한국컴퓨터정보학회 하계학술대회 논문집 제18권 제2호 (2010. 7)

유전자 알고리즘을 이용한 경로찾기 시뮬레이션 시스템 설계 및 구현

강명주°, 박광용*
°청강문화산업대학 컴퓨터게임과
*경기대학교 관광전문대학원
e-mail: mjkkang@ck.ac.kr, pky0626@naver.com

Design and Implementation of Genegtic Algorithm Simulation System for A Path Finding

Myung-Ju Kang°, Kwang-Yong Park*
°Dept of Computer Game, ChungKang College of Cultural Industries
*Graduate School of Tourism Science, Kyonggi University

● 요 약 ●

게임이나 네비게이션 시스템, 관광경로 설계에 있어서 경로찾기는 매우 중요한 부분 중의 하나이다. 일반적으로 TSP(Traveling Salesman Problem), RPP(Rural Postman Problem), CPP(Chinese Postman Problem)와 같은 경로찾기 문제들은 일반적인 알고리즘으로 최적해를 구할 수 없다. 문제크기가 커질수록 해집합이 폭발적으로 커짐으로써 전체 해집합을 탐색하는데 많은 비용이 든다. 따라서, 이러한 문제들은 유전알고리즘이나 Simulated Annealing과 같은 휴리스틱 알고리즘을 이용하여 근사최적경로를 찾는다. 본 논문에서는 이와 같은 경로찾기 문제의 근사 최적해를 구하기 위한 시뮬레이션 시스템을 설계하고 구현하였다. 본 연구에서 구현한 시뮬레이션 시스템에는 유전알고리즘 엔진(GA 엔진)과 사용자 인터페이스를 제공한다. 사용자 인터페이스는 유전알고리즘에 사용될 파라미터를 설정하는 부분이며, GA 엔진은 유전알고리즘의 연산자들을 제공하는 부분이다. 본 논문에서 구현한 시뮬레이션 시스템은 게임과 같은 경로찾기 등에 활용될 수 있다.

키워드: 경로찾기(Path Finding), 유전알고리즘(Genetic Algorithm), 시뮬레이션시스템(Simulation System)

I. 서론

게임이나 네비게이션 시스템, 관광경로 설계에 있어서 경로찾기는 매우 중요한 부분 중의 하나이다. 경로찾기는 문제 크기에 따라 최적해를 구하는데 많은 비용이 든다.

지금까지 경로찾기 해법을 위한 알고리즘들에 대한 연구가 많이 진행되어 왔다. TSP (Traveling Salesman Problem), RPP (Rural Postman Problem), CPP (Chinese Postman Problem) 등이 대표적인 경로찾기 문제들이다. 이러한 문제들은 문제 크기가 커짐에 따라 해집합이 폭발적으로 늘어나 일반적인 알고리즘으로 최적해를 찾기 힘들다.

특히 게임에서는 다양한 오브젝트와 벽 등의 장애물로 구성된 복잡한 게임 맵에서 캐릭터나 NPC(Non Player Character)가 목적지까지 길을 찾아 이동하기 위해서는 장애물을 회피하면서 가능한 최단 경로로 이동해야 한다. 게임에서의 경로찾기에서 가장 기본적인 방법은 현재 위치에서 임의의 방향으로 이동시키는 것이다. 이 방법은 주인공 캐릭터가 목적지까지 갈 수 있다는 것을 보장해 주지 못하며, 목적지까지 도달하더라도 최단 경로로 이동하는 경우는 매우 드물다[1]. 또 하나의 방법은 스택을 이용한 깊이우선탐색 방법을 사용한다. 깊이우선 탐색은 맵 트리에서 깊이를 먼저 탐색하는 방법이다. 최악의 경우 맵 전체를 탐색해야하는 경우가 발행할 수 있다. 또한 주인공 캐릭터가 목적지까지 도달할 수는 있지만 최단 경로를 찾는 것이 힘들다[1].

지금까지 많이 사용되는 길찾기 방법은 A* 알고리즘이다. A* 알고리즘은 주인공 캐릭터가 시작 위치에서 현재 도착한 위치까지의 거리와 앞으로 이동해야 할 목적지까지 거리의 추정치를 더하여 최소값을 갖는 경로로 이동시키는 방법이다[2]. 현재 위치에서 앞으로 남은 목적지까지 남은 거리의 계산은 일반적으로 유클리디안 거리로 계산된다. 따라서, 현재 위치까지의 정보는 계속 갱신이 되며 앞으로 남은 목적지까지의 거리 추정치는 계속 계산해야 한다. 또한, 주인공 캐릭터가 벽이나 장애물을 만났을 때 이를 회피하여 다시 계산해야하는 단점이 있다.

최근에는 유전알고리즘과 같은 휴리스틱 알고리즘을 이용한 경로찾기 방법이 연구되고 있다[3]. 유전알고리즘은 해 공간을 병렬로 탐색할 수 있으며 지역최소해와 계산의 복잡도를 줄일 수 있는 장점이 있다. 그러나, 전역최적해로의 수렴과정에서 급속한 지역 최소해로의 수렴이 발생할 수 있다. 이로 인해 복잡한 게임 맵에 적용할 경우 목적지까지의 경로 탐색을 실패할 수 있다는 단점이 있다.

본 연구에서는 유전알고리즘을 이용하여 근사 최적의 경로를 찾는 해법을 시뮬레이션할 수 있는 시스템을 설계하고 구현하였다.

II. 유전자 알고리즘

2.1 유전자 알고리즘 개요

유전자 알고리즘(Genetic Algorithms, GAs)은 탐색 및 최적화 문제를 푸는데 사용되는 적응방법으로써, 생물학적인 유기체의 유전학적인 처리를 기반으로 하는 휴리스틱 알고리즘이다[4, 5].

유전자 알고리즘은 다윈의 진화론을 기본으로 만들어진 알고리즘이다. 즉, 자연의 진화와 적자생존의 원리를 적용한 것으로, 환경변화에 잘 적응하고 우성인 개체는 계속 살아남아서 자손을 번식시키면서 진화해 나가고, 그렇지 못한 열성인 개체는 자연도태된다는데 착안해서 프로그램 알고리즘으로 만든 것이다.

프로그램화된 유전자 알고리즘에서의 염색체는 해결하고자 하는 문제의 특성에 따라 스트링으로 코드화된 개체로 표현하며, 각 염색체는 그 문제의 해 집합 중의 하나가 된다. 유전자 알고리즘에서는 각 염색체의 우성과 열성의 정도를 수치화된 값으로서 결정하여 다음 세대로의 진화과정에 반영한다. 이러한 수치화된 값은 해결하고자 하는 문제의 목적 값(예를 들면, TSP나 RPP인 경우에는 총 라우팅 비용)을 이용하여 각 염색체들의 상대적인 수치로서 계산되며, 우성과 열성의 정도를 판단한다. 또한, 유전자 알고리즘에서의 진화 과정은 현재 세대에서 상대적으로 우성인 개체들을 선택하여 교배를 통해 다음 세대로 진화하는데, 유전자 알고리즘에서의 교배는 교차연산이나 돌연변이연산과 같은 유전 연산자(Genetic Operator)를 통해 이루어진다. 이 과정에서 상대적으로 열성인 개체들은 다음 세대에서 배제가 되고, 상대적으로 우성인 개체들만을 이용하여 빠른 시간 내에 최적해로 수렴할 수 있다.

[그림 1]은 유전자 알고리즘을 나타내고 있다. [그림 1]에서 모집단은 한 세대를 구성하는 개체들의 집합을 나타낸다. 단계 (2)는 초기 세대의 모집단을 구성하는 단계로서, 랜덤하게 생성된다. 단계 (3)과 단계 (7)은 초기 세대의 모집단에 대한 각 염색체의 우성과 열성 정도를 판별하기 위한 평가 부분이다. 그리고, 단계 (4)~(8)은 유전자 연산의 반복 처리에 의한 진화과정을 나타내는 부분으로서, 자연 개체에서의 교배에 해당되는 교차연산과 돌연변이연산이라는 유전자 연산이 이루어진다. 교차연산은 선택된 두 염색체 사이에 조작이 이루어지며 이 과정에서 실제 염색체 교배가 이루어진다. 돌연변이연산은 선택된 하나의 염색체에 대해 조작이 이루어지는 과정으로서, 교차연산에 의해 생성할 수 없는 상태를 만들기 위한 연산 과정이다. 단계 (5)는 현 세대의 모집단에서 다음 세대로 진화할 염색체들을 선택하여 새로운 모집단을 구성하는 과정으로서, 상대적으로 우성인 개체들이 선택될 가능성이 높도록 처리된다.

```
(1) Begin
(2)    Initialize population;
(3)    Evaluate population;
(4)    While not maxgeneration
(5)        Select chromosome for next population;
(6)        Crossover and Mutation;
(7)        Evaluate population;
(8)    end while
(9) end
```

그림 1. 유전 알고리즘
Fig 1. Genetic Algorithm

2.2 유전자 알고리즘 구성요소

가. 교차연산자(Crossover)

교차연산은 실제로 선택된 스트링 한 쌍에 대한 실질적인 진화 조작이다. 교차연산에는 크게 단일점 교차와 이점교차 방법이 사용된다. 다음은 일반적인 단일점 교차와 이점교차 방법을 나타내고 있다.

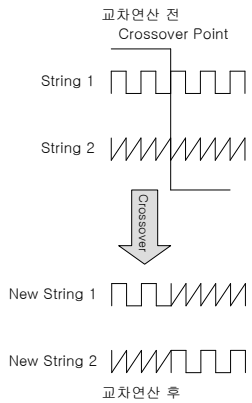

그림 2. 단일점 교차연산
Fig 2. One-Point Crossover

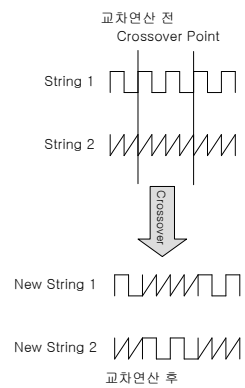

그림 3. 이점 교차연산
Fig 3. Two-Point Crossover

[그림 3]과 [그림 4]의 방법은 염색체의 구성이 0과 1로 이루어진 경우를 나타내고 있다. 그러나 TSP와 같은 노드 중심의 문제에서는 염색체의 구성이 0과 1로 이루어진 이진염색체로 구성할 수 없다. 왜냐하면 노드 중심의 문제에서는 각 염색체의 요소들이 중복해서는 안 되기 때문이다. 따라서 노드 중심의 염색체는 그래프의 노드들로 구성된다. 예를 들어, 5개의 노드로 구성된 TSP 문제에서는 5개의 문자열로 구성되며, 각 염색체의 구성요소의 순서는 그 노드를 방문하는 순서가 된다. 이와 같은 노드 중심의 문제에서는 주로 PMX (Partially Matched eXchange)방법 또는 OX (Ordered eXchange) 방법을 사용한다. 다음은 PMX 방법을 설명하고 있다.

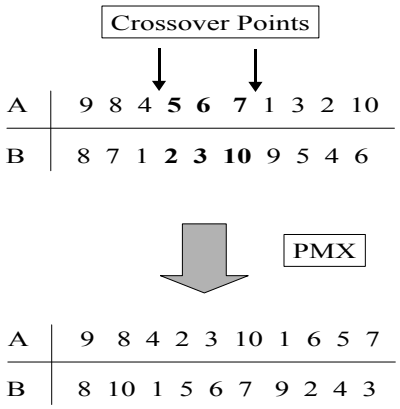

그림 4. PMX 수행의 예
Fig 4. An Example of PMX

나. 돌연변이연산자(Mutation)

돌연변이연산은 교차연산과는 달리 2개의 스트링을 이용해서 새로운 2개의 스트링을 만드는 것이 아니라 하나의 스트링을 부분적으로 조작하여 새로운 스트링을 만들어 내는 조작이다. 즉, 돌연변이연산은 교차연산이 실질적인 진화 동작을 하는 것과는 달리 하나의 스트링 상에서 변화를 줌으로써 교차연산시 지나칠 수도 있는 상태를 만들어 내는 연산자이다.

돌연변이연산 방법에는 교환방법(Exchange Method), 반전방법(Inversion Method) 등이 있다. 교환 방법은 선택된 염색체의 스트링 중 두개의 요소의 위치를 바꾸는 것이고, 반전 방법은 선택된 염색체의 스트링 중 두 지점을 택해 두 지점 사이의 스트링을 반전하는 것이다. 다음은 교환 방법과 반전 방법을 설명하고 있다.

A = 1 **0** 1 **1** 0 1

A' = 1 **1** 1 **0** 0 1

A = 1 0 | 1 1 0 | 1

A' = 1 0 **0** **1** **1** 1

다. 적합함수(Fitness Function)

유전자 알고리즘에서는 모집단에 속한 각 염색체들에 대한 평가를 한 후 그 결과를 선택 알고리즘에 적용하여 다음 세대로 진화하는 모집단을 생성하게 된다. 일반적으로 새로운 세대의 모집단을 생성하기 위한 선택 방법으로 Roulette Wheel 방법을 사용한다. 선택 연산을 수행하기 위해서는 각 염색체에 대한 목적함수와 적합함수(Fitness Function)가 필요한데, 목적함수는 문제에 맞게 함수식을 세워야 한다.

다음은 TSP 문제에 사용되는 목적함수와 적합 함수의 예이다.

k번째 염색체의 목적함수 C(k):

$$minimize\ C(k) = \sum_{i=1}^{n-1} d_k(i, i+1)$$

여기서, i는 염색체를 구성하는 스트링의 순서, 즉, 노드 번호를 나타내고, d(i, i+1)은 i 노드와 i+1 노드 사이의 라우팅 비용을 나타낸다.

k번째 염색체의 적합 함수 F(k):

$$F(k) = \left(\frac{1}{C(k)}\right)^m \quad \text{(식 5)}$$

여기서, C(k)는 위에서 설명한 k번째 염색체의 목적함수를 나타내고, m은 스케일링(Scaling)인자로서 0보다 큰 정수이다. 이 스케일링 인자는 우성과 열성의 차이를 크게 해 줌으로써 우성인자를 갖는 염색체의 선택 확률을 높여준다.

다음은 Roulette Wheel 선택 방법에서 사용하기 위한 각 염색체의 선택확률을 계산하기 위한 수식이다.

선택 확률 P(k):

$$P(k) = \frac{F(k)}{\sum_{i=1}^{n} F(i)} \quad \text{(식 6)}$$

여기서, P(k)는 k번째 개체의 상대적인 적합함수 값으로서, 염색체의 상대적인 우성 정도를 나타낸다.

III. 시스템 설계 및 구현

3.1 시스템 구성

본 논문에서 설계한 시스템의 구성은 크게 사용자 인터페이스 부분과 GA 엔진 부분, 그리고 실험 결과로 구성 된다. 사용자 인터페이스는 유전자 알고리즘에 적용되는 파라미터 설정 부분으로, 모집단(population) 크기, 최대 진화 횟수, 교차연산 방법과 비율, 돌연변이연산 방법과 비율 등으로 이루어져 있다. GA 엔진은 사용자 인터페이스에서 정의된 파라미터를 이용하여 유전자 알고리즘이 적용되는 부분이다. 다음은 본 연구에서 설계한 시뮬레이션 시스템의 구성도를 나타내고 있다.

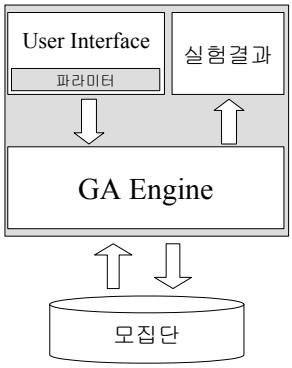

그림 5. 시뮬레이션 시스템 구성
Fig 5. The Structure of Simulation System

3.2 사용자 인터페이스

사용자 인터페이스는 GA 엔진에서 사용될 파라미터를 설정하는 부분이다. 다음은 본 논문에서 구현한 사용자 인터페이스를 나타내고 있다. GA 알고리즘에서 적용되는 파라미터들은 모집단(Population) 크기, 최대 진화횟수(Generation Count), 교차연산(Crossover) 방법 및 비율, 돌연변이(Mutation) 연산 방법 및 비율 등이다.

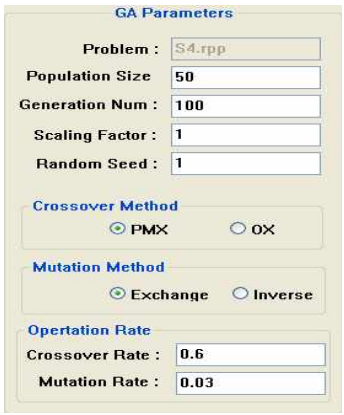

그림 6. 유전 알고리즘 파라미터 설정 인터페이스
Fig. 6. Setting Interface of Parameters for GA

3.3 GA 엔진

GA 엔진은 적용되는 문제에 따라 염색체의 구성이 달라지기 때문에 템플릿을 이용한 객체지향 설계 방법을 사용하였다. [그림 7]은 본 논문에서 설계한 전체적인 GA 엔진의 구성을 나타내고 있다. GA 알고리즘에서 적용되는 연산자들(평가, 선택, 교차연산, 돌연변이연산)은 주어진 염색체의 구성에 따라 달라지므로 본 시뮬레이션 시스템을 사용하는 사용자가 직접 코딩을 하여 적용할 수 있도록 설계하였다.

[그림 7]에서 CHROMOSOME 클래스는 하나의 염색체에 대한 클래스이고, POP 클래스는 CHROMOSOME 클래스를 상속받은 클래스로 하나의 염색체에 대한 목적값과 적합값을 가지고 있다. POPULATION 클래스는 POP 클래스에서 상속된 것으로 전체 모집단에 대한 정보를 가지고 있고, 실제 진화를 위한 멤버 함수들로 구성된다.

그림 7. 시뮬레이션 시스템을 위한 엔진 구성
Fig. 7. Engine Structure for Simulation System

3.4 실험 결과

[그림 8]은 본 논문에서 설계 및 구현한 유전알고리즘 시뮬레이션 엔진을 이용하여 TSP 문제의 경로찾기 해를 구하는 화면이다. 경로찾기에 사용된 문제는 외부의 파일에서 주어지고 파라미터 설정 UI를 통해 유전알고리즘에 적용될 파라미터를 설정하면, GA 엔진을 통해 근사 최적해를 구하여 그 결과와 경로를 사용자에게 보여준다.

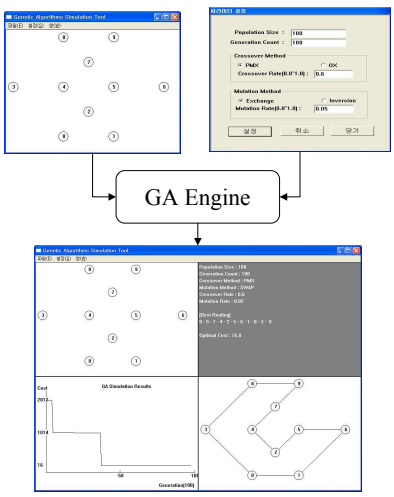

그림 8. 시뮬레이션 실행 화면
Fig 8. The Example of Simuation for TSP

Ⅳ. 결론

본 논문에서는 게임, TSP, RPP, CPP 등에서 사용되는 경로찾기 문제를 유전알고리즘으로 탐색하기 위한 엔진과 시뮬레이션 시스템을 설계하고 구현하였다.

기존 알고리즘들은 그래프의 노드와 에지의 크기가 작은 문제에서는 전역 최소해를 찾을 수 있지만, 문제 크기가 커짐에 따라 해집합이 폭발적으로 커져 최적 경로를 찾기 힘들다.

따라서 본 논문에서는 이러한 문제를 해결하기 위해 유전알고리즘을 이용한 경로찾기 문제 해법을 시뮬레이션할 수 있는 시스템을 설계하고 구현하였다.

시뮬레이션 시스템에는 사용자 인터페이스를 이용하여 유전알고리즘의 파라미터를 설정할 수 있으며, 본 연구에서 구현한 GA 엔진을 통해 근사 최적해를 구할 수 있다. 근사 최적해는 사용자가 분석을 쉽게 할 수 있도록 유전알고리즘의 진화과정과 최적 경로를 그래프로 나타내도록 하였다.

참고문헌

[1] Ron Penton, Data Structures for Game Programmers, Thomson, 2004

[2] Mat Buckland, AI Techniques for Game Programming, Thomson, 2002

[3] 강명주, 무향 Rural Postman Problem 해법을 위한 유전 알고리즘에서 그래프 변환에 의한 디코딩 알고리즘, 한국컴퓨터정보학회논문지, 제12권, 2호, 2007

[4] Goldberg, D. E., Genetic Algorithm in Search, Optimization & Machine Learning, Addison-Wesley, 1989.

[5] Ladd, S. R., Genetic Algorithms in C++, M&T Books, 1995.

[6] Michalewicz Z., Genetic Algorithms + Data Structures = Evolution Programs, Second, Extended Edition, Springer-Verlag, 1994.