Journal of the Korean Institute of Industrial Engineers
Vol. 26, No. 2, pp. 101-109, June 2000.

토너먼트 경쟁에 의한 경쟁 공진화 알고리듬†

김선진[1] · 김여근[1] · 김재윤[1] · 곽재승[2]

[1]전남대학교 산업공학과 / [2]순천제일대학 품질경영과

A Competitive Coevolutionary Algorithm with Tournament Competitions

Sun-Jin Kim[1] · Yeo-Keun Kim[1] · Jae-Yun Kim[1] · Jai-Seung Kwak[2]

A competitive coevolutionary algorithm is a probabilistic search method that imitates the biological process that two or more species competitively coevolve through evolutionary arms race. The algorithm has been used to efficiently solve adversarial problems that can be formulated as the search for a solution that is correct over a large space of test cases. We develop an efficient competitive coevolutionary algorithm to solve adversarial problems with high complexity. The algorithm developed in this paper employs three methods: tournament competitions, exchanging of entry fee, and localized coevolution. Analyzed in this paper are the effects of the methods on the performance of the proposed algorithm. The extensive experiments show that our algorithm can progress an evolutionary arms race between competitive coevolving species and then outperforms existing approaches to solving the adversarial problems.

1. 서 론

일반적으로 실세계에서 발생하는 문제들은 복잡도가 대단히 높다. 한 종(species)의 진화과정을 모방한 단순진화 알고리듬에 의해 복잡하고 동적인 문제를 해결하고자 할 경우 흔히 국소최적에 조기수렴하여 전체최적에 도달하지 못하는 경우가 발생된다. 이를 해결하기 위해 1990년 이후 진화 알고리듬에 기반을 둔 공진화 알고리듬(coevolutionary algorithm)이 개발되기 시작하였다. 공진화 알고리듬은 두 종 이상이 상호작용하고 적응하면서 변화하는 생물의 공진화과정을 모방한 일종의 탐색기법이다(Koza, 1992). 이 기법은 복잡하고 동적인 문제를 해결하는 하나의 방법론으로 인식되어 지고 있다(Moriarty and Miikkulainen, 1997).

공진화모형은 기생(parasitism)과 공생(symbiosis)관계를 갖는 생물의 진화과정을 모방한 숙주-기생충(host-parasite)모형과 공생모형으로 크게 분류할 수 있다. 숙주-기생충모형은 다른 종과 적대(기생)관계를, 공생모형은 상호협조(상리공생)관계를 모형화한 것이다. 숙주-기생충모형은 Hillis(1991)에 의해 처음 제안되었다. 그는 두 종을 경쟁적으로 진화시킴으로써 하나의 종을 진화시키는 것보다 더 좋은 해를 얻을 수 있음을 정렬망(sorting network) 문제를 사용하여 보였다. 그 이후 경쟁진화과정을 모방한 여러 연구가 진행되어 오고 있다(Holmes *et al.*, 1995; Olsson, 1996; Pagie and Hogeweg, 1998; Paredis, 1995). 이 모형은 게임형태의 문제나 소프트웨어 개발단계에서와 같이 테스팅하는 문제 등에 적용되었다.

한편 공진화하는 종간에 협조관계를 유지하면서 진화하는 과정을 모방한 공생모형은 Husband와 Mill(1991)이 생산계획문제에 적용하면서 시작되었다. 이 모형에 관한 연구로는 Bull *et al.* (1995), Potter (1997), Moriarty와 Miikkulainen(1997) 등의 연구가 있다. 이 모형은 관련성이 있는 여러 부분 문제들로 이루어진 문제를 해결하는 데 주로 사용되고 있다.

이 연구의 목적은 복잡성이 높은 경쟁관계 문제를 효율적으로 해결할 수 있는 경쟁 공진화 알고리듬의 개발에 있다. 경쟁(적대)관계 문제는 일반적으로 게임전략, 테스팅을 필요로 하는 문제 및 제약을 갖는 문제와 같이 해결하고자 하는 문제의

† 이 연구는 한국과학재단 특정기초연구(과제번호 : 98-0200- 09- 01-3) 지원으로 수행되었음.

잠재해와 이러한 해에 대한 상대자(예로, 게임에서의 상대방, 테스트문제, 제약조건 등)가 경쟁의 구조로 정의되는 문제를 말한다. 구하는 문제의 잠재해와 이 해에 대한 상대자와의 테스트를 통해 점진적으로 좋은 해를 탐색해 가는 방법은 자연계에서 숙주와 이에 기생하는 기생충이 계속적으로 상호변화에 적응하며 공진화하는 현상과 유사하다. 경쟁 공진화에서 어느 한 종이 일방적으로 우세하게 되면, 상대의 진화를 유도할 수 있는 무기경쟁이 깨져서 결국 상호진화가 멈추게 되고, 이는 조기수렴의 결과를 초래할 수 있다. 따라서 경쟁 공진화 알고리듬은 다양하고 동적인 환경(해 영역)에서도 진화적 무기경쟁이 지속될 수 있도록 설계되어야 한다.

이 연구에서는 효율적인 경쟁 공진화 알고리듬의 개발을 위하여 다음과 같은 방법을 채용한다. 첫째, 상대 모집단을 효과적으로 테스트할 수 있는 경쟁전략으로 토너먼트 경쟁을 도입한다. 둘째, 높은 적응력을 갖는 개체로 진화할 수 있는 개체들을 유지시키기 위해 경쟁상대와의 상대적인 척도 즉, 경쟁개체의 난이도를 고려한 적응도 평가전략을 사용한다. 이 평가방법을 참가비 교환이라 한다. 셋째, 두 종간의 경쟁과 진화는 이웃단위로 이루어진다. 이러한 경쟁과 진화는 개체의 다양성을 유지하여 궁극적으로, 두 종간의 진화 불균형을 방지하여 공진화가 지속될 수 있도록 하기 위함이다.

제안한 알고리듬의 효율성과 활용성을 보이기 위하여 기존 경쟁 공진화 알고리듬에 관한 연구에서 알고리듬 성능 분석을 위해 사용된 예제 문제를 대상으로 실험한다. 기존 연구들에서 사용된 적용문제는 진화목표에 따라 크게 두 가지로 분류할 수 있다. 하나는 진화목표가 현재 상대의 행동에 완전히 의존하여 결정되는 동적(dynamic)인 특성을 갖는 문제(Angeline and Pollack, 1993; Koza, 1992; Rosin and Belew, 1997)이며, 다른 하나는 진화목표가 미리 주어지고 정적(static)인 특성을 갖는 문제(Hillis, 1991; Paredis, 1995)이다. 본 연구에서는 동적인 특성을 갖는 문제로 Nim게임 문제, 정적인 특성을 갖는 문제로 정렬망 문제를 사용한다. 본 연구에서는 두 문제에 제안한 알고리듬을 적용한 실험결과를 보이고 이를 분석함으로써, 제안한 알고리듬이 특성이 서로 다른 문제에 효율적으로 적용될 수 있음을 보이고자 한다.

2. 제안한 경쟁 공진화 알고리듬

경쟁 공진화 알고리듬은 공진화하는 종간에 균형적인 상호진화가 이루어질 수 있도록 하여야 한다. 따라서, 경쟁 공진화 알고리듬에 관한 기존 연구들은 숙주와 기생충 종간에 진화의 순환 또는 불균형을 방지하면서 지속적인 무기경쟁을 유도하기 위하여 다음 세 가지 측면에서 여러 방법들이 제안되었다. 첫째, 경쟁상대 선택전략에 관한 연구로 Koza(1992)는 상대 모집단 전체, Pagie와 Hogeweg(1998)는 상대 모집단의 이웃, 그리고 Rosin과 Belew (1997)는 상대 모집단에서 샘플링한 일부 개체들과 경쟁하는 방법을 제안하였다. 둘째, 개체의 적응도 부여 방법에 관한 연구로 Smith *et al.*(1993)은 개체들간의 유사성, Rosin과 Belew (1997)은 경쟁상대의 난이도를 고려하여 적응도를 부여하는 방법을 개발하였다. 셋째, Rosin과 Belew(1997), 그리고 Nolfi(1998)는 진화과정 동안 발견한 강한 개체를 보존하는 전략에 관하여 연구하였다. 제안한 경쟁 공진화 알고리듬에서는 다음에서 설명할 방법들을 모두 결합하여 알고리듬의 효율성을 높이고자 한다.

2.1 토너먼트 경쟁

경쟁 공진화 알고리듬에서는 상대 모집단과의 경쟁 결과에 따라 개체의 적응도가 달라지므로, 어떤 개체와 어떻게 경쟁하느냐는 중요한 문제이다. 만약, 상대 모집단에 있는 개체에 대한 특별한 정보없이 그들 모두와 경쟁하여 개체의 적응도를 부여하면, 일부 개체와 경쟁하여 부여된 적응도보다 정확한 적응도 정보를 제공할 가능성은 클 수 있으나, 계산소요시간이 많이 요구된다. 반면, 상대 모집단의 일부 개체들을 샘플링하여 경쟁하는 방법은 알고리듬의 계산시간에 대한 부담을 줄일 수 있으나, 샘플링에서 발생하는 오류로 인하여 실제 그 개체의 적응도나 잠재력을 잘못 판단할 가능성이 있으므로 반드시 좋은 해로의 탐색을 보장하기 어렵다.

따라서 이 연구에서는 두 모집단의 개체를 경쟁시키는 방법으로 토너먼트 경쟁을 사용한다. 이 경쟁방법은 단일 모집단에서 토너먼트 적응도를 이용하여 개체의 적응도를 결정하는 Angeline과 Pollack(1993)의 연구에서 착안하여 경쟁 공진화 알고리듬에 맞도록 적절히 변형한 것이다. 토너먼트 경쟁은 전체 경쟁을 통하여 개체의 적응도 순위를 완전히 결정하는 방법에 비해 좋은 개체를 빨리 파악할 수 있고, 이러한 개체들은 좋은 해로의 이동을 빠르게 유도할 수 있을 것으로 기대된다. 그리고 일단 패배한 개체들은 더 이상 경쟁을 할 수 없지만, 동일한 단계에서 패배한 개체들 사이의 선별을 완전히 임의적으로 하도록 하여 모집단의 다양성도 높일 수 있을 것으로 기대된다.

토너먼트 경쟁은 다음과 같은 절차에 의해 이루어진다. 먼저, 숙주와 기생충을 짝지어 경쟁시킨다. 숙주 또는 기생충 중에서 어느 한 쪽 집단이 모두 패배하게 되면 끝내고, 그렇지 않으면 경쟁에서 패배한 숙주와 기생충의 제거와 생존한 숙주와 기생충의 경쟁을 반복한다. 단, 숙주와 기생충은 임의로 짝지어 경쟁시키며, 짝지어지지 않는 숙주 또는 기생충은 경쟁에 참여하지 않으나 생존해 있는 것으로 본다. <그림 1>은 숙주와 기생충이 각각 5개씩 있을 때 토너먼트 경쟁의 예를 보인 것이다.

2.2 참가비 교환에 의한 적응도 부여

경쟁 공진화 알고리듬에서는 초기에 낮은 적응도를 갖는 개

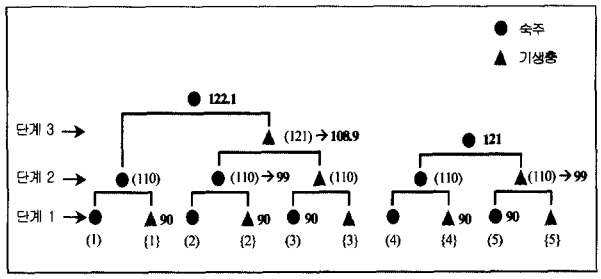

그림 1. 토너먼트 경쟁과 참가비 교환에 의한 적응도 부여.

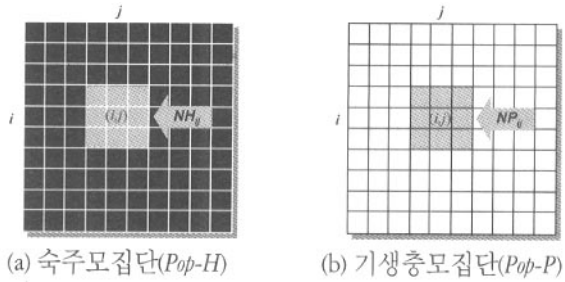

그림 2. 숙주모집단과 기생충모집단.

체라도 진화가 진행되면서 새로운 상대에게는 높은 적응도를 부여받을 수 있으므로, 단순진화 알고리듬과는 달리 낮은 적응도를 갖는 개체의 제거에 신중할 필요가 있다. 또한 과거에 상대 개체에 의해 패배되었던 개체가 현 세대에 다시 출현하면 상대 개체는 이를 새로운 개체로 인식하고, 이에 적응하고 진화하려는 전략을 채택함으로써 공진화 과정에서 순환현상이 발생할 수 있다(Nolfi, 1998). 이와 관련하여 본 연구에서는 참가비 교환에 의한 적응도 부여방법을 사용한다.

참가비 교환에 의한 적응도 부여방법이란 각 개체가 갖는 적응도의 일부를 가지고 경쟁에 참여하고, 경쟁 결과에 따라 승리하면 상대 개체의 참가비를 적응도로 획득하고, 그렇지 않으면 자신의 참가비를 상대에게 빼앗기는 방식으로 적응도를 부여하는 방법이다. 이 방법은 게임에서 참가비의 개념과 유사하며, 토너먼트 경쟁과 함께 사용하면 동일한 승리횟수를 가진 개체라도 경쟁상대에 따라 적응도가 다르게 부여되어 적응도 배분 효과를 갖게 된다.

예를 들어, <그림 1>에서 참가비 교환에 의한 적응도 부여 과정과 효과를 보기로 하자. 모든 개체의 초기 적응도를 100, 경쟁의 참가비를 자신의 적응도의 10%로 가정하자. 단계 1에서 숙주 (3), (5)와 기생충 {1}, {2}, {4}는 패배하게 되어 자신의 초기 적응도로부터 참가비 10을 잃게 되고, 승리한 개체들은 참가비 10을 얻게 된다. 단계 2에서는 숙주 (2)와 기생충 {5}가 패배하여 자신의 적응도 110으로부터 참가비 11을 잃게 되고, 승리한 개체들은 그만큼의 참가비를 획득하여 적응도가 121이 된다. 단계 3의 결과도 동일한 방법에 의한 것이다. 여기서, 참가비의 개념을 사용하지 않는다면, 숙주 (1)과 숙주 (4)는 모두 2회 승리하여 동일한 적응도를 부여받게 되지만, 참가비 교환에 의한 적응도 부여방법은 같은 승리횟수를 가진 개체라도 경쟁상대에 따라 다르게 적응도가 부여될 수 있다.

2.3 이웃진화전략

진화 알고리듬에서 한 세대에 모집단 전체가 동시에 진화하지 않고, 매 세대마다 모집단에서 일부분만을 선택하여 진화하는 이웃진화는 모집단의 다양성을 유지하고, 알고리듬의 성능을 높이는 데 공헌하는 것으로 알려져 있다(Pagie and Hogeweg, 1998). 또한 경쟁 공진화 알고리듬에서 이웃단위의 경쟁 및 진화는 매 경쟁마다 새로운 이웃이 선택되므로, 경쟁상대를 선택하기 위한 특별한 방법을 요구하지 않는다는 특징을 갖는다.

이웃진화를 위하여 숙주 모집단(*Pop-H*)과 기생충 모집단(*Pop-P*)은 <그림 2>와 같이 2차원 격자구조로 구성되며 토러스(torus) 형태를 갖는다고 본다. 토러스는 두 개의 반대 모서리(edge)가 서로 연결되어 있는 구조를 말한다. 두 모집단에서 위치 (i, j)에 있는 개체의 이웃을 각각 NH_{ij}, NP_{ij}로 정의하자. 예를 들어, 3×3의 이웃구조를 사용한다면 <그림 2>에서 보는 바와 같이 개체 (i, j)와 주위 8개의 개체들이 이웃에 포함된다. 모집단이 토러스 형태이므로, 모든 위치에 있는 개체들은 동일한 개수의 개체를 포함하는 이웃을 갖는다. 이웃은 크기나 형태에 따라 다양한 방법으로 정의할 수 있으나, 이 연구에서는 정사각형 형태의 이웃구조를 사용한다.

2.4 알고리듬의 절차

이 연구에서 개발한 경쟁 공진화 알고리듬의 절차는 다음과 같다.

단계 1 (초기화)
숙주개체와 기생충개체로 이루어진 두 모집단 *Pop−H*와 *Pop−P*를 구성한다.

단계 2 (이웃설정)
임의의 위치(i, j)를 선택하고, 두 모집단의 이웃 NH_{ij}, NP_{ij}를 정의한다.

단계 3 (경쟁에 의한 적응도 평가)
3.1 NH_{ij}, NP_{ij} 내의 모든 개체에게 초기 적응도를 동일하게 부여한다.
3.2 토너먼트 경쟁을 통하여 각 개체의 적응도를 평가한다.

단계 4 (NH_{ij}와 NP_{ij}의 진화)
4.1 $EN = NH_{ij}$, $F = 0$으로 둔다.
4.2 EN에서 단계 3.2의 결과에 의해 가장 높은 적응도를 보유한 개체를 선택하여 보관한다.
4.3 EN에서 적응도에 따라 재생산에 참여할 개체를 선별한다.
4.4 선별된 개체들에 대하여 교차율과 돌연변이율에

따라 유전연산을 적용하여 자손개체를 생산한다.

4.5 단계 4.2에서 선택된 개체를 EN에 있는 임의의 개체와 대체한다.

4.6 만약, $F=0$이면, $F=1$, $EN=NP_{ij}$로 두고, 단계 4.2로 간다. 그렇지 않으면 단계 5로 간다.

단계 5 (종료조건)

알고리듬의 종료조건을 만족하면 끝내고, 그렇지 않으면 단계 2로 간다.

단계 2는 진화 중심인 위치 (i, j)를 선택하고, 이웃을 정의하는 단계이다. 단계 3에서는 토너먼트 경쟁을 통해 개체의 적응도 평가가 이루어진다. 이때 개체들이 진화하는 동안 적응도를 유지하면, 자손개체의 적응도 결정에 어려움이 따른다. 또한 상대적으로 승리하기 쉬운 경쟁상대를 만나 높은 적응도를 부여받았거나 그렇지 못한 경우로 낮은 적응도를 부여받았다면, 왜곡된 적응도를 보상해 줄 방법이 명확하지 않다. 따라서 이 연구에서는 현재 진화되는 시점만을 고려하여 적응도를 평가하도록 하기 위하여 단계 3.1에서 개체들에게 동일한 적응도를 부여하도록 하였다. 단계 3.1에 의하여 위에서 언급한 단점을 극복할 수 있으며, 이는 실세계에서 게임에 참가하는 참가자들은 초기에 동등한 조건에서 게임을 시작한다는 개념과 모순되지 않는다. 단계 4는 두 모집단이 이웃단위로 진화하는 과정이다. 단계 4.1에서 NH_{ij}와 NP_{ij}에서 가장 높은 적응도를 갖는 개체를 선택함으로써 Rosin과 Belew (1997)에 의해 제안된 우수개체 보존전략과 유사한 효과를 보일 수 있다. 기존 연구에서는 우수개체를 보존하기 위하여 일정크기의 집합을 두었다. 그러나 제안한 알고리듬은 이웃진화로 인하여 명시적으로 우수개체 보존전략을 사용하지 않으면서도 모집단의 관점에서 보면 복수개의 우수개체가 존재하는 효과를 갖는다.

3. Nim게임에의 적용

제안한 경쟁 공진화 알고리듬의 특성을 분석하기 위해 Nim게임문제에 이를 적용하고, 기존 알고리듬과 성능을 비교 분석한다.

3.1 문제정의와 개체표현

3.1.1 문제설명

하나의 돌더미는 여러 개의 돌로 구성되어 있다. 이러한 돌더미가 다수 개 있다고 하자. 두 참가자는 번갈아 가면서 돌더미 중 하나의 돌더미에서 1개 이상의 돌을 가져간다. 게임의 승자는 마지막 남은 돌을 가져간 참가자가 된다. 이 연구에서는 각각 3, 4, 5, 4개의 돌로 구성된 4개의 돌더미로부터 게임을 시작한다고 본다. 적용문제는 이러한 게임에서 첫번째 참가자의 최적 전략을 찾는 것이다.

3.1.2 개체표현

첫번째 참가자의 전략을 숙주개체로 표현하고, 두번째 참가자의 전략을 기생충개체로 표현하였다. 첫번째 참가자와 두번째 참가자의 전략을 각각 숙주와 기생충 모집단으로 구성하였고, 평가 순서만 다를 뿐이지 두 참가자에 대한 전략의 표현은 동일하다.

개체에서 각 인자(gene)의 위치는 참가자가 한번의 행동을 취한 후 4개의 돌더미들의 상태 즉, 각 돌더미에 있는 돌의 남은 개수를 뜻한다. 따라서 모든 가능한 경우의 수는 599($4\times5\times6\times5-1$)가지이므로 개체는 599 bits로 표현된다(Rosin and Belew, 1997). 총 경우의 수에서 1을 감한 이유는 행동을 취한 후의 돌의 상태를 나타내므로, 초기 상태 즉, (3, 4, 5, 4)의 상태는 발생되지 않기 때문이다. 개체의 각 인자는 0 또는 1로 표현되고, 현재 취하는 행동에 대한 평가는 게임의 다음 회에 어떤 행동을 취하느냐에 따라 결정된다. 현재 하나 이상의 돌이 남아있는 돌더미($k>0$)들 중에서 차례로 하나씩 선택하고, 선택된 돌더미에서 0부터 $(k-1)$개의 돌이 남아있는 상태를 의미하는 인자를 순서대로 조사하여 처음으로 1이 나타나는 인자의 상태를 다음 행동으로 취한다. 만약 조사했던 어떠한 인자에서 1이 발견되지 않으면, 평가했던 것 중에서 첫번째 행동을 취한다.

예를 들어, 현재의 상태가 (2, 1, 0, 2)인 경우, 참가자가 다음 행동으로 취할 수 있는 모든 상태 즉, (0, 1, 0, 2), (1, 1, 0, 2), (2, 0, 0, 2), (2, 1, 0, 0), (2, 1, 0, 1)은 다음 행동의 후보가 된다. 그리고 이러한 상태를 의미하는 인자의 값이 처음으로 1을 나타내는 경우를 참가자가 취하는 다음 행동으로 본다. 만약, 위의 다섯 가지 상태를 나타내주는 인자위치의 인자값이 각각 0, 0, 1, 1, 0이라면 처음으로 1이 나타난 세번째의 경우인 (2, 0, 0, 2)가 현재의 상태 (2, 1, 0, 2)의 다음 상태라고 해석된다. 이 행동은 참가자가 현재의 상태로부터 두번째 돌더미에서 1개의 돌을 가져가는 것을 뜻한다. 만약 평가 가능한 상태의 모든 인자에서 1이 발견되지 않았다면, 첫번째 전략 (0, 1, 0, 2)을 취하게 된다.

3.2 실험설정

3.2.1 유전 연산자와 파라미터 설정

두 모집단에서 개체들은 이점교차(two-point crossover)를 통해 진화한다. 그리고 개체 돌연변이율에 의해 개체들을 선택하고, 선택된 개체에 대하여 인자 돌연변이율에 의해 인자를 돌연변이하는 방법을 두 모집단에 적용하였다.

파라미터는 예비실험을 통하여 비교적 좋은 성능을 보인 결과로 두 모집단에 동일하게 설정하였다. 첫째, 모집단 크기는 25×25의 격자구조로 두었다. 둘째, 이웃은 3×3의 격자구조를 사용하였다. 셋째, 토너먼트선별을 사용했으며, 토너먼트의 크기는 2로 하였다. 넷째, 교차율은 0.7, 그리고 개체 돌연변이율은 0.1, 인자 돌연변이율은 0.4로 두었다. 다섯째, 각 개체의 초기 적응도는 100으로 하고, 경쟁에 대한 참가비는 자신의 적응도의 10%로 하였으며, 토너먼트 경쟁의 횟수는 1회로 하

였다. 끝으로 알고리듬 종료조건으로는 숙주 모집단에서 생산된 개체 수가 100만개이면 종료하였다. 실험은 C++ 프로그램 언어로 구현되었으며, 400 MHz Pentium CPU를 장착한 IBM-PC에서 수행되었다.

3.2.2 테스트문제 설정

이 연구에서 알고리듬의 성능은 진화가 종료된 후 마지막 숙주 모집단 즉, 첫번째 참가자의 최종 전략들이 테스트 문제들에 대해 평가된 결과로 비교하였다. 이 적용문제에서는 첫번째 참가자의 최적 전략을 찾는 것을 목적으로 하기 때문에 테스트 문제는 두번째 참가자의 전략들(기생충 모집단)이 된다. 테스트 문제는 기존 연구들 중에서 비교적 좋은 성능을 보이는 것으로 알려진 Rosin과 Belew(1997)의 방법에 의해 강한 기생충을 다양하게 추출하여 3개의 테스트 문제를 만들었다. 각 테스트 문제는 500개의 개체들(500개의 전략)로 구성되었다.

3.3 알고리듬 성능의 비교 분석

3.3.1 비교 알고리듬

제안한 알고리듬의 성능 분석을 위하여 기존의 알고리듬 – 정적(static) 알고리듬과 Rosin과 Belew(1997)의 알고리듬(R&B 알고리듬) – 과 그 성능을 비교한다. 정적 알고리듬은 상대 전략 또는 테스트 문제로 이루어진 모집단은 진화하지 않고, 구하는 문제의 잠재해로 이루어진 단일 모집단만을 고정된 상대 모집단에 대해 적응하면서 진화하는 알고리듬으로 정의한다. 이 연구에서 고정된 모집단은 3.2.2절에서 테스트 문제를 생성하는 방법과 동일하게 모집단 크기의 모집단을 만들었다. 이는 테스트 문제를 강하고 다양한 개체로 구성하기 위함이다. 그리고 개체의 적응도는 테스트 모집단의 모든 개체와 경쟁하여 승리한 횟수로 부여하였으며, 진화와 경쟁도 이웃단위가 아닌 전체 모집단단위로 하였다.

Rosin과 Belew (1997)은 경쟁관계에 있는 문제를 해결할 수 있는 경쟁 공진화 알고리듬을 개발하였다. 그들은 효율적인 공진화를 위하여 적응도 배분(fitness sharing), Hall of fame(과거 세대의 강한 개체의 보존전략), 배분 샘플링(shared sampling) 등의 방법을 제안하였다. 적응도 배분은 중요 적소(niches)를 보존하기 위하여, Hall of fame은 장기적 진화의 진보를 위하여, 배분 샘플링은 효율적인 적응도 테스팅을 위하여 도입되었다. 이들이 제안한 방법과 알고리듬에 관한 구체적인 내용은 Rosin과 Belew의 연구(1997)를 참조할 수 있다. 이 연구에서는 이들 세 방법을 모두 사용한 경쟁 공진화 알고리듬과 비교하였다. 배분 샘플링에서 샘플 개체의 수와 Hall of Fame의 크기는 각각 모집단의 10%로 두었다. 진화 및 경쟁은 전체 모집단단위로 하였다.

두 비교 알고리듬에서 유전 연산자 및 파라미터 설정은 3.2.1절에서와 동일하게 하였다. 단, 정적 알고리듬은 단일 모집단을 운영하므로 단순히 생산된 자손 수가 종료조건이고, 기타 알고리듬은 숙주 모집단에서 생산된 개체 수가 종료조건이 된다.

표 1. Nim게임에서 알고리듬의 성능비교

	정적 알고리듬		R&B 알고리듬		제안한 알고리듬	
	승리 횟수	완전해	승리 횟수	완전해	승리 횟수	완전해
Test1	444.90	–	499.00	8	499.70	8
Test2	433.60	–	499.90	9	499.70	8
Test3	422.50	–	498.20	8	499.70	7
평가횟수	6.98E+08		1.52E+08		2.11E+06	
계산시간 (초)	2,372		5,212		603	

3.3.2 성능 비교분석

실험은 각 문제에 대하여 10회 반복하였다. 그 결과는 <표 1>에 제시되어 있다. <표 1>에서 '승리횟수'는 각 실험에서 알고리듬을 종료한 후 숙주 모집단에 있는 개체들이 테스트문제와 경쟁하여 승리한 횟수 중 최고값을 구하여 이를 평균한 것이다. '완전해'는 10회의 반복실험에서 테스트 문제를 모두 승리한 개체를 발견한 실험의 횟수이다. 여기에서 dash(–)는 어떤 실험에서도 완전해를 발견하지 못했음을 의미한다. 실험결과, 제안한 알고리듬은 성능측면에서 정적 알고리듬보다는 우수한 성능을 보이고, R&B 알고리듬과는 거의 비슷한 결과를 보였다. 비록 제안한 알고리듬은 명시적인 Hall of Fame을 사용하지 않았지만 좋은 성능을 보였다. 정적 알고리듬은 비교적 강한 기생충들의 집합으로 숙주를 진화시켰지만, 단일 모집단을 운영하는 이 방법보다는 숙주와 기생충간의 상호경쟁을 통해 진화하는 경쟁 공진화 방법들이 훨씬 더 좋은 결과를 보였다. 이를 통해 복잡하고 동적인 문제를 해결하는 데 경쟁 공진화 기법이 더 효과적임을 알 수 있다.

정적 알고리듬은 비록 기생충이 진화되는 시간이 절약된다고 하더라도 모집단 크기의 테스트집합에 대해 모든 개체가 평가되기 때문에 많은 계산시간을 필요로 하였다. 또한 R&B 알고리듬에서는 경쟁상대를 샘플링하고, 과거의 강한 개체들을 보존하는 Hall of Fame의 운영을 통해 개체가 평가되고 진화되기 때문에 많은 평가횟수와 계산시간이 요구되었다. 그리고 숙주와 기생충 모집단의 적응도 평가와 연산과정이 직렬로 이루어져서 다른 알고리듬보다는 훨씬 많은 시간을 필요로 했다. 제안한 알고리듬에서는 개체의 적응도를 평가하는 데 토너먼트 경쟁의 특성으로 인하여 비교적 적은 횟수의 경쟁을 하고, 이를 통해 두 모집단에 있는 개체가 동시에 평가됨으로써 계산시간 측면에서 좋은 성능을 보였다. 제안한 알고리듬의 진화 과정에 관한 분석은 5절에서 다루기로 한다.

4. 정렬망 문제에의 적용

이 장에서는 3절에서 다룬 Nim게임과는 성격이 다른 정렬망 문제에 제안한 알고리듬을 적용하여 성능을 비교 분석한다. 비교 알고리듬은 3절의 두 알고리듬을 그대로 사용하였으며, 유전 연산자 및 파라미터도 특별한 언급이 없는 부분은 3.2.1절과 동일하게 사용하였다.

4.1 문제정의와 개체표현

4.1.1 문제정의

정렬망 문제는 임의로 나열된 숫자열에서 두 숫자의 비교-교환을 통해 오름차순(또는 내림차순)으로 정렬하는 망을 찾는 문제이다. <그림 3>은 구성요소 개수, $n=6$인 숫자열을 오름차순으로 정렬하는 망을 나타내고 있다. 각 수평선은 목록에 있는 한 구성요소를 나타내고, 각 수직선은 두 구성요소들의 비교를 나타낸다. 이 정렬망은 왼쪽에서 오른쪽으로 진행하면서 비교된 두 구성요소의 숫자가 오름차순으로 되어있지 않으면 순서를 교환하여 정렬하는 것을 나타낸다. 정렬망 문제의 목적은 최소의 비교-교환으로 올바르게 숫자를 나열할 수 있는 정렬망을 찾는 것이다.

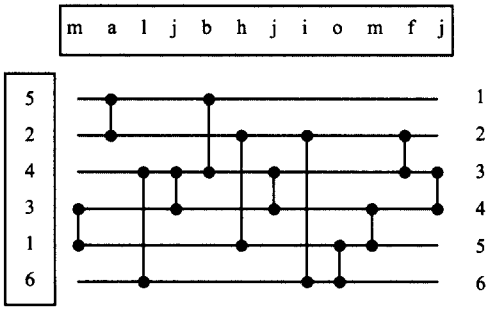

그림 3. 정렬망과 개체표현.

4.1.2 개체표현

정렬망을 숙주개체로 표현하고, 숙주개체의 정렬 능력을 테스트하기 위한 문제를 기생충개체로 표현하였다. 이때 숙주개체의 길이 즉, 비교-교환하는 횟수는 지금까지 알려진 최적 정렬망 길이를 사용하였다(Juillé, 1995). <그림 3>에서 정렬망 위쪽에 표현된 문자열은 하나의 숙주개체를 나타내고, 정렬망 좌측에 표현된 숫자열은 하나의 기생충개체를 나타낸다. 숙주개체의 첫번째 인자, m은 기생충개체의 4번째와 5번째 숫자쌍 (3, 1)을 비교하여 오름차순으로, 즉 (1, 3)으로 위치를 교환한다. 이 과정을 숙주개체의 좌측에서 우측으로 반복한다. 정렬망을 통해 정렬된 숫자열의 정렬정도는 숫자열의 1번째와 2번째, 2번째와 3번째, …, $n-1$번째와 n번째의 숫자쌍 각각이 오름차순으로 되어있는지를 측정하여 평가한다. 6개의 숫자열을 정렬한 경우, 5번 비교에서 2번만이 올바르게 되어 있으면 2/5, 즉 40% 정렬된 것으로 판단한다.

4.2 성능 비교분석

4.2.1 실험설정

성능 평가는 알고리듬을 종료했을 때 숙주 모집단내에 있는 개체가 정렬능력을 테스트하기 위해 미리 설정된 테스트 문제에 평가되어 이를 해결하는 정도로 하였다. 정렬할 숫자열이 7 이하인 경우는 모든 가능한 경우를 평가하였다. 예를 들어, 6개의 숫자열을 정렬하는 경우, 모든 가능한 순서인 6!(=720)가지를 평가하였다. 정렬할 숫자열이 8인 경우는 모든 가능한 경우에서 임의로 5,000개를 선별하여 사용하였다. <표 2>의 세 번째 열이 이를 나타낸다. 정렬할 숫자열의 수에 따라 숙주 개체의 길이, 종료 조건인 숙주 모집단에서 생산된 개체 수를 <표 2>와 같이 두었다.

표 2. 정렬망 문제의 파라미터

숫자열의 수	숙주 길이	테스트 수	생산된 개체 수
5	9	120	3,000
6	12	720	1.50E+06
7	16	5,040	1.20E+07
8	19	5,000	1.50E+07

숙주 모집단의 교차 및 돌연변이는 3.2.1절에서와 동일하게 두었고, 기생충 모집단에서는 그 특성에 따라 0.7의 교차율로 순서교차와 0.1의 돌연변이율로 역순연산자를 사용하여 자손을 생성하였다.

정적 알고리듬에서 적응도는 테스트 집합 중 정렬하기 어려운 테스트 개체에 대해서만 평가하여 이들 테스트 개체를 모두 정렬한 개수로 하였다. 여기서 정렬하기 어려운 개체는 정렬정도를 측정하여 정렬할 숫자열이 5인 경우는 2회 이하만 올바른 순서로 되어있는 개체 중 50개를 임의로 선택, 정렬할 숫자 열이 6, 7인 경우는 2회 이하만 올바른 순서로 되어있는 개체 중에서 100개를 임의로 선택, 정렬할 숫자열이 8인 경우는 1회 이하만 올바른 순서로 되어 있는 개체 중에서 100개를 임의로 선택하였다. 그리고 제안한 알고리듬에서 참가비는 자신의 적응도의 5%로 하고, 모든 알고리듬에서 모집단의 크기를 10×10으로 한 것을 제외하고는 Nim게임 문제와 동일한 조건으로 하였다.

4.2.2 성능 비교

5, 6, 7개의 숫자열 문제에서는 10회 반복실험을 했으나, 8개의 숫자열 문제에서는 해공간이 넓어 계산시간이 많이 요구되어 5회 반복실험만 하였다. 실험결과는 <표 3>과 같다. 제안한 알고리듬은 성능이나 계산시간 측면에서 다른 알고리듬보다 좋은 결과를 보였다.

표 3. 정렬망 문제에서 알고리듬의 성능비교

	정적 알고리듬		R&B's 알고리듬		제안한 알고리듬	
	승리횟수	완전해	승리횟수	완전해	승리횟수	완전해
5개 숫자열정렬	120.00	10	120.00	10	120.00	10
평가횟수	1.73E+05		80,600		5,961	
계산시간(초)	0.36		1.35		0.50	
6개 숫자열정렬	677.60	3	711.60	8	720.00	10
평가횟수	1.67E+08		3.98E+07		2.42E+06	
계산시간(초)	316		652		218	
7개 숫자열정렬	4,730.40	1	5,025.60	9	5,040.00	10
평가횟수	1.34E+09		3.18E+08		1.99E+07	
계산시간(초)	3,119		6,174		1,704	
8개 숫자열정렬	3,456.20	–	4,594.40	1	4,575.20	1
평가횟수	1.67E+09		3.98E+08		2.90E+07	
계산시간(초)	4,411		8,341		2,329	

5. 토너먼트 경쟁의 효과와 공진화 과정 분석

제안한 알고리듬의 성능을 보이기 위하여 3장과 4장에서 각기 다른 특성을 가진 문제에 적용하여 기존 알고리듬과 비교 분석하였다. 분석결과, 해의 질에 있어서는 제안한 알고리듬이 비교기법에 뒤지지 않았으며, 계산시간은 상대적으로 아주 적게 소요되었다. 이러한 결과를 보이는 이유를 좀 더 구체적으로 보이기 위하여, 이 장에서는 제안한 토너먼트 경쟁과 이웃진화가 갖는 장점, 그리고 제안한 알고리듬에서 지속적인 진화적 무기경쟁이 일어나는 현상을 보이고자 한다. 실험은 Nim 게임 문제를 대상으로 하였다.

5.1 토너먼트 경쟁

경쟁전략간의 성능을 비교 분석하기 위하여 먼저 모집단 단위의 진화와 경쟁방법을 사용하였다. 이웃진화의 경우는 다음 절에서 다루기로 한다. 비교 대상이 되는 경쟁전략으로는 상대 모집단의 모든 개체와 경쟁하는 전체경쟁, 상대 모집단에서 샘플링된 개체와 경쟁하는 샘플링경쟁, 그리고 이 연구에서 사용한 토너먼트 경쟁 등 세 가지이다. 샘플링 경쟁에서는 모집단의 30%의 샘플크기를 갖고 임의로 샘플링을 한 경우(임의 샘플경쟁)와 과거 적응도를 기준으로 높은 적응도를 가진 개체를 샘플링 한 경우(정보 샘플경쟁)에 대하여 성능비교를 하였다. 그리고 모든 경쟁방법에서 개체의 적응도는 상대 개체와의 경쟁에서 얻은 승리횟수로 두었다.

실험결과는 <표 4>와 같다. 실험결과, 승리횟수 측면에서는 토너먼트 경쟁이 가장 좋은 결과를 보이지만 완전해 측면에서는 정보 샘플경쟁이 더 나은 결과를 보인다. 그러나 계산시간 측면에서는 토너먼트 경쟁이 가장 우수하였다.

두 샘플경쟁 사이의 결과를 보면, 완전해 측면에서는 정보 샘플경쟁이 임의 샘플경쟁보다 좋은 결과를 보였지만, 승리횟수 측면에서는 반대의 결과를 보인다. 정보 샘플경쟁에서 완전해는 많이 찾으나 해의 편차가 심하여 승리횟수의 결과는 좋지 않게 나타났다. 이는 개체들이 아직 높은 경쟁력을 갖지 못한 상태에서 승리하기 어려운 상대 개체가 많이 샘플링되어 이로부터 테스트되면 대부분 낮은 적응도를 부여받게 된다. 한 실험에서 얻은 숙주와 기생충 모집단의 평균 적응도 변화를 <그림 4>에 표현하였다. 일정세대 후 기생충 모집단이 일방적으로 우세하여, 즉, 두 종간의 불균형이 발생하여 숙주개체를 모두 이기게 되고, 이로 인하여 숙주의 진화를 유도할 수 있는 무기경쟁이 더 이상 이뤄지지 못하고 조기수렴하는 결과를 보임을 알 수 있다.

전체경쟁은 많은 계산시간을 요구하지만 성능의 결과는 만족스럽지 못하였다. 임의의 샘플경쟁과 비교할 때도 성능에 있어서 크게 차이가 있는 것으로 보이지는 않는다. 이는 전통적인 진화 알고리듬에서는 상대가 고정되어 있으나, 경쟁 공진화 알고리듬에서는 상대 개체도 진화하며, 매 세대마다 개체

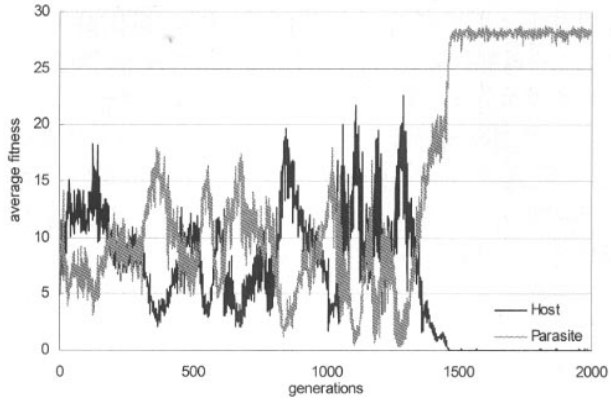

그림 4. 정보 샘플경쟁에서 숙주와 기생충의 평균 적응도 변화.

표 4. 모집단단위의 경쟁/진화에서 경쟁방법에 따른 성능비교

	전체경쟁		정보 샘플경쟁		임의 샘플경쟁		토너먼트 경쟁	
	승리횟수	완전해	승리횟수	완전해	승리횟수	완전해	승리횟수	완전해
Test1	455.10	2	306.10	5	465.60	–	468.86	1
Test2	451.30	1	300.30	5	460.80	1	464.00	1
Test3	452.10	1	297.20	4	455.40	–	458.14	1
평가횟수	8.25E+08		5.01+08		5.01E+08		2.03E+06	
계산시간(초)	10,940		5,844		5,410		912	

는 이전 세대에서 갖는 진화정보인 승리횟수에 무관하게 적응도를 평가하기 때문인 것으로 보인다. 이는 또한 통계학에서 샘플링 이론과 그 개념이 일치한 것으로 판단된다. 그러나 예비실험을 통해 상대가 고정되어 있는 상태에서 전체경쟁과 임의 샘플경쟁과의 성능을 비교한 결과, 후자가 좋지 않은 결과를 보였다. 이로써 경쟁 공진화 알고리듬은 상대가 고정되어 있는 경우와는 다른 특성을 갖는다는 것을 알 수 있다.

토너먼트 경쟁이 비교적 좋은 결과를 보인 것은 선택 압력(selective pressure)과 모집단 다양성이 적절히 조화를 이루기 때문인 것으로 판단된다. 토너먼트 경쟁에 의하면 개체의 상대적 순위가 결정되면서 경쟁 상대에 따라 적응도 값에 의한 상대적 순위에 차이가 날 수 있다는 임의성도 갖는다. 또한 첫번째 토너먼트에서 패배한 개체들은 같은 적응도를 부여받기 때문에 이들 개체들은 차별성을 두지 않아 다양성 유지에 도움이 되는 것으로 여겨진다.

경쟁 공진화 알고리듬에서는 상대 모집단의 동적인 특성들로 인하여 현재 상대 개체에 대해서만 평가한 적응도로 순위가 완전히 부여되어 진화하게 되면, 초기 세대에서 잠재적으로 좋은 해가 사라져 버릴 수 있다는 단점을 갖는다. 이런 경우에는 상위의 좋은 개체들이 진화를 유도하게 하고 나머지 개체들의 다양성을 높임으로써 진화과정에서 해공간을 다양하게 탐색할 수 있게 하는 토너먼트 경쟁이 더 효과적일 수 있다.

5.2 이웃진화

토너먼트 경쟁에서 모집단 단위의 진화와 이웃진화의 성능분석을 하여 보자. 이때 이웃진화는 3×3의 이웃구조에서 이루어진다. <표 5>에서 이웃단위로 경쟁과 진화를 하는 '토너먼트+승리횟수'와 모집단 단위로 진화하면서 승리횟수로 적응도를 부여한 <표 4>의 '토너먼트경쟁'과 비교하면, 이웃진화가 해의 질을 상당히 향상시킴을 알 수 있다. 이웃진화는 중요한 적소들을 유지함으로써 더 좋은 결과를 보이는 것으로 생각된다. 이와 비슷한 이유로 이웃의 모든 개체와 '전체경쟁'하면서 이웃진화를 하는 경우도 모집단 단위로 진화하는 경우보다는 좋은 결과를 보였다. 그리고 제안한 알고리듬에서 적

표 5. 이웃단위의 경쟁/진화에서 경쟁 및 적응도 평가방법에 따른 성능비교

	전체경쟁+승리횟수		토너먼트+승리횟수		토너먼트+참가비	
	승리횟수	완전해	승리횟수	완전해	승리횟수	완전해
Test1	493.80	3	494.80	4	499.70	8
Test2	493.30	3	493.40	2	499.70	8
Test3	492.00	2	494.90	3	499.70	7
평가횟수	1.27E+07		2.14E+06		2.11E+06	
계산시간(초)	685		613		603	

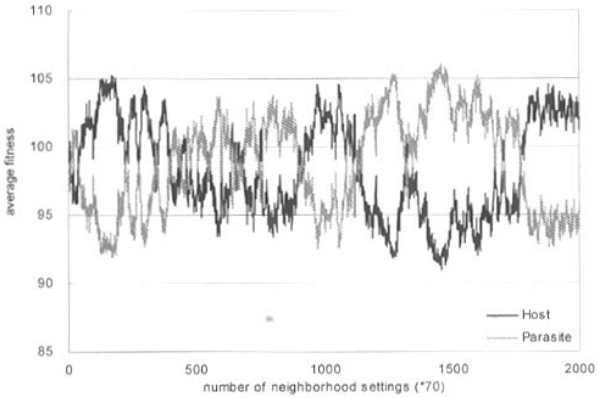

그림 5. 제안한 알고리듬에서 숙주와 기생충의 평균 적응도 변화.

응도 부여방법으로 사용한 참가비 교환방법은 단순히 경쟁에서의 승리횟수에 의해 적응도를 부여하는 방법보다 더 좋은 결과를 보였다. 이는 앞에서도 언급하였듯이 참가비 교환에 의한 적응도 평가가 경쟁상대의 난이도를 반영하여 적응도 배분효과를 갖기 때문인 것으로 생각된다.

5.3 숙주와 기생충간의 상호진화 과정 분석

경쟁 공진화 알고리듬의 성능 향상을 위한 가장 중요한 조건은 종간의 상호 경쟁에 의해 공진화가 계속되어야 한다는 것이다. 즉, 진화적 무기경쟁이 지속될 수 있어야 한다. 여기에서는 제안한 알고리듬의 공진화 과정을 보이고자 한다. <그림 5>는 이웃이 70회 정의될 때마다 숙주와 기생충 모집단의 평균 적응도를 구하여 그 변화를 보인 것이다. 이 그림으로부터 숙주와 기생충 모집단은 진화 시간이 흐름에 따라 교대로 비교우위를 차지하면서 공진화해 감을 알 수 있다.

6. 결 론

이 연구에서는 복잡성이 높은 경쟁관계 문제를 효율적으로 해결할 수 있는 경쟁 공진화 알고리듬을 제안하였다. 경쟁 공진화 알고리듬에서는 상호 관련성이 있는 종들이 지속적인 무기경쟁을 통하여 진화가 균형적으로 이루어질 수 있어야 하며, 다양하고 동적인 환경에서 효과적으로 진화할 수 있어야 한다. 이를 위하여 이 연구에서는 토너먼트 경쟁, 참가비 교환에 의한 적응도 평가, 이웃단위의 경쟁 및 진화전략을 사용한 경쟁 공진화 알고리듬을 제안하였다. 토너먼트 경쟁은 선택 압력과 모집단 다양성이 조화되도록 하여 좋은 성능을 보였다. 참가비 교환에 의한 적응도 평가는 경쟁 상대에 따라 적응도가 다르게 부여되어 적응도 배분 효과를 보이며, 이웃단위의 경쟁 및 진화는 모집단의 다양성과 함께 중요한 적소를 유지함으로

써 알고리듬의 성능 향상에 기여하였다.

제안한 경쟁 공진화 알고리듬의 성능 분석을 위하여 특성이 다른 두 예제 문제 – Nim게임 문제와 정렬망 문제 – 에 이를 적용하고, 기존 알고리듬과 성능을 비교 분석하였다. 실험결과, 제안한 알고리듬은 기존 알고리듬보다 우수한 성능을 보였으며, 진화적인 무기경쟁이 지속됨을 확인하였다. 제안한 알고리듬은 진화 알고리듬이 갖는 적용의 유연성에 의해 복잡하고 동적인 여러 형태의 경쟁관계를 갖는 문제에 쉽게 적용 가능하다.

참고문헌

Angeline, P. J. and Pollack, J. B. (1993), Competitive environments evolve better solutions for complex tasks, *Proceedings of the 5th International Conference on Genetic Algorithms*, 264-270.

Bull, L., Fogarty, T. C. and Pipe, A. G. (1995), Artificial endosymbiosis, *Advances in Artificial Life: Third European Conference on Artificial Life*, 273-289.

Hillis, W. D. (1991), Co-evolving parasites improve simulated evolution as an optimization procedure, *Artificial Life* II, 313-324.

Holmes, J., Routen, T. W. and Czarnecki, C. A. (1995), Heterogeneous co-evolving parasites, *Proceedings of Artificial Neural Nets and Genetic Algorithms '95*, 156-159.

Husband, P. and Mill, F. (1991), Simulated co-evolution as the mechanism for emergent planning and scheduling, *Proceedings of the 4th International Conference on Genetic Algorithms*, 264-270.

Juillé, H. (1995), Incremental co-evolution of organisms: A new approach for optimization and discovery of strategies, *Advances in Artificial Life: Third European Conference on Artificial Life.*

Koza, J. R. (1992), *Genetic Programming: On the Programming of Computers by Means of Natural selection*, The MIT Press, Cambridge.

Moriarty, D. E. and Miikkulainen, R. (1997), Forming neural networks through efficient and adaptive coevolution, *Evolutionary Computation*, 5(4), 373-399.

Nolfi, S. (1998), Coevolving predator and prey robots: Do 'Arms Races' arise in artificial evolution?, *Artificial Life* 4, 311-335.

Olsson, B. (1996), Optimization using a host-parasite model with variable-size distributed populations, *International Conference on Evolutionary Computation '96*, 295-299.

Pagie, L., and Hogeweg, P. (1998), Evolutionary consequences of coevolving targets, *Evolutionary Computation*, 5(4), 401-418.

Paredis, J. (1995), Coevolutionary computation, *Artificial Life* 2, 355-375.

Potter, M. A. (1997), The design and analysis of a computational model of cooperative coevolution, *Ph.D. dissertation*, George Mason University.

Rosin, C. D. and Belew, R. K. (1997), New methods for competitive coevolution, *Evolutionary Computation*, 5(1), 1-29.

Smith, R. E., Forrest, S. and Perelson, A. S. (1993), Searching for diverse, cooperative populations with genetic algorithms, *Evolutionary Computation*, 1(2), 127-149.