|})], \quad -500 \le x_i \le 500 \tag{15}$$

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.

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.

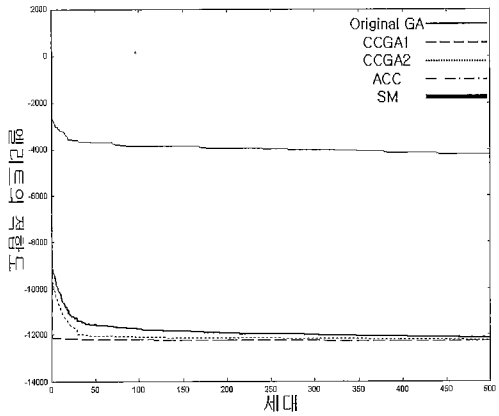

그림 12 Schwefel함수 원형에 대한 실험결과

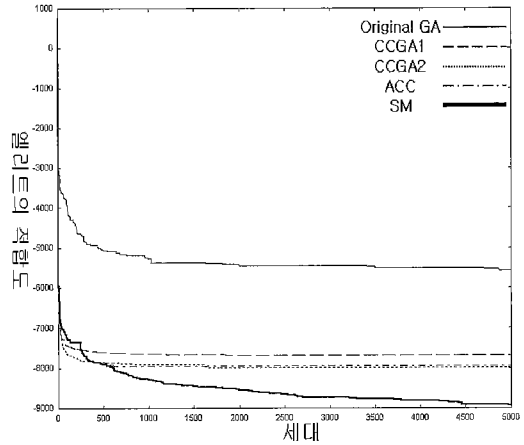

그림 13 변수축을 회전시킨 Schwefel함수에 대한 실험결과

4.4 ICP(Inventory Control Problem)[10]

본 논문에서는 실제 문제 적용으로 상품 재고 제어문제(ICP)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 옮길때도 비용이 들고, 생산지에서 창고까지 옮기는데 걸리는 시간도 고려한다. 즉 상품 요청을 하고 옮기는동안 소비자가 와서 물건을 살 수도 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 상품의 비용 만큼의 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. 실험 데이타는 10가지의 상품 각각에 대하여 한 번의 시간에 0~5개까지 팔릴 수 있도록 무작위로 생성하였고, 데이타 개수는 1000개로 하였다.

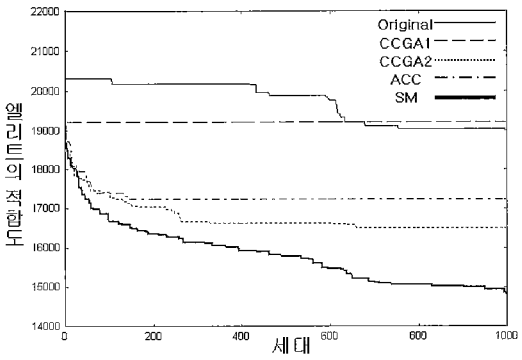

그림 14 상품재고 제어 문제의 실험결과

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 보

다는 좋지만, 변수 의존성을 제대로 반영하지 못함으로 인해서 그다지 좋은 결과를 보이지 못하였다. 그러나 SM의 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 제시하였다. 실험 결과 제시한 알고리즘이 대상이 된 실험 함수들에 대하여 기존의 알고리즘보다 좋은 성능을 보였다. Potter와 DeJong이 제안한 CCGA1은 Nash 평형점이 많은 문제들에 대해서는 그 중의 하나의 Nash 평형점으로 수렴하여 다른 점들을 탐색하지 못함으로써 그다지 좋은 성능을 보이지 못하였고, CCGA2는 CCGA1의 이런 문제점을 개선 하였으나 변수 의존성을 진화상황에 제대로 반영하지 못함으로써 그다지 좋은 성능을 보이지 못하였다. 그리고 ACC는 진화 도중 병합만을 고려함하여 진화 말기에는 모든 종족이 하나로 병합되어 진화속도가 저하됨으로써 좋은 결과를 보이지 못하였다. 그러나 본 논문에서 제안한 SMGA는 변수 의존성을 고려하면서도 진화 도중 병합과 분할을 반복함으로써 ACC의 문제점을 해결하여 다른 알고리즘들 보다 좋은 성능을 보여주었다.

현재 벤치마크 함수들과 ICP에 대해서 실험을 하였으나 제안하는 알고리즘의 타당성과 일반성을 검증하기 위해서 보다 많은 변수 의존성이 높은 여러가지 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.

참 고 문 헌

[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), 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] Salomon, R. (1996). Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions. *BioSystems 39*, 210-229

[7] Michalewicz, Z. (1995). Genetic Algorithms + Data Structures = Evolution Programs. Third, Extended Edition. Springer-Verlag.

[8] 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.

[9] 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.

[10] 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.

박 성 진

1998년 숭실대학교 인공지능학과 졸업. 2000년 8월 숭실대학교 대학원 컴퓨터학과 석사학위 취득. 현재 (주)전능메디칼 소프트웨어 연구원 재직중. 관심분야는 신경망, 유전자 알고리즘, 인공생명 등

김 명 원

1972년 서울대학교 응용수학과 졸업 1981년 University of Massachusetts(Amherst) Computer Science 석사 학위 취득. 1986년 University of Teax(Austin) Computer Science 박사 학위 취득. 1975년 ~ 1978년 한국과학기술연구소 연구원. 1982년 ~ 1985년 institute for Computing Science & Computer Applications(Univ. of Texas) 연구원. 185년 ~ 1987년 AT&T Bell Labs.(Naperville)Member of Technical Staff. 1987년 ~ 1994년 한국전자통신연구소 책임연구원. 1991년 ~ 1993년 충남대학교 전자계산학과 겸임부교수. 1994년 ~ 현재 숭실대학교 컴퓨터학부 부교수. 1992년 ~ 1993년 한국신경회로망 연구회 회장. 1993년 ~ 1995년 정보과학회 뉴로컴퓨팅 연구회 위원장. 1993년 ~ 현재 IEEE Neural Network Council 한국지부장. 1993년 ~ 현재 한국정보과학회 뉴로컴퓨팅 연구회 회장. 1997년 ~ 현재 한국뇌과학회 부회장. 1998년 ~ 현재 한국인지학회 부회장. 관심분야는 유연추론, 신경회로망, 퍼지시스템, 진화알고리즘, 패턴인식, 자동추론, 기계학습, 데이터 마이닝, creativity engineering 등.