게임 몬스터 생성에 적합한 유전알고리즘

Genetic Algorithm for Game Monster Generation

박상욱, 이원형
중앙대학교

Park Sang-Wook, Lee Won-Hyung
Chung-Ang University

요약

대부분의 게임에는 플레이어를 상대하는 몬스터가 존재한다. 이 몬스터는 대부분 미리 정해진 방법과 데이터로 생성되며, 환경이나 플레이어에 적응하는 방식은 거의 없었다.

본 논문에서는 몬스터 생성을 위해 개선된 유전 알고리즘을 소개한다. 이 알고리즘은 상동염색체 구조가 적용되어있다. 기존의 유전알고리즘에서 각 개체가 오직 하나의 genome만을 가지고 있다. 하지만, 상동 염색체 구조를 가지고 있는 유전 알고리즘에서는 각 개체가 각 좌위에 한 쌍의 대립 유전자를 지니게 된다.

단순한 유전알고리즘과 개선된 유전알고리즘을 비교하기 위해 간단한 이진 문제를 가지고 시뮬레이션 해 보았다. 실험결과 제안된 알고리즘이 더 적은 세대 수로 답을 찾을 수 있다는 것을 알게 되었다.

Abstract

There are Monsters for game player in computer game. The Monsters generated by steady methods and data. And There are few methods that can be adapted to environment or player.

This paper introduces a reformed Genetic Algorithm for Monster generation. This algorithm is applied to Homologous Chromosomes(HC). In existing GAs, An Individual have only one genome. But, In proposed algorithm, each Individual has a pair of allele genes on each locus.

To compare proposed algorithm with Simple Genetic algorithm, I simulated the solution of a simple Binary problem. After experiments, I conclude that the suggested Algorithm reduced the number of generations more than SGA.

I. 서 론

Holland가 유전알고리즘(Genetic algorithm)의 기본적인 형태에 대해 제안한 후, 유전알고리즘은 여러 분야에서 사용되었다[1],[2],[3].

유전알고리즘은 세대(Generation)의 수가 많이 지나야 적합한 해를 찾아낼 수 있다는 단점이 있다. 결과적으로 처리 시간은 답을 위해 세대수가 얼마나 필요한가에 달려있다고 볼 수 있다.

또한 적합도 평가(Fitness evaluation) 방법이 고정되어있지 않고 시간이 지남에 따라 변하는 알고리즘도 있다. Evolutionary Art 나 Genetic Art에서 이런 경우가 많다[2]. Evaluation을 사람이 직접 하는 경우도 많으며, 이런 경우 Evaluation Function이 정의되지 않는다[3].

특히 본 논문에서 논의하는 컴퓨터 게임의 경우 이런 현상이 더욱 심하다고 할 수 있겠다. 현재까지 유전알고리즘을 적용한 게임이 많이 있지는 않다. 하지만 유전알고리즘이 적용된 소수의 게임을 보면, 게임이 진행됨에 따라 적합도 평가에 해당되는 방법이 매우 불규칙적으로 바뀐다는 것을 알 수 있다.

이처럼 Evaluation이 변하는 프로그램에서는 적은 세대 수로도 좋은 결과를 얻는 것이 중요하다.

본 논문에서는 상동염색체(Homologous Chromosome)를 적용한 알고리즘을 소개한다. 시뮬레이션을 해 본 결과 기존의 GA보다 Generation Number가 약 27.9%감소했다.

2장에서는 기본적인 유전알고리즘에 대해 간략히 소개한다. 3장에서는 몬스터 생성에 최적화된 유전알고리즘에 대해 소개한다. 4장에서는 제안한 알고리즘의 성능을 예측하기 위해 실시한 실험에 대해 서술하고 5장에서 결론을 내린다.

2. 기존 연구

2.1 유전알고리즘

유전알고리즘들은 조금씩 다르고 해결할 문제에 따라 변형될 수 있다. 하지만 전체적인 알고리즘은 같다. 아래에 일반적인 유전알고리즘의 중요한 실행과정을 순서대로 나열했다.

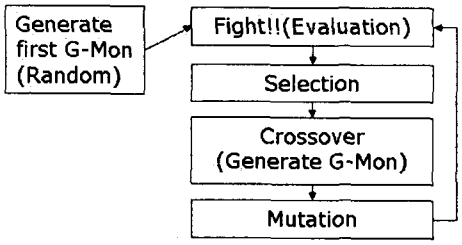
(1) 초기 집단 생성 Initial population generation

(2) 적합도 평가 Fitness evaluation

(3) 선택 Selection
(4) 교배 Crossover
(5) 돌연변이 Mutation
(2)-(5)의 과정이 반복되게 된다.

2.2 유전알고리즘을 이용한 몬스터 생성

2.1의 유전알고리즘을 사용해 몬스터를 생성할 때의 개략적인 처리과정을 아래의 그림에 나타내었다.



▶▶ 그림 1. 유전알고리즘을 사용한 몬스터 생성의 처리과정

3. 몬스터 생성에 최적화된 유전알고리즘

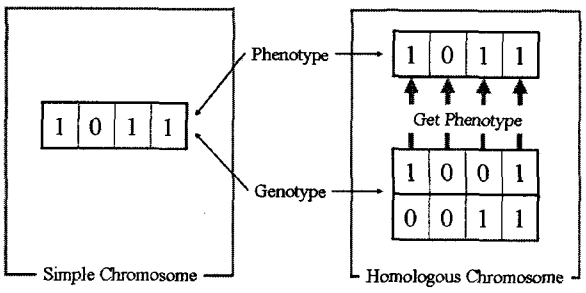
3.1 제안한 알고리즘

기존의 유전알고리즘에서 각 개체는 상동염색체 구조가 아닌 하나의 유전자만을 가지고 있었다. 하지만, 상동염색체 구조를 적용한 알고리즘에서는 각 좌위(locus)에 대립 유전자(allele gene)가 한 쌍씩 존재하게 된다. 각 좌위의 유전자가 같을 경우 표현형(Phenotype)은 둘 중 하나의 유전자(gene)로 표현하면 된다. 하지만 서로 다를 경우 우성유전자(Dominant gene)가 어떤 것인지 판단하여 표현형(Phenotype)을 결정하게 된다. 때문에 표현형(Phenotype)을 알기 위한 함수가 필요하게 된다. 또한 교배(Crossover)단계에서도 다른 알고리즘을 사용할 수 있다.

1) 몬스터 생성

2) 상동염색체(Homologous Chromosome)구조의 적용

그림 2에 상동염색체와 기존의 염색체 구조를 나타내었다.



▶▶ 그림 2. 기존의 염색체 구조와 제안한 상동염색체 구조. 제안한 구조는 대립유전자를 지니고 있다.

기존의 유전자 구조는 각 좌위(locus)에 오직 하나의 유전자(Gene(Data))만을 가지고 있다. 하지만 제안한 알고리즘에서는 표현형으로 쓰이지 않는 유전자를 가지고 있다.

이러 이유로 제안한 알고리즘에서는 따로 표현형을 얻어내야 한다. 표현형을 얻어내는 처리는 각 좌위별로 이루어진다. 여기서는 각 좌위별로 OR연산을 통해 표현형을 구한다.

3) 초기 집단 생성 Initial population generation

한 세대는 여러 개체로 이루어지며, 각 세대의 개체는 알고리즘 설계에 따라 달라진다. 여기서는 12개체를 사용하였다. 초기 값은 랜덤함수에 의해 주어진다. 랜덤함수의 seed값은 시간함수에 의해 입력된다.

4) 적합도 평가 Fitness evaluation

적합도 평가점수 S는 승리 점수 w, 전투시간 점수 t, 남은 체력 점수 Hp의 합으로 결정된다. 식 1에 나타내었다.

$$S = w + t + Hp \tag{1}$$

승리 점수 w는 플레이어가 이겼을 때 20, 플레이어가 졌을 때 -20, 전투가 중간에 중단되면 5점을 주는 것으로 했다. t는 60초 정도의 시간이 전투시간으로 적합하다고 보고, 이 시간과 실제 전투시간의 차이를 30에서 빼서 구한다. 식 2에 나타내었다.

$$t = 30 - |60 - t_{fight}| \tag{2}$$

Hp는 전투 후 50%정도 남는 것이 적당하다고 보고 50에서 남은 체력의 비율을 빼서 구했다.

5) 선택 Selection

선택은 Ranking-based selection을 기초로 한다[4],[5].

(1) Evaluation 단계의 Score가 높은 순서로 Individual을 정렬한다.
(2) 점수가 높은 6개의 Individual을 선택한다.
(3) 나머지 Individual은 버린다.
(4) 점수가 높은 순서대로 짝을 지어 Crossover 단계로 넘어간다.

6) 교배 Crossover

각 부모개체의 상동염색체를 분리하는 단계에서 Cross 지점을 선택하여 crossover하는 과정을 거친 뒤, 두 염색체를 무작위로 결합시킨다.

7) 돌연변이 Mutation

일반적인 돌연변이 방법을 사용한다. 모든 유전자에 대해 무작위로 돌연변이를 일으킨다.

4. 실험

4.1 실험 설계

몬스터는 힘, 체력, 스피드의 데이터를 각각 4비트씩 총 12비트로 저장하게 된다. 이 12비트가 몬스터의 유전자가 된다.

일반적인 유전알고리즘에서는 위의 12비트만 가지고 있으면 알고리즘을 수행할 수 있다. 하지만 제안한 유전알고리즘을 적용하기 위해서는 각 비트에 대립하는 대립유전자로서 같은 크기의 유전자를 같이 지니고 있다. 그리고 앞서 설명했듯이 각 비트의 값을 구하기 위해서는 서로 대립하는 두 비트의 우열을 가려 표현형을 구하게 된다.

4.2 의사 코드

기본 유전알고리즘의 의사코드는 아래와 같다.

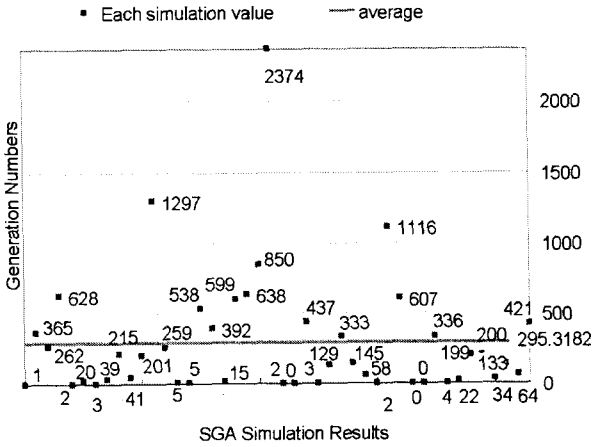
```
int Solution[12] = {1,0,0,1,0,1,1,1,0,0,0,1};
int gene[12][12];
initial();
srand((unsigned int)time(NULL)/2);
for(int end=0; end!=1; age++){
  Fitness_Evaluation();
  Sort_by_score();
  /*Crossover*/
  for(int i=0; i<6; i=i+2){
    int crossover;
    crossover = rand()%2;
    for(int j=0; j<12; j++){
      if(j <= cross){
        gene[i+6][j] = gene[i][j];}
      else{
        gene[i+6][j] = gene[i+1][j];}
    }
  }
  mutation();
}
```

제안한 알고리즘의 의사코드는 아래와 같다.

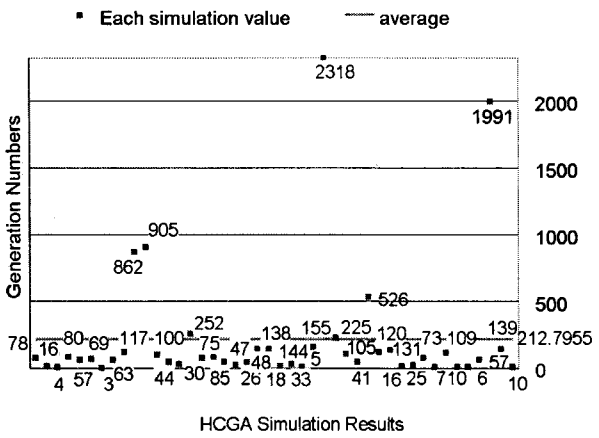
```
int Solution[12] = {1,0,0,1,0,1,1,1,0,0,0,1};
int gene[12][12][3];
initial();
srand((unsigned int)time(NULL)/2);
for(int end=0; end!=1; age++){
  /* Getting Phenotype */
  for(int i=0; i<12; i++){
    for(int j=0; j<12; j++){
      if(gene[i][j][0] || gene[i][j][1])
          gene[i][j][2] = 1;
      else gene[i][j][2] = 0;
    }
  }
  Fitness_Evaluation();
  Sort_by_score();
  /* Crossover */
  for(int i=0; i<6; i=i+2){
    meiosis1 = rand()%2;
    meiosis2 = rand()%2;
    for(int j=0; j<12; j++){
      gene[i+6][j][0] = gene[i][j][meiosis1];
      gene[i+6][j][1] = gene[i][j][meiosis2];
    }
  }
  mutation();
}
```

4.3 실험 결과

실험결과를 아래 표에 나타내었다. 실험결과 제안한 알고리즘이 약 27.9% 줄어들었다.



▶▶ 그림 3. 기존의 유전알고리즘을 사용한 실험결과



▶▶ 그림 4. 제안한 알고리즘을 사용한 실험결과

5. 결론

컴퓨터 게임에서의 몬스터 생성을 위해 최적화시킨 유전알고리즘을 제안하고 실험해 보았다. 제안한 유전알고리즘은 기존의 알고리즘과는 달리 대립유전자를 포함하는 염색체 구조를 갖고 있으며, 이것이 성능에 좋은 영향을 미친다는 결론을 내렸다.

일반적인 게임의 전투를 생각해 알고리즘을 만들어보았다. 하지만 실제 게임에 적용을 하기 위해서는 약간의 변형이 필요할 것이다.

▌참 고 문 헌▐

[1] J.H.Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, Michigan,

[2] Altenberg L, "The schema theorem and Price's theorem", Foundations of Genetic Algorithms vol.3. ed L D Whitley and M D Vose, San Mateo, CA: Morgan Kaufmann, pp.23-49, 1995

[3] Atmar W, "Notes on the simulation of evolution", IEEE Trans, Neural Networks, NN-5 130-47, 1994

[4] Gary Greenfield, "Robot Paintings Evolved Using Simulated Robots", Applications of Evolutionary Computing, EvoWorkshops 2006, Springer. pp.611-621, 2006

[5] Tristan Basa, Christian Anthony Go, Kil-Sang Yoo, Won-Hyung Lee, "Using Physiological Signals to Evolve Art", Applications of Evolutionary Computing, EvoWorkshops 2006, Springer. pp.632-641, 2006

[6] J.E.Baker, "Adaptive Selection Methods for Genetic Algorithms", Proc. 1st Int. Conf. on Genetic Algorithms and Their Applications, Pittsburgh. PA, pp.101-111, 1985

[7] D. Whitley: "The Genitor Algorithm and Selection Pressure: Why Rank-based Allocation of Reproductive Trials is Best", Proc. 3rd Int. Conf. on Genetic Algorithms and Their Applications, Arlington. VA, pp.116-121, 1989