유전자 알고리즘을 이용한 대전형 액션게임의 지능캐릭터

이 면 섭[†] · 조 병 헌[††] · 성 영 락[†††] · 정 성 훈[††††] · 오 하 령[†††††]

요 약

본 논문에서는 유전자 알고리즘을 이용하여 대전형 액션 게임의 캐릭터를 지능화 하는 방법을 제안한다. 초기에 게임 규칙을 전혀 모르는 캐릭터가 학습이 진행되면 세대를 거듭하면서 스스로 게임규칙을 학습하고 기술을 습득하며 진화할 수 있는 지능을 갖추게 된다. 제안한 캐릭터가 환경변화에 잘 적응하는지를 평가하기 위하여 게임 도중에 게임규칙을 바꾸어 규칙 변경 전후의 결과를 비교 검토하였다. 실험결과 게임 규칙이 변경된 후에도 빠르게 새로운 규칙에 적응하였다. 본 논문에서 제안한 방법은 대전형 액션게임뿐만 아니라 PC게임이나 인터넷 온라인 게임 등의 캐릭터 구현에도 쉽게 적용가능한 장점이 있다.

키워드 : 유전자 알고리즘, 게임, 인공지능, 지능 캐릭터

An Intelligent Characters for Fighting Action Games Using Genetic Algorithms

Myun-sub Lee[†] · Byeong-heon Cho[††] · Sung-hoon Jung[†††] · Yeong-rak Seong[††††] · Ha-ryoung Oh[†††††]

ABSTRACT

This paper proposes a method to provide intelligence for characters in fighting action games by using genetic algorithm. The proposed characters without any knowledge on the rules of the game learn the rules and techniques for generations, and have the capability of evolving. To evaluate adaptability for varying circumstances, we changed the rules and compared the results. The experimental results show that the intelligent characters can adapt to the new rules. An advantage of the proposed method is that it can be easily applied to characters for other category of games such as PC games and internet online games.

Key Words : Genetic Algorithm, Game, Artificial Intelligence, Intelligent Character

1. 서 론

1999년 GDC(Game Developers Conference)이후 가장 뚜렷한 경향은 인공지능 기술을 사용하여 게임 캐릭터들을 좀 더 사실적이고 생명체와 비슷한 행동을 하도록 노력하고 있다. 이를 위한 방법으로는 FSM(Finite State Machine), FuSM(Fuzzy State Machine), Decision Tree, 규칙 기반 시스템 등이 있다[1, 2]. 그러나 이러한 방법들은 게임을 실감나게 하기 위해 혹은 단순한 반복을 회피하기 위해 사용되었고, 인공지능이 게임 플레이어의 지능적인 행동에 직접 관여하지는 않았다. 또한 모든 상황을 미리 설정해 놓고 그 규칙대로 코딩이 이루어지므로 한번 코딩이 이루어지면 게임 중간에 규칙을 바꿀 수가 없다는 단점이 있다[3, 4].

David Carmel[5]과 John E. Laird[6]등이 인공지능 게임에 관한 논문을 발표하였으나 논문의 주된 내용은 게임에서 캐릭터가 효율적인 공격을 위한 위치 선정 알고리즘에 대한 것이며, I. Faybish[7]는 플레이어가 유리한 위치를 분석하고 그 위치를 선정하기 위한 가능한 방법을 탐색하는데 유전자 알고리즘을 적용하였다. 이들 방법은 인공지능을 이용한 게임에서 캐릭터 자신이 학습하고 진화하는 지능 캐릭터와는 거리가 있다.

본 논문에서는 유전자 알고리즘을 이용하여 대전형 액션 게임을 위한 지능 캐릭터를 구현하는 방법을 제안한다. 본 논문에서 제안하는 지능캐릭터는 게임의 규칙이나 상대 캐릭터에 대하여 전혀 모르는 상태에서의 지능 캐릭터가 상대 캐릭터와 대결하고 그 결과를 이용하여 게임의 규칙을 학습하고 이렇게 획득한 정보를 다음 세대의 캐릭터에게 전달해 줌으로써 세대수가 증가할 수록 게임규칙과 환경에 스스로 적응해 나가는 캐릭터이다. 유전자 알고리즘은 해결하고자 하는 문제를 정해진 형태의 염색체로 표현한 다음 이들을 진화시키는 방법이다. 각 캐릭터가 얼마나 빠르게 학습이

† 정 회 원 : 인천전문대학 컴퓨터정보과 부교수
†† 준 회 원 : 국민대학교 전자공학과 공학박사
††† 종신회원 : 국민대학교 전자정보통신공학부 부교수
†††† 정 회 원 : 한성대학교 정보통신공학과 부교수
††††† 정 회 원 : 국민대학교 전자정보통신공학부 교수
논문접수 : 2004년 6월 11일, 심사완료 : 2005년 4월 25일

진행되는지와 게임의 규칙이 어느 순간 변경되었을 경우에 얼마나 빠르게 적응하는지를 살펴보기 위하여 실험을 하였다. 실험에서는 각 캐릭터에 대해서 랜덤하게 거리별로 100번씩 행동을 주고받아 얻어진 점수를 적합도로 사용하였다. 실험결과 비교적 짧은 세대만에 게임에 적합한 값으로 학습이 되었다. 또한 게임의 규칙 및 점수가 많이 변화된 경우에도 비교적 빠른 시간에 새로운 규칙에 적응하는 캐릭터로 진화됨을 볼 수 있었다. 이러한 결과는 본 논문에서 제안한 유전자 알고리즘을 이용한 지능캐릭터 구현이 효과적임을 보여주는 것이다. 제안한 알고리즘은 대전형 액션게임 이외에 PC 게임이나 다수의 종족이 존재하는 온라인 게임에도 적용할 수 있는 일반적인 방법으로서 그 응용분야가 넓다고 하겠다.

본 논문의 구성은 다음과 같다. 2절에서는 게임에서의 인공지능 기법에 대하여 알아보며, 3절에서는 지능형 대전 액션게임의 캐릭터의 구현으로, 본 논문에서 적용한 유전자 알고리즘의 각 파라미터에 대하여 설명한다. 4절에서는 대전 액션게임에 대해서 설명하며, 5절에서는 지능 캐릭터의 진화 능력과 게임규칙 변경 전·후의 결과를 알아보고, 6절에서 결론을 맺는다.

2. 게임에서의 인공지능 기법

게임에서의 인공지능의 역할은 등장 캐릭터의 지능적인 행동을 구현하기도 하며 상대 캐릭터의 보조자 역할을 수행하기도 한다. 이를 구현하기 위해서는 FSM(Finite State Machine), FuSM(Fuzzy State Machine), Decision Tree, 규칙 기반 시스템 등이 있으며 최근에는 유전자 알고리즘도 사용되고 있다[8]. FSM은 현재 가장 널리 사용되고 있는 인공지능 처리 방식으로, 인공지능이 특별히 요구되지 않는 게임에 주로 사용되며, 상태수가 늘어나면 상태도를 정리하기가 어려우며 캐릭터의 행동을 예측하기 쉽다는 단점이 있다[9, 10]. FuSM은 FSM에 퍼지 이론을 적용한 것으로, 입력과 출력에 퍼지함수로 일정한 랜덤 기능을 적용하여 동일한 외부 상황에도 다른 출력을 얻을 수 있도록 한 방법이다. 논리적인 기반에서 애매한 정보를 처리, 추론하는데 적합하고 이를 게임에 적용시키면 좀더 구체적인 캐릭터 제어를 할 수가 있으며 상대편의 행동을 예측하기가 어려워, 보다 현실적인 게임을 만들 수가 있다[9, 10, 11]. Decision Tree는 FSM의 단점을 보완하는 방법으로 상태수가 많아지더라도 현재의 상태에 따라 다음 행동을 트리 형태로 나누어 해당하는 행동을 하도록 한다. 그러나 한번 정해진 Decision tree가 계속 사용되기 때문에 인공지능이 담당하는 캐릭터의 행동 예측이 가능하다는 단점이 있다. 규칙기반 시스템은 전통적인 인공지능 학문에 기반을 둔 기법으로 많은 규칙을 나열하고 현 상황에 맞는 규칙을 선택하여 이 규칙에 따른 결과를 얻는 것이다[9, 10]. 이 방법은 복잡한 판단이 가능하므로 FSM이나 Decision Tree에 비하여 높은 지능을 구현할 수 있다. 그러나 규칙수가 늘어날 경우 현 상태를 표시하는 상태를 찾아내는 매칭에 많은 시간이 소요되며 단순한 지능으로 액션게임에 적용하기는 어렵다.

게임에 인공지능을 적용하는 연구는 체스나 바둑 등에서 경로 찾기(A* 알고리즘 등)분야에서 비교적 많은 연구가 진행되었지만 게임 캐릭터에 지능을 부여하여 스스로 행동하고 판단하는 지능적인 행동을 할 수 있는 캐릭터와는 거리가 있다. 이와 같이 기존의 인공지능 기법에서는 미리 게임 규칙이 정해져 있기 때문에 게임 캐릭터가 스스로 판단하고 행동할 수가 없다. 이러한 문제점을 해결하고자 본 논문에서는 유전자 알고리즘을 적용하여 지능 캐릭터가 스스로 게임 규칙을 학습하고 환경에 적용할 수 있는 게임 캐릭터를 구현하였다. 대전 액션 게임의 특성상 게임 캐릭터의 습성이나 행동 양식이 너무 복잡하고 다양하기 때문에 탐색 공간이 무한대일 경우가 있으며, 게임의 난이도를 수식적으로 정의하기에 무리가 있다. 또 게임의 특성상 전적으로 최적해를 요구하지도 않는다. 이와 같이 게임의 복잡성과 모호성, 다양성을 충족시키고자 본 연구에서는 대전형 액션 게임의 지능 캐릭터 구현에 유전자 알고리즘을 적용하였다.

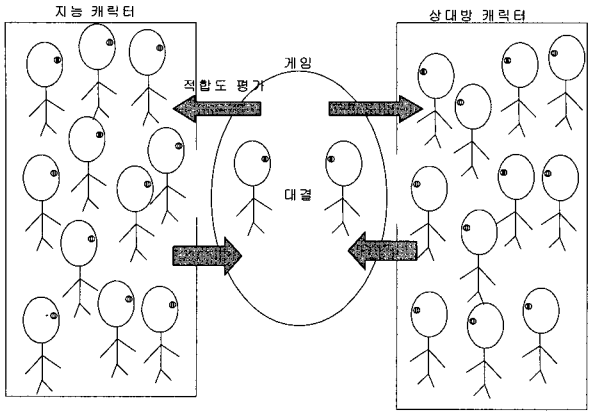
3. 지능형 대전 액션게임의 캐릭터 구현

본 절에서는 대전형 액션게임에서 유전자 알고리즘을 이용한 지능 캐릭터의 구현에 대하여 설명한다. 지능 캐릭터는 게임규칙을 모르고 상대 캐릭터와 게임을 진행하지만 선택, 교배, 돌연변이를 거듭하면서 진화하여 최적의 상태로 수렴하게 된다. 게임규칙에서 캐릭터의 행동은 모두 11가지이며 이동이나 방어 행동이 6종류이고 공격 행동은 5가지로 정하였다.

3.1 지능 캐릭터의 구현

1975년 Holland에 의해 소개된 유전자 알고리즘은 자연생태계의 진화이론을 이용한 탐색과정이다. 유전자 알고리즘은 염색체(chromosome), 적합도 함수(fitness function), 선택 연산자, 교배 연산자, 돌연변이 연산자, 그리고 종료 조건 등으로 정의되며 알고리즘의 흐름도는 (그림 1)과 같다. 그림에서 사각형 안의 각 캐릭터는 염색체를 나타내며 지능 캐릭터와 상대방 캐릭터를 나타내고 있다. 여기서 상대방 캐릭터는 무작위로 행동하는 것이며 지능 캐릭터는 게임 규칙에 따라 행동한다. 가장 먼저 적합도를 계산하기 위해서 모든 개체는 한번씩 게임을 하여 그 결과로서 적합도를 평가하게 된다. 이 적합도에 따라 교배에 참여할 부모 염색체가 선정되며 두개의 부모 염색체로부터 자식 염색체 두개가 생성된다.

염색체는 탐색공간에서 찾고자 하는 해의 후보를 나타낸다. 본 논문에서 염색체는 캐릭터간의 거리와 상대 캐릭터의 행동에 따라 지능 캐릭터가 어떤 행동을 할 것인지를 규정한다. 이에 따라 염색체는 2차원 구조를 가진다. 가령, 각 캐릭터가 할 수 있는 행동의 종류가 m 개이고, 상대방과의 거리를 n 단계로 구분한다면, 염색체는 (그림 2)와 같이



(그림 1) 알고리즘의 개요

상대행동 \ 거리		0	1	2	3	4	5	6
0	정지상태	4	9	1	10	0	9	2
1	전진	7	2	6	1	4	3	7
2	후진	9	7	1	0	8	10	1
3	막기	3	5	9	3	2	7	10
4	앉기	2	7	6	1	10	0	6
5	점프	2	6	9	3	8	3	3
6	아래주먹	3	9	7	0	5	9	9
7	윗 주먹	10	7	9	2	7	3	2
8	아래 발공격	3	1	9	7	9	2	8
9	윗 발 공격	7	4	1	6	10	8	2
10	필살기	0	9	3	2	5	4	6

(그림 2) 염색체의 구조

m*n 배열의 구조가 된다. 그리고 배열내의 각각의 값은 특정 상황에서의 지능 캐릭터의 행동을 정의하는 것으로 0에서 m-1까지의 값으로 표현된다. 알고리즘 시작 시에 각 염색체의 값은 무작위 값으로 초기화된다.

적합도는 염색체의 적합성을 나타내는 것으로서 개체의 생존 능력을 평가하기 위한 기준이다. 본 논문에서는 어떤 캐릭터가 각각의 거리에서 100 시간 단위 동안에 얻은 점수를 적합도로 정의하였다. 선택(selection)연산은 후에 설명할 교배 연산 시에 두 개의 부모 염색체를 선택할 때 사용된다. 본 연구에서는 널리 알려진 룰렛휠 선택(roulette wheel selection)[15] 방법을 사용하였다. 룰렛휠 선택은 선택확률(적합도)에 따라 이전의 집단에서 염색체를 선택하는 것으로서, 유전자를 변경하지는 않으므로, 집단 내에 있는 유전자 형태의 변화에는 영향을 미치지 않는다. 또한 확률적 속성 때문에 선택과정에서 최적자를 반드시 선택하지 못하는 단점을 가진다. 본 연구에서는 한 세대 내에서 가장 좋은 적합도를 가지는 개체는 반드시 다음 세대에 자신의 우성인자를 전달할 수 있도록 하였다.

교배(crossover) 연산자는 선택된 염색체들이 교배를 통해 서로의 유전 정보를 교환함으로서 새로운 염색체를 생성하는 과정이다. 본 연구에서는 유전자의 형태가 이차원이므로 다차원 교배인 블록 균등교배(block uniform crossover)를 적용하였다[12, 13, 14]. 돌연변이(mutation)는 자연계의 돌연변이를 모방한 것으로 염색체내의 임의의 부분을 돌연변이 확률을 토대로 변경시키는 것이다. 이렇게 함으로써 초기 세대에서 모든 염색체의 특정 비트가 하나의 숫자로 고정되는 것을 방지해 주고 또한 부모해에 존재하지 않는 속성을 도입하여 탐색 영역을 확대해 주기도 한다. 본 연구에서는 염색체 표현이 정수이므로 돌연변이 확률에 해당되면 난수를 발생시켜 선택된 값으로 대치하였다.

종료조건은 한 세대에서 다음 세대로 넘어가는 과정을 끝내는 시점을 결정하는 것이다. 일반적으로는 최대 세대수를 지정하여 그 횟수만큼 수행하는 방법과 수렴된 유전자의 수가 일정 비율 이상이면 종료하는 방법이 있다. 본 연구에서 최대 세대수는 1000회로 하면서도, 그 이전에라도 100%의 해를 찾는 경우에는 정지하도록 하였다.

4. 대전 액션 게임

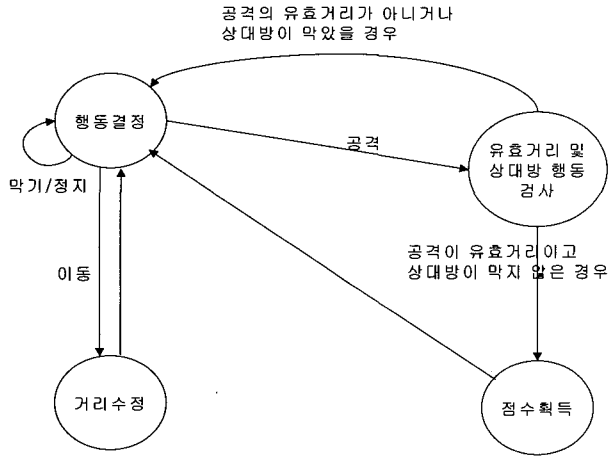
제안한 알고리즘의 성능 및 유용성을 평가하기 위하여 간단한 대전 액션 게임을 구현하였다. 앞에서 설명한 바와 같이 제안된 알고리즘의 중요한 특징 중의 하나는 게임의 규칙이 부분적으로 변경될 경우 자동적으로 새로운 규칙에 적응해 나가는 능력이다. 이를 입증하기 위해서 구현된 게임에서는 두 개의 게임 규칙을 정하고 필요에 따라 다른 게임 규칙으로 바꿀 수 있도록 하였다. 본 절에서는 구현된 대전 액션 게임에 대해서 설명한다.

구현된 게임에서 캐릭터가 취할 수 있는 행동의 종류로는, 정지 상태(ID), 전진(GO), 후진 (BK), 막기(GD), 앉기(DN), 점프(JP), 아래 주먹공격(DP), 윗 주먹공격(UP), 아래 발 공격 (DK), 윗 발공격(UK), 그리고 필살기 공격(SP)등 11가지이다. 이중 공격 행동은 아래 주먹공격, 윗 주먹공격, 아래 발 공격, 윗 발 공격과 필살기 등이다.

두 캐릭터는 모든 거리에서 어떠한 공격도 가능하지만 점수로 인정받기 위해서는 공격에 따라 제한된 유효거리 내에서 공격이 이루어질 때에만 점수가 인정된다. 앞서 설명한 바대로 구현된 게임에서는 두 개의 게임 규칙을 가진다. <표 1>은 두 게임 규칙에서의 각 공격의 유효거리 및 점수를 나타낸 것이다. 예를 들어 DP는, 게임 규칙 1에서는 상대방과의 거리가 0-2일 때 사용할 경우 1점을 획득하지만, 게임 규칙 2에서는 거리가 0-1일 때에만 점수를 획득할 수 있고 공격 점수도 5점이 된다. 게임과정에서 점수를 획득하는 과정은 (그림 3)과 같다. 그림에서와 같이 유효거리 이내에서 공격이 이루어 질 경우만 점수를 획득할 수 있다. 여기서 이동(전진, 후진)은 지능 캐릭터 자신이 상대 캐릭터와의 거리를 보고 결정하게 된다. 좀더 구체적으로 설명하면 다음과 같다.

〈표 1〉 게임규칙 변경 전후 거리별 공격 형태 및 점수

a. 게임 규칙 1							
거리 / 행동	0	1	2	3	4	5	6
DP(1점)	D P						
UP(2점)	U P						
DK(3점)			D K				
UK(4점)			U K				
SP(5점)				S P			
b. 게임 규칙 2							
거리 / 행동	0	1	2	3	4	5	6
DP(5점)	D P						
UP(4점)	U P						
DK(3점)		D K					
UK(2점)		U K					
SP(1점)				S P			



(그림 3) 지능 캐릭터의 행동

상대방의 공격에 대처하는 방법으로는 GO, BK 등으로 상대방과의 거리를 조절하거나, GD, JP, DN 등의 방어 자세를 취하거나, 같이 공격하는 방법이 있다. 거리 조절 방법은 상대방이 현재 취하고 있는 공격의 유효거리 밖으로 이동하는 것이다. 예를 들어 게임 규칙 1에서 상대방이 거리 2에서 DP 공격을 해 올 경우에 BK 하면 상대방의 공격이 무효가 된다. 그러나 만약 거리가 1인 경우에는 GO나 BK을 해도 타격을 받는다. 이런 경우에는 방어 자세를 취하는 방법이 있다. 유효한 방어 자세에 대한 규정은 게임 규칙에 따라 다르게 정의하였다. 게임 규칙 1의 경우 각 공격에 대해 취할 수 있는 방어 자세는 다음과 같다.

- DP, UP, DK, UK 공격에 대해서 GD하면 공격 점수가 인정되지 않는다.
- DP, DK 공격에 대해서 JP하면 공격 점수가 인정되지 않는다.
- UP, UK 공격에 대해서 DN하면 공격 점수가 인정되지 않는다.
- SP 공격에 대해서 GD하면 공격 점수의 50%만 인정된다.

그러나 게임 규칙 2에서는 모든 방어 자세가 인정되지 않는다. 즉 어떠한 방어 자세를 취해도 모든 공격 점수가 인정된다. 그러므로 게임 규칙 2에서는 거리 조절을 하거나 맞공격을 통해서 방어할 수 있다.

<표 2>는 앞서 제시한 게임 규칙들을 이용하여 게임규칙 1에서의 최적 행동을 구한 것이다. 최적의 행동이란 주어진 상황에서 자신의 점수의 증가치와 상대방 점수의 증가치의 차이를 최대로 만드는 자신의 행동을 의미한다. 예를 들어 서로간의 거리가 0이고 상대방이 ID 행동을 취하고 있을 때를 가정해 보자. ID는 공격 행동이 아니므로, 상대방의 점수의 증가치는 0이다. 한편 <표 1>에 의하면 거리 0에서 취할 수 있는 유효한 공격으로는 DP와 UP가 있는데 DP가 UP에 비해서 더 높은 공격 점수를 가진다. 그러므로 이 상황에서는 UP 공격이 최적의 행동이 된다.

<표 2>에서 * 로 표시된 것은 어떠한 행동을 취해도 무방하다는 의미이다. 예를 들어 상대방이 거리 0에서 GD 자세를 취하고 있다면, 지능 캐릭터가 어떠한 행동을 취하더라도 두 캐릭터의 점수에는 전혀 변화가 없다. 또한 $\overline{1}$로 표시된 것은 1번 행동(GO) 외의 어떠한 행동도 무방하다는 의미이다. 만약 서로간의 거리가 6인 상태에서 상대방이 SP 공격을 할 때 지능 캐릭터가 GO 한다면, 상대방의 공격이 유효하게 된다. 그러나 그 외에는 어떠한 행동을 하더라도 서로의 행동이 상대에게 영향을 주지 않으므로 두 캐릭터의 점수에 변화가 생기지 않으므로 최적의 행동이 된다.

〈표 2〉 최적일 경우 엄색체 구조와 행동의 종류

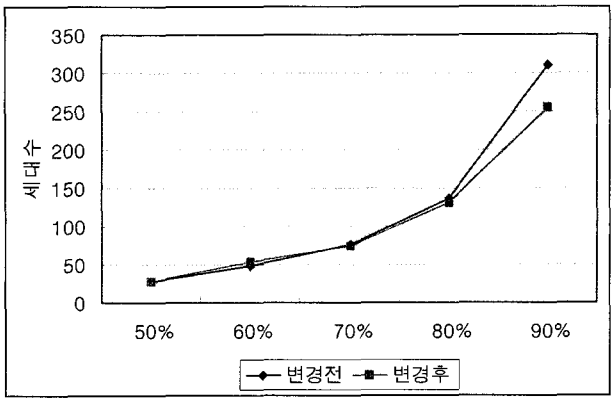
a. 게임규칙 1								
상대행동 / 거리		0	1	2	3	4	5	6
0	ID	7	7	9	10	10	10	*
1	GO	7	7	9	10	10	10	*
2	BK	7	7	9	10	10	10	*
3	GD	*	*	*	10	10	10	*
4	DN	6	6	8	10	10	10	*
5	JP	7	7	9	10	10	10	*
6	DP	7	7	9	10	10	10	*
7	UP	7	7	9	10	10	10	*
8	DK	7	7	9	10	10	10	*
9	UK	7	7	9	10	10	10	*
10	SP	7	7	9	10	10	10	$\overline{1}$

5. 실험 결과 및 분석

본 논문의 실험에 사용된 유전자 파라미터 값은 최대 세대수는 1000, P.S.는 50과 70이며, Pc=1, Pm=0.01로 하였다. 표 3은 집단크기(P.S.) 50과 70에 대하여 다섯 개의 랜덤 함수 SEED 값을 적용하여 반복 실험을 한 결과이다. 표3에서 MAX %는 최대 세대수 1000회를 수행하였을 때 나온 최고값이며, MAX.GEN.은 이 때의 세대수이다. 그리고 최적해의 비율 50~90%에 도달할 때의 세대수를 나타내고 있다. 여기서 총 유전자 개수 중에서 최적해의 50%~90%일 때의 세대수를 알아보는 이유는 사용자의 게임 수준에 맞는 게임 수준을 정하기 위해서다. (그림 4)는 <표 3>의 내용 중에서

<표 3> SEED별 규칙 변경 전후의 결과 비교

실험 1. 게임 규칙 1.								
실험	P.S.	MAX %	MAX GEN	50 %	60 %	70 %	80 %	90 %
1	50	92	526	36	67	80	180	361
2		96	938	23	38	81	145	284
3		93	993	29	50	87	151	305
4		98	940	36	68	115	154	345
5		95	619	33	46	77	133	272
평 균		94	803	31	53	88	152	313
6	70	93	591	25	55	86	147	346
7		100	542	31	59	64	68	316
8		100	423	17	31	49	89	163
9		100	778	29	47	101	207	451
10		100	586	33	48	82	170	283
평균		98	584	27	48	76	136	311
실험 2. 게임 규칙 2								
실험	P.S.	MAX %	MAX GEN	50 %	60 %	70 %	80 %	90 %
1	50	100	890	27	49	98	123	203
2		100	538	44	61	101	152	238
3		100	684	42	83	112	173	281
4		100	508	19	54	94	196	347
5		100	659	21	60	88	152	351
평 균		100	655	30	61	98	159	284
6	70	100	667	24	58	62	64	204
7		100	503	27	50	67	187	315
8		100	709	28	52	80	131	259
9		100	740	31	54	93	150	253
10		100	608	25	51	74	125	246
평균		100	645	27	53	75	131	255



(그림 4) PS 70일 때 경우 변경 전후의 평균값 비교

P.S.가 70일 때, 규칙 변경 전후의 평균값의 변화를 그래프로 보여주고 있다. 그림에서 보듯이 지능 캐릭터는 세대가 증가함에 따라 진화하면서 최적해의 90%에 도달하는 세대수는 규칙변경 전이 311세대, 규칙변경 후가 255세대로, 변경 후가 같은 좀더 빠르게 도달하는 것을 알 수 있다. 또, 게임 규칙을 중간에 변경하더라도 지능 캐릭터가 스스로 진화하여 학습하고 있음을 볼 수 있다.

(표 4)와 (표 5)는 SEED 값을 변화시키면서 세대(로그스케일, 1, 2, 4, …, 512)별로 최적해의 몇 %에 접근하는지를 표시한 것이다. 즉, 집단크기가 50이고 게임규칙 변경 전후 각각의 SEED 값과 세대수 별로 최적해에 어느 정도 접근하는지를 %로 표시한 것이다. 본 논문의 목적이 100%의 최적 해를 찾는 것이 아니므로 세대수는 전체 탐색공간의 50% 이전에서 해의 수렴 정도를 알아보기 위해서 512 세대까지만 표시하였다. <표 4>에서 보듯이 게임규칙 변경 전 실험에서는 초기 값이 10% 부근에서 시작하지만 규칙 변경 후의 실험에서는 39% 부근에서 시작하는 것을 볼 수 있다. <표 5>는 같은 방법으로 집단크기가 70일 때 게임규칙 변경전후에 세대별로 해의 수렴정도를 비교 한 것이다. 여기에서도 초기 값이 게임규칙 변경후가 변경전보다 더 높게 나타났다. 이와 같이 게임 규칙 변경후에 초기 값이 높게 나타나는 이유는 게임규칙을 변경하여 다시 게임을 시작하더라도 초기 염색체부터 시작하는 것이 아니라 규칙을 변경하기 전의 진화된 염색체를 이용하기 때문에 게임규칙 변경 후에 적용되는 적합도의 값이 높기 때문이다.

(그림 5)는 <표 4>에서 P.S.가 50일 때 SEED 값이 23인 것과 <표 5>에서 P.S.가 70이고 SEED 값이 23일 때 세대가 증가하면서 최적해에 도달하는 정도를 %로 나타낸 것이다. 그래프의 좌측은 게임 규칙 변경전 1세대부터 512세대까지이고, 우측은 게임 규칙 변경 후 1세대부터 512세대(513~1024)의 결과이다. 그림 5에서 보는 바와 같이 게임 규칙 변경 전에는 꾸준히 상승하다가 게임 규칙이 바뀌는 순간에 성능이 갑자기 떨어지며 게임 규칙이 바뀌고 나서 일정 세대가 지난 후에야 비로서 규칙 변경전의 수준까지 도달함을 알 수 있다. 그림에서 게임규칙 변경 전에는 물론 게임 규칙 변경 후에도 지능 캐릭터는 진화하고, 새로운 환경에도

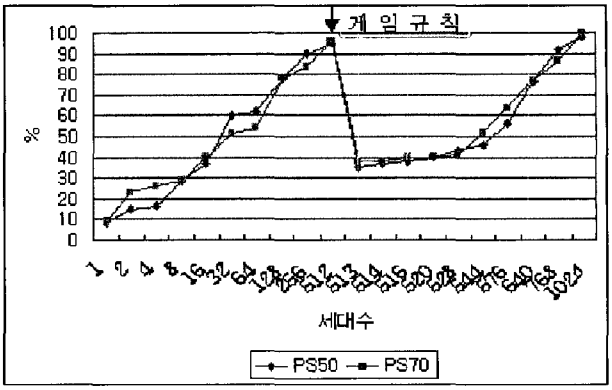
〈표 4〉 P.S. 50일 때 규칙 변경 전후의 비교

GEN / SEED	게임 규칙 1									
	1	2	4	8	16	32	64	128	256	512
11	9	9	15	21	35	46	57	73	84	92
23	9	15	15	29	37	60	62	78	90	95
35	10	20	23	29	45	53	64	76	89	90
47	15	17	18	21	34	50	57	70	87	95
83	7	12	18	26	31	43	68	79	87	95
평균	10	14	17	25	36	50	61	75	87	93
GEN / SEED	게임 규칙 2									
	1	2	4	8	16	32	64	128	256	512
11	41	43	43	43	44	52	70	82	94	97
23	35	37	40	40	43	46	56	76	92	98
35	40	40	41	43	44	47	56	71	86	94
47	38	43	43	44	47	50	65	71	88	100
83	41	43	43	44	47	53	62	79	86	94
평균	39	41	42	42	45	49	61	75	89	96

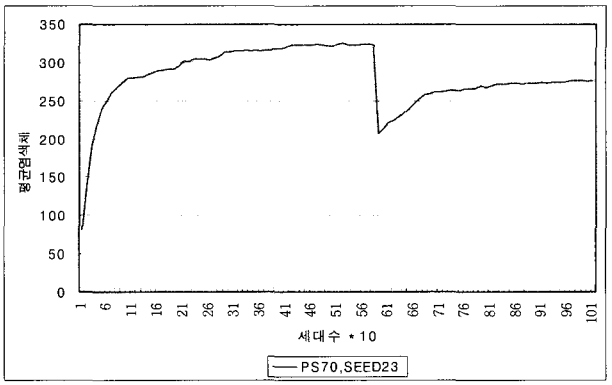
〈표 5〉 P.S. 70일 때 규칙 변경 전후의 비교

GEN / SEED	게임 규칙 1									
	1	2	4	8	16	32	64	128	256	512
11	9	12	18	23	32	51	62	78	84	90
23	9	23	26	29	40	51	54	78	84	96
35	10	26	29	40	45	60	75	85	95	100
47	15	17	20	31	34	46	60	75	84	92
83	15	17	21	31	43	51	65	76	87	98
평균	11	19	22	30	38	51	63	78	86	95
GEN / SEED	게임 규칙 2									
	1	2	4	8	16	32	64	128	256	512
11	37	38	38	40	44	52	62	79	95	97
23	38	38	40	40	41	52	64	77	86	100
35	38	40	41	43	47	50	65	79	88	97
47	37	38	43	43	44	50	64	77	89	95
83	43	43	43	46	49	56	65	85	94	100
평균	43	43	43	46	49	56	65	85	94	100

잘 적응하고 있음을 보여주고 있다. 게임규칙을 변경하자마자 초기값이 떨어지는 이유는 게임규칙이 변경되면 적합도가 떨어지며, 최적해를 찾는 비율도 낮기 때문이다. 한편 0% 근처까지 가지 않는 이유는 규칙이 완전히 다르지 않고, 초기 염색체를 다시 생성하여 시작하는 것이 아니라 규칙을 변경하기 전의 near-optimal에 가까운 염색체를 이용하기 때문이다.



(그림 5) 게임규칙 변경 전후의 적응성 비교



(그림 6) PS 70, SEED 23일 때 게임 규칙

(그림 6)은 집단크기가 70이고 SEED가 23일 때 게임규칙 변경하기 전후의 평균 적합도의 변화를 보여주고 있다. 그림의 좌측이 게임규칙 변경전이고, 우측이 변경후의 평균 적합도이다. 규칙 변경전의 542 세대 이후, 변경후의 503 세대 이후에는 데이터가 존재하지 않는다. 이는 최대 세대수 중에서 가장 좋은 해를 찾은 세대수 이후에는 더 이상 실행하지 않도록 했기 때문이다(표 3 참조). 게임 규칙 변경후에 일정 세대가 지나더라도 평균 적합도가 낮게 수렴되는 것은 규칙 변경후의 규칙에서 공격에 대한 배점을 낮게 했기 때문이다. 게임규칙 변경 전에 5가지 공격으로 한번씩 하여 얻을 수 있는 점수는 2+2+4+5+5+5=23이고, 변경 후 얻을 수 있는 점수는 5+5+3+3+1+1=18이다(표 1 참조).

6. 결 론

본 논문에서는 유전자 알고리즘을 이용하여 진화를 통하여 스스로 게임의 규칙을 학습하고 게임의 규칙이나 환경변화에도 스스로 적응해 나가는 지능형 캐릭터를 구현하였다. 제안한 유전자 알고리즘을 이용한 게임에서 지능 캐릭터는 초기에는 게임 규칙을 모르는 상태에서 상대 캐릭터와 게임을 시작하지만 일정한 세대가 반복됨에 따라 게임규칙을 학습하고, 진화하면서 새로운 환경에 적응할 수 있음을 확인하였다. 지능 캐릭터의 진화를 통한 적응 능력을 평가하기

위한 방법으로는 게임 도중에 게임규칙을 바꾸어 규칙 변경 전후의 결과를 비교 검토하였다. 게임 규칙을 변경하기 전의 결과에서 지능 캐릭터는 게임규칙을 학습하여 진화하였으며, 규칙 변경 후에도 즉, 새로운 환경에서도 스스로 진화하면서 잘 적응함을 보였다. 게임에서 지능 캐릭터의 역할은 무조건 이기는 것이 아니라(최적해를 찾는 것이 아니라) 상대방과 비슷한 수준으로 게임을 할 수 있게 적절한 해를 찾는 것이다. 그러므로 실제 게임에 적용할 경우에 목표값을 조정하여 사용자의 수준에 맞게 수준을 조정할 수 있도록 하였다. 향후 과제로는 실제적인 게임 환경에 적용시켜 보는 것과 온라인 게임과 같은 다양한 게임에서 게임 도중에 규칙을 변경하더라도 스스로 학습하여 규칙을 찾아내는 지능 캐릭터를 개발하는 것이다.

참 고 문 헌

[1] Daniel Johnson, Janet Wiles, "Effective Affective User Interface Design in Games", International Conference on Affective Human Factors Design, Singapore

[2] John E. Laird, "Using a Computer Game to Develop Advanced AI," IEEE Computer, July, 2001, pp.70-75, 2001.

[3] S. Woodcock, "Game AI : The State of the Industry," Gamasutra Magazine Nov., 2000, Vol.01.

[4] D.C. Pottinger and John E. Laird, "Game AI : The State of the Industry Part 2," Gamasutra Magazine Nov., 2000, Vol. 08.

[5] D.Carmel and S.Markovitch, "Learning Models of Opponent's Strategy in Game Playing," CIS Report #9318, 1993.

[6] John E. Laird and M.van Lent, " Human-level AI's Killer Application : Interactive Computer Games," Proc. AAAI 2000, AAAI Press/MIT Press, pp.1171-1178.

[7] I.Faybish, "Applying the Genetic Algorithm to the Game of Othello," VRIJE University.

[8] 이만재, "게임에서의 인공지능 기술," 한국정보처리 학회지, Vol.9, No.3, pp.69-76 May, 2002.

[9] Daniel Johnson, Janet Wiles, "Computer Games With Intelligence", IEEE International Fuzzy Systems Conference, 2001.

[10] Darrell Whitley "An Overview of Evolutionary Algorithms" Journal of Information and Software Technology. 43: pp. 817-831, 2001.

[11] Daniel Johnson, Janet Wiles, "Computer Games with Intelligence" Fuzz-IEEE, 2001, Melboorne, Australia.

[12] T. N. BUi and B. R. Moon, "On multi-dimensional encoding/ crossover", International Conference on Genetic Algorithms, pp49056, 1995.

[13] C. Anderson, k. Jones, and J. Ryan, "A two-dimensional genetic algorithm for the Ising problem", Complex system, 5:327-333, 1991.

[14] L. J. Eshelman, R. A. Caruana and J. D. Schaffer, "Bases in the Crossover Landscape," Proc. 3rd Int. Conf. on Genetic Algorithms, LA. pp.10-19, 1989.

[15] Andrew B. Kahng and Byung Ro Moon, "Toward More Powerful Recombinations", Proceedings of the Sixth International Conference on Genetic Algorithms.

이 면 섭

e-mail : leems@icc.ac.kr

1985년 국민대학교 전자공학과(공학사)

1987년 인하대학교 전자공학과(공학석사)

1999~현재 국민대학교 전자공학과 박사과정

1990~현재 인천전문대학 컴퓨터정보과 전임강사, 조교수, 부교수

관심분야 : 유전자 알고리즘, 게임, 컴퓨터 그래픽

조 병 헌

e-mail : d995552@hanmail.net

1997년 국민대학교 전자공학과(공학사)

1999년 국민대학교 전자공학과(공학석사)

2005년 국민대학교 전자공학과(공학박사)

관심분야 : 유전자 알고리즘, 시뮬레이션, 신경망

성 영 락

e-mail : yeong@kookmin.ac.kr

1989년 한양대학교 전자공학과(공학사)

1991년 한국과학기술원 전기 및 전자공학과(공학석사)

1995년 한국과학기술원 전기 및 전자공학과(공학박사)

1995년~1996년 한국과학기술원 위촉연구원

1996년~1998년 국민대학교 전자공학부 전임강사

1998년~2002년 국민대학교 전자공학부 조교수

2002년~현재 국민대학교 전자정보통신공학부 부교수

관심분야 : 시뮬레이션, 고장감내, 내장형 시스템

정 성 훈

e-mail : hjung@hansung.ac.kr

1988년 한양대학교 전자공학과(공학사)

1991년 한국과학기술원 전기 및 전자공학과(공학석사)

1995년 한국과학기술원 전기 및 전자공학과(공학박사)

1995년~1996년 한국과학기술원 전기및전자공학과(위촉연구원)

1996년~1998년 한성대학교 정보전산학부 정보통신공학전공 전임강사

1998년~2002년 한성대학교 정보전산학부 정보통신공학전공 조교수

2002년~현재 한성대학교 정보통신공학과 부교수

오 하 령

e-mail : hroh@kookmin.ac.kr

1983년 서울대학교 전기공학과(공학사)

1983년~1986년 삼성전자 종합연구소

1988년 한국과학기술원 전기전자과 컴퓨터공학전공(공학석사)

1992년 한국과학기술원 전기전자과 컴퓨터공학전공(공학박사)

1992년~1996년 국민대학교 전자공학부 조교수

1996년~2001 국민대학교 전자공학부 부교수

2001년~현재 국민대학교 전자정보통신공학부 교수

관심분야 : 병렬처리, 내장형 시스템, 고장감내