유전자 알고리즘을 이용한 테트리스 AI 기법의 설계 및 구현

박종길* · 이성실* · 최경암* · 최준혁* · 김진일**

*동의대학교

Design and Implementation of AI methodologies for Tetris Game using Genetic Algorithm

Jong-Kir Park* · Seong-Sil Lee* · Kyoung-Am Choi* · Jun-Hyeok Choi* · Jin-Il Kim**

*Dong-Eui University

E-mail : jk8817@naver.com, jikim@deu.ac.kr

요 약

유전자 알고리즘을 이용하여 스스로 테트리스 게임을 플레이하는 AI 기법을 제안한다. 테트리스에 필요한 요소들을 고려하여 각 요소마다 가중치를 곱한 값을 통해 블록을 이동시킬 자리를 정한다. 해당 알고리즘은 8가지의 고려 요소를 가지며, 각 요소별 최적의 가중치를 구하기 위해 유전자 알고리즘을 적용하였다. 본 연구의 성능을 분석하기 위하여 직접 설계·제작한 테트리스로 게임을 정확하게 진행해 나가는가를 실험하였다. 실험 결과, 제안 기법에 따라 테트리스를 진행하는 것을 확인하였다.

키워드

테트리스, 고려 요소, 가중치, 유전자 알고리즘

Ⅰ. 서 론

최근, 인공지능의 비약적인 발전으로 이미 사회의 광범위한 분야에서 사람을 대신하는 AI들이 자리 잡고 있으며, 그 중에서도 가장 큰 이슈가 되었던 것은 구글의 딥 마인드가 개발한 바둑 AI 프로그램 '알파고'였다. 알파고는 2016년 3월, 현재 세계 최상위 수준급의 프로 기사 이세돌 9단과의 5번의 대국에서 최종 전적 4승 1패로 승리하며 현존 최고의 AI로 등극하여 세계를 놀라게 하였고, 이 대국을 통해 인공지능의 새 장을 열었다는 평가를 받았다.

이와 같이 인간의 오락 분야로 자리하고 있는 게임들에도 많은 인공지능들이 등장하고 있는데, 그 중 가장 대중적인 게임 중 하나인 테트리스에 유전자 알고리즘을 적용시켜, 테트리스를 진행하도록 하였다. 테트리스를 플레이하는데 필요한 요소는 여러 가지가 있는데, 이번 실험에서는 8가지의 고려 요소를 두고 AI를 설계하였다. 본 연구에서는 각 고려 요소들의 채택과, 해당 요소들이 어떻게 유전자 알고리즘에 적용하는지에 대하여 평가하고, 이를 테트리스 게임에 적용하였다.

Ⅱ. AI 기법의 적용 설계

1. AI 기법의 적용 요소

AI 스스로 블록을 이동시키려면 블록의 이동을 정하도록 할 해가 필요로 하며, 그 해를 구하기 위한 고려 요소들이 있어야 한다. 본 실험에서 고려 요소들은 라인 클리어에 방해 요소로 작용하거나 AI가 게임 오버가 되는 원인들을 보며 추가하였다.

본 AI 기법의 적용에는 8가지의 고려 요소를 포함하도록 했는데 해당 요소들은 다음과 같다.

표 1. AI에 사용된 요소

요소	내용
sumHeight	높이의 합
maxHeight	최대 높이
(max-min)Height	최대, 최저 높이 차
sumHeight/length	평균 높이
holes	빈 공간의 수
wells	우물
blockades	라인 별 블록 수
clears	클리어 되는 라인 수

위의 표에 언급된 고려 요소들에 적당한 가중치를 곱하여 나오는 값을 해로 둔다.

AI는 블록을 각 라인별로 이동시켰을 때의 해를 구하고 가장 해가 큰 자리로 블록을 이동시킨다. 이를 위한 최적의 해를 만들어내는 가중치를 유전자 알고리즘을 통해 구한다. 가중치의 적합도는 해당 개체의 최종 스코어에 기반을 둔다.

2. 유전자 알고리즘의 연산

한 세대는 16개의 개체를 가지며, 각 개체는 8가지 고려 요소를 유전자로 가진다. 최초의 세대의 유전자는 난수 값으로 생성된다.

한 개체는 고려 요소들에 가중치를 곱한 값을 기반으로 블록을 이동 시키며, 게임 종료 되었을 때까지 클리어 한 라인 수를 기반으로 최종 점수를 가지게 된다.

선택 연산 과정에서 부모 세대에서 최종 점수가 가장 높은 상위 4개 개체는 다음 세대에서도 그대로 사용하게 되며, 나머지 개체의 유전자는 부모 세대의 유전자를 균등 교차 방식을 이용하여 가지게 된다. 이 과정에서 상위 개체를 제외한 나머지 개체의 유전자에서 10%의 확률로 돌연변이가 발생하도록 설계하였다.

Ⅲ. 테트리스 게임의 적용

본 실험에서는 설계한 AI의 성능을 분석하기 위해 직접 Java 언어를 사용, 이클립스를 통해 테트리스 게임을 제작하였다. 제작한 테트리스 게임은 기본적으로 10X20 사이즈의 보드를 가지며 보드의 사이즈, AI의 적용 여부 등을 설정 가능하게 설계하였다. 그림 2는 AI를 적용시킬 테트리스 게임의 UI 모습이다.

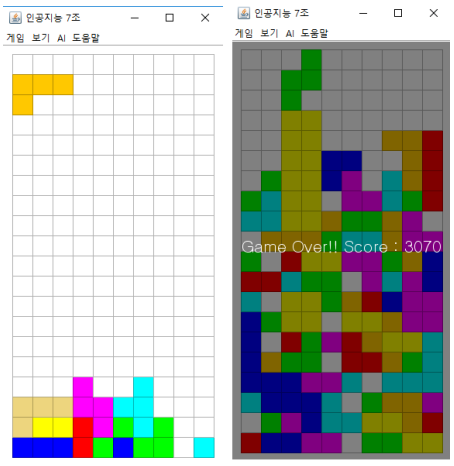

그림 1. 테트리스 실행 모습

각 세대별 개체의 유전자 값은 부모 세대가 끝나고 시작되면서 txt파일로 저장되며, 이클립스를 통해 세대별 최고, 중간, 최저 점수와 각 개체 별 점수를 확인할 수 있다.

Ⅳ. 결 론

본 실험에서는 유전자 알고리즘을 이용하여 AI 기법을 구현하였으며, 이를 테트리스 게임에 적용하였다.

테트리스 게임 진행에 필요한 요인들을 고려요소로 삼았으며, 이에 가중치를 적용시킨 해를 기반으로 블록의 이동을 결정했다. 또, 최적 해에 접근하기 위해 상위 개체를 계속 가져오는 방법을 사용했다. 연구 결과, AI가 정상적으로 테트리스를 플레이해 나감을 확인했다.

본 연구의 결과는 향후, 지능을 요구하는 다양한 게임에서 적용해 볼 수 있을 것으로 기대된다.

표 2. 테트리스 게임 결과(1세대, 23세대)

```
Generation 1  - Candidate 1 : score = 0
Generation 1  - Candidate 2 : score = 4
Generation 1  - Candidate 3 : score = 0
Generation 1  - Candidate 4 : score = 6
Generation 1  - Candidate 5 : score = 0
Generation 1  - Candidate 6 : score = 0
Generation 1  - Candidate 7 : score = 0
Generation 1  - Candidate 8 : score = 3
Generation 1  - Candidate 9 : score = 1
Generation 1  - Candidate 10: score = 0
Generation 1  - Candidate 11: score = 0
Generation 1  - Candidate 12: score = 3136
Generation 1  - Candidate 13: score = 0
Generation 1  - Candidate 14: score = 0
Generation 1  - Candidate 15: score = 0
Generation 1  - Candidate 16: score = 0
Generation 1  - max = 3136, med = 0, min = 0

Generation 23 - Candidate 1 : score = 1880894
Generation 23 - Candidate 2 : score = 439647
Generation 23 - Candidate 3 : score = 75853
Generation 23 - Candidate 4 : score = 3302
Generation 23 - Candidate 5 : score = 3
Generation 23 - Candidate 6 : score = 1
Generation 23 - Candidate 7 : score = 51644
Generation 23 - Candidate 8 : score = 1
Generation 23 - Candidate 9 : score = 0
Generation 23 - Candidate 10: score = 10670
Generation 23 - Candidate 11: score = 7
Generation 23 - Candidate 12: score = 222
Generation 23 - Candidate 13: score = 129
Generation 23 - Candidate 14: score = 43999
Generation 23 - Candidate 15: score = 120304
Generation 23 - Candidate 16: score = 5036
Generation 23 - max = 1880894, med = 3302, min = 0
```

표 3. 각 세대별 최고, 중간, 최저 점수

```
Generation 1  - max = 3136, med = 0, min = 0
Generation 2  - max = 2114, med = 0, min = 0
Generation 3  - max = 11072, med = 17, min = 0
Generation 4  - max = 7033, med = 474, min = 0
Generation 5  - max = 40817, med = 1658, min = 53
Generation 6  - max = 136156, med = 1358, min = 37
Generation 7  - max = 202364, med = 995, min = 0
Generation 8  - max = 154899, med = 20415, min = 0
Generation 9  - max = 121751, med = 18840, min = 0
Generation 10 - max = 195534, med = 12003, min = 0
Generation 11 - max = 209659, med = 8312, min = 0
Generation 12 - max = 192457, med = 5577, min = 0
Generation 13 - max = 135023, med = 26541, min = 250
Generation 14 - max = 207593, med = 42185, min = 0
Generation 15 - max = 228453, med = 53241, min = 19
Generation 16 - max = 178107, med = 39390, min = 861
Generation 17 - max = 612100, med = 11073, min = 0
Generation 18 - max = 1067104, med = 43894, min = 3335
Generation 19 - max = 732524, med = 30197, min = 0
Generation 20 - max = 794819, med = 20010, min = 0
Generation 21 - max = 2001331, med = 26113, min = 0
Generation 22 - max = 1921193, med = 3920, min = 12
Generation 23 - max = 1880894, med = 3302, min = 0
```

참고문헌

[1] Cendrowska, J. “PRISM:An Algorithms for Inducing Modular Rules”, Int. J. of Man-Machine Studies, Vol. 27 pp.349-379, 1987.

[2] 문병로, “쉽게 배우는 유전 알고리즘 : 진화적 접근법”, 한빛아카데미, 2014.

[3] Tistory, “유전 알고리즘으로 테트리스 AI 최적화하기”, http://orcacode.tistory.com/8, 2016. 08