Journal of Knowledge Information Technology and Systems

ISSN 1975-7700

http://www.kkits.or.kr

# A Search of the Shortest Route with a Limit on a Number of Transfer using Genetic Algorithm

**Myun_Sub Lee***

*Department of Computer Science and Engineering, Incheon University*

## ABSTRACT

The major role of intelligent character in an artificial intelligence game is to be smarter and to add more interest on character behaviors. Moreover, to apply artificial intelligence on behaviors and movements of intelligent characters, the time to process artificial intelligence should be shortened because even without the process of artificial intelligence, it takes time to visualize and print out a character with intelligence. In this study, the method that searching a spanning tree with the lowest cost which is limited in a number of transfer by a using genetic algorithm is suggested. In genetic algorithm, a roulette selection was chosen to be the selection method, and hybridization method of uniform crossover was applied. The elite conservation strategy for throwing back the optimal parameter to the following generation was applied to shorten the searching time. The Dandelion Code, an effective method to encode a tree in the evolution algorithm, was used. As a result of experiment, when the definite number was 50, 60, 70, and 80, the number of transfer limitation was 3 and 5 to compare the cost and time of spanning tree. When the definite number was 80 and the number of transfer was 5, the route was searched 0.161 seconds earlier.



**KEYWORDS:** Artificial intelligence, Action game, Keypads, Intelligent character, Genetic algorithm, Spanning tree

**ARTICLE INFO:** Received 14 February 2017, Revised 7 March 2017, Accepted 7 April 2017.

*Corresponding author is with the Department of Computer Science & Engineering, Incheon University, (songdo-dong) 119 Academy_ro, Yeonsu-gu, Incheon, KOREA.

*E-mail address*: nantian@inu.ac.kr

## 1. 서 론

실제 게임에서 임의의 한 지점에서 다른 목표 지점까지 이동할 경우에 최소 비용만을 고려하면 모든 캐릭터는 최소 비용을 갖는 하나의 경로만으로 이동하게 된다. 따라서 게임에서는 캐릭터들이 다양한 경로로 이동할 수 있으면서도 가능한 빨리 이동할 수 있는 경로를 선택 할 필요가 있다.

본 연구에서는 최소 비용 스패닝 트리에 경유횟수 제한을 두었다. 출발지에서 목적지로 향하는 경로를 탐색하되 경유하는 횟수를 제한하여 하나의 경로에 너무 많은 정점들이 연결되지 못하도록 하였다. 경유횟수 제약이란 일정 횟수 이상은 정점 접속을 할 수 없다는 제약이다. 경유횟수를 제한하는 최단 경로는 비용이 있는 그래프에서 경유 횟수 제약을 만족하며, 스패닝 트리의 변 비용이 최소가 되는 트리이다. 경유횟수를 제약하는 최소 비용트리의 탐색 방법으로 근사해법의 하나인 유전자 알고리즘에 Dandelion 코드를 추가하여 스패닝 트리에 부호화 방법을 이용하였다. 염색체를 Dandelion 스트링으로 표현하고 Dandelion 코드의 디코딩 알고리즘을 이용하여 스패닝 트리로 디코드 하였다. 디코드 한 최소비용 트리로부터 각 염색체의 적합도를 구하고 선택된 개체에 유전자의 선택과 교배, 돌연변이 조작을 하여 가장 좋은 해를 탐색하도록 하였다.

본 논문의 구성은 다음과 같다. 2장에서는 경유횟수에 제약이 있는 최소 비용트리 대해 구체적인 예를 들어 설명하고, 3장에서는 유전자 알고리즘에 대해 간단히 소개한다. 4장에서는 Dandeloin Code에 대해 설명과 그 디코딩 방법 및 엔코딩 알고리즘에 대해 설명한다. 5장에서는 실험 결과에 대해 설명하고 비교 분석한다. 6장에서는 본 연구를 결론에 대하여 설명한다.

## 2. 경유횟수 제약이 있는 최소 비용트리

정점 수 N인 연결 그래프 G에서 어떤 제약 조건을 두어 최소 비용트리를 구하는 문제를 제약이 있는 스패닝 트리라 한다[1,2]. 본 연구에서는 스패닝 트리의 정점을 일정 횟수 이상은 경유하지 못하도록 하는 제약 조건을 추가하였다. 정점에 a에서 n까지 번호를 붙여 N번째의 정점을 마스터 허브, 그 외의 정점을 로컬 허브라 부른다. 최소 비용트리가 있을 경우 각 로컬 허브와 마스터 허브의 연결은 오로지 하나 뿐이고 정점을 통하여 로컬 허브로 연결되는 변의 수를 경유횟수로 하였다. 본 연구에서는 모든 로컬 허브의 제한된 경유횟수를 만족하는 최소 비용트리에서 비용이 최소가 되는 경로를 구하는 것이 목적이다.

<그림 1>은 비용을 가지는 연결 그래프이다. 정점 h가 마스터 허브이고, 그 이외의 정점 a에서 g까지는 로컬 허브이다. 경유횟수 제약이 없는 경우 G의 최소 비용트리는 <그림 2>가 되고, 비용은 47이 된다. 이 때 경유횟수를 2로 제한하면 <그림 2>의 트리에서는 마스터 허브 h로부터 b와 d가 경유횟수는 3이 되므로 경유 횟수 제약을 만족하지 못한다. 마스터 허브로부터 로컬 허브까지의 각각 비용은 h-b의 비용은 23, h-d의 비용은 19이다. 게임에서는 한 번에 이동할 수 있는 거리(비용)가 짧아야 신속하게 이동할 수 있다. <그림 3>은 비용이 53인 최소 비용트리이다. 이 경우 정점을 경유하는 횟수를 2로 제한한 최소 비용트리이다. 이 때 h-b의 비용은 15이고, h-d의 비용은 17이다. 이렇게 경유횟수를 제한하게 되면 총 비용은 증가하지만 로컬 경로의 비용은 감소된다.

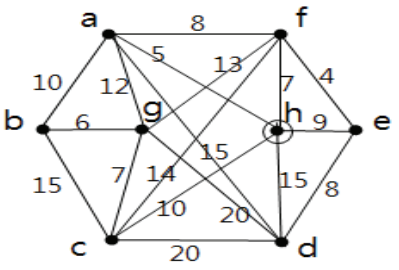


그림 1. 연결 그래프
Figure 1. Connecting graph

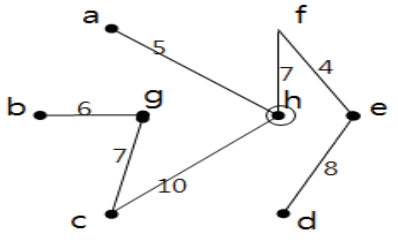


그림2. 스패닝 트리
Figure 2. Spanning tree

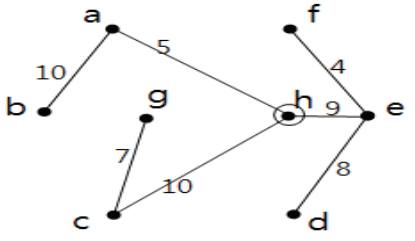


그림3. 제약이 있는 스패닝 트리
Figure 3. Constrained spanning tree

## 3. 유전자 알고리즘

유전자 알고리즘(G.A.)[3,4,5]은 자연도태의 원리를 이용한 것으로 자연 생태계의 염색체를 교배와 돌연변이를 반복하여 세대가 거듭될수록 자연 환경에 적응하는 개체는 다음세대로 전달되고, 적응하지 못하는 개체는 도태되는 진화 알고리즘이다. 매 세대마다 주어진 문제를 해결하기 위해 염색체의 적합도를 계산하고, 계산된 적합도에 따라 개체를 선택한 후 두 개체를 교차하고, 해가 한 방향으로 수렴하는 것을 방지하기 위하여 돌연변이 조작을 반복하여 좋은 해를 탐색해 나가는 과정이다.

유전자 알고리즘에 필요한 파라미터로는 개체수 Npop, 최대 세대수 Gmax, 교배 확률 Xrate, 돌연변이 확률 Mrate 이다. 본 연구에서 유전자 알고리즘의 적용은 다음과 같다. 초기 염색체로 유전자가 되는 데이터를 입력하고, Npop개의 초기 염색체 집단을 생성한다. 유전자는 정점의 번호로서 1부터 n까지의 랜덤하며 이것을 하나의 염색체라 한다[6,7].

경유횟수 제약 조건이 있는 최소비용 스패닝 트리에서는 스패닝 트리를 구성하기 위한 비용이 적을수록 적합도가 높게 된다. 먼저 선택조작을 하기 전에 모든 염색체에 대해서 각 스패닝 트리의 비용을 구한다. 이 때 경유횟수 제약을 만족하지 않는 염색체의 비용은 1로 한다. 모든 염색체 중에서 적합도가 가장 나쁜 염색체를 탐색하여 스패닝 트리의 비용을 $C_{worst}$로 한다. 하나의 염색체 x의 비용을 Cost(x)로 하면 적합도 Val(x)는

$$Val(x) = C_{worst} - C_{ost}(x) + 2 : \text{if } C_{ost}(x) \neq 1$$
$$Val(x) = 1 ; \text{if } C_{ost}(x) = 1$$

이 된다. 즉, 가장 나쁜 스패닝 트리의 비용에서 각각의 비용을 뺀 후 2를 더한 것이 적합도이다. 따라서 비용이 작은 것일수록 적합도가 높으며, 제약을 만족하지 않는 염색체는 적합도가 매우 낮게 된다.

선택은 다음 세대에 남을 염색체를 선택하는 것

으로 적합도가 높으면 높을수록 선택될 확률이 높다. 교배는 2개의 염색체를 부분적으로 서로 바꿈으로써 새로운 개체를 생성하는 것이다[8]. 이 때 부모의 형질이 자손에게 적절히 계승되어야 하며 교배 확률에 따라 한 쌍의 염색체로부터 다음 세대의 새로운 염색체를 만들어 내는 유전자 조작이다. 생성 후에는 부모 염색체는 없어지고 부모 염색체로부터 일부의 성질을 물려받은 새로운 자식 염색체가 만들어지므로 부모 염색체의 적합도 보다 더 좋은 염색체가 만들어 질수 있다. 본 연구에서는 룰렛 선택과 균등 교배를 각각 적용하였다[9,10].

돌연변이는 돌연변이 확률에 따라 염색체의 어느 한 유전자를 아주 다른 성질의 유전자로 조작을 가하여 변이를 만들어 내는 과정이다[11,12]. 이 조작에 의해 적합도가 낮게 되는 경우도 있지만 국소해에 빠지는 것을 방지할 수가 있는 효과가 더 크다.

## 4. Dandelion Code

정점수가 N인 완전 그래프의 최소 스패닝 트리는 $N^{N-2}$개 이다. 그러므로 1에서 N까지 정수(중복 허용)를 Dandelion 스트링 이라고 부른다[13]. 임의의 Dandelion 스트링 ∅를 정의역[2, N-1], 치역이 [1, N]의 함수라고 하자. 2 ≤i ≤ N-1 이고, i에 대해 ∅(i)=$d_i$ 일 때 이와 같은 Dandelion 스트링을 ∅=($d_2$, $d_3$, … ,$d_{N-1}$) 라고 표현하고, Dandelion 스트링은 모두 $N^{N-2}$개 이다.

한편 최소 스패닝 트리에서 정점 i (i ≠ N)부터 정점 N으로의 경로는 오직 하나뿐이므로 그 경로의 정점 i의 다음에 오는 정점은 바로 결정된다. 그 정점을 succ(i)라고 표시하고, succ는 완전 그래프의 하나인 최소 스패닝 트리로 표시한다. <그림 4>에서 정점 1부터 N=7로 향하는 경로는 우선 정점2를 통과한다. 따라서 succ(1)=2이다. 같은 방법으로 succ(2)=5, succ(3)=6, …, succ(6)=4가 된다. 이것을 표로 표시하면 <표 1>과 같다.

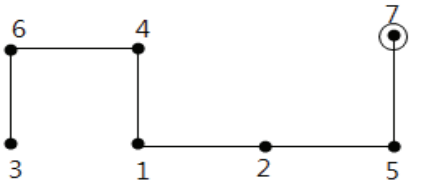


그림4. 트리 T
Figure 4. Tree T

표1. 트리 T의 succ.
Table 1. succ. of tree T

| 정점번호 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| succ | 2 | 5 | 6 | 1 | 7 | 4 |

위와 같은 함수 succ는 모두 $N^{N-1}$ 개이므로 정의역이 [1, N-1]이고, 치역이 [1, N] 의 함수로 되었다 하더라도 스패닝 트리에 대응하지 않는 것이 존재한다. 그래서 임의의 Dandelion 스트링에 대응하는 스패닝 트리를 작성하는 복호화 알고리즘과 반대로 임의의 스패닝 트리로부터 Dandelion 스트링를 작성하는 부호화 알고리즘을 디코딩하는 방법이 Dandelion Code[14,15]이다. 다음은 디코딩에 대한 설명이다. Dandelion 스트링 ∅=($d_2$, $d_3$, … ,$d_{N-1}$)이 주어졌을 때 이 Dandelion 스트링에 대응하는 스패닝 트리는 다음 알고리즘으로 구한다.

■ 스텝 1. Dandelion 스트링에 포함하는 사이클을 $Z_1$, $Z_2$, … , $Z_t$라 하고, 각 사이클에 포함되는 요소 $Z_i$의 최소값을 $b_i$라 한다. 각 사이클 Zi의 최소값 $b_i$는 $Z_i$의 우측에 놓고, 또, i < j 이라면 bi < bj이라고 한다.

■ 스텝 2. 사이클을 $Z_1$, $Z_2$, … , $Z_t$ 에 포함되는 요소를 각 사이클 내의 순서 $Z_1$부터 Zt의 순서

의 리스트를 Π라고 한다.

■ 스텝 3. 최소 스패닝 트리 T에 포함되는 변은 2종류로 다음 (a) 와 (b)로 구성된다.

(a) 리스트 Π에 포함되는 요소가 그 순서가 되고, 정점 1부터 정점 N으로의 경로의 통과정점이 되도록 한다. 즉, succ(1) = Π(1), succ(Π(1)) = Π(2), …, succ(Π(p-1)) = Π(p), succ(Π(p)) = N으로 한다. 단, Π(j)는 리스트 Π의 j 번째 요소이고, p는 리스트 Π에 포함되는 요소수이다.

(b) 리스트 Π에 포함되지 않는 요소 i에 대해서는 변($i_i$,$d_i$)를 트리 T에 포함시킨다. 즉, succ(i)=$d_i$로 한다.

예를 들어 <표 2>와 같이 2부터 12까지의 정점 N과, Dandelion 스트링 ∅가 주어졌을 때 단계별로 설명하면 다음과 같다.

표2. Dandelion 스트링
Table 2. Dandelion string

| N | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| D.S ∅ | 4 | 11 | 2 | 5 | 3 | 8 | 10 | 4 | 1 | 6 | 12 |

■ 스텝 1 : Dandelion 스트링 ∅에서 포함되는 리스트로부터 순서대로 $Z_1$=(2, 4), $Z_2$=(3, 11, 6), $Z_3$=(5), $Z_4$=(12)이다. 여기서 $Z_1$은 두 개의 사이클이며, $Z_2$는 세 개 사이클, $Z_3$와 $Z_4$는 하나의 사이클로 이루어진다. 이 때 b1=2, b2=3, b3=5, b4=12가 되므로 사이클의 순서는 이대로지만 $Z_1$과 $Z_2$ 중에서 정점 순서는 $Z_1$=(4, 2), $Z_2$=(11 ,6, 3)으로 바뀌게 된다.

■ 스텝 2. $Z_1$ 부터 나란하게 정렬하면 Π = (4,2,11,6,3,5,12)로 된다.

스텝3. (a) Π에 포함되는 요소로부터 succ를 구하면 <표 3>의 스텝(a)가 얻어진다.

■ 스텝 3. (b) 남은 변은 표2의 Dandelion 스트링을 이용하여 구하면 표3의 스텝(b)와 같이 된다. 이와 같이 얻어진 succ의해 <그림 5>와 같은 트리 T로 나타낼 수 있다. <표 3>의 스텝3(b) 순서를 정리하면 1-4-2-11-6-3-5-12-13 이다. 나머지 정점 7,8,9,10은 사이클이 없으므로 차례대로 연결해 주면 된다. 이 순서대로 그래프를 그리면 <그림 5>와 같다.

다음은 엔코딩에 대한 설명이다. 정점 수 N의 스패닝 트리 T가 있을 때, T에 대응하는 Dandelion 스트링 ∅=($d_2$, $d_3$, … ,$d_{N-1}$)는 다음 알고리즘으로 구할 수가 있다. 알고리즘 중에 있는 최소값이라고 하는 것은 리스트 중에서 우측에 있는 모든 값보다 작은 값을 의미한다.

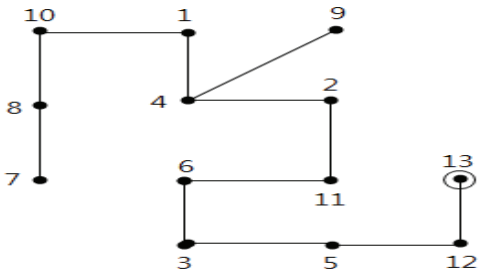


그림5. Dandelion 스트링 트리 T
Figure 5. tree T of Dandelion String

표3. Dandelion 스트링에 대응하는 succ.
Table 3. succ responds to Dandelion string.

| 정점번호 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 스텝3(a) | 4 | 11 | 5 | 2 | 12 | 3 | | | | | 6 | 13 |
| 스텝3(b) | 4 | 11 | 5 | 2 | 12 | 3 | 8 | 10 | 4 | 1 | 6 | 13 |

■ 스텝 1. 스패닝 트리에 있는 정점 1부터 정점 N으로의 통과하는 정점을 통과하는 방향의 순서대로 정렬한 리스트 Π를 구한다.

■ 스텝 2. Π의 요소를 좌측부터 조사하여 우측 최소값으로 끝나는 몇 개의 사이클 $Z_1$, $Z_2$, … , Zn 에 할당한다.

■ 스텝 3. T에 대응하는 Dandelion 스트링 ∅ =($d_2$, $d_3$, … ,$d_{N-1}$)의 각 요소 di 는 다음과 같이 구한다.

(a) 각 사이클 Zi에 포함되는 요소는 Dandelion 스트링 에서도 사이클을 구성하도록 한다. 즉, Zi=($z_1$, $z_2$, … , $z_q$) 라면 $dz_1 = z_2$, $dz_2 = z_3$, , $dz_q=z_1$ 이다.

(b) 리스트 Π에 포함되지 않은 요소 i (2 ≤ i ≤ N-1)에 대해서는, di = succ(i)로 한다. 예를 들어 <그림 6>과 같은 트리 T 가 있을 때 Dandelion 스트링은 다음과 같이 구할 수 있다.

엔코딩 알고리즘의 스텝 1에서 정점 1부터 정점 24까지 단일 경로를 결정한다. 먼저 루트 1부터 정점 25까지 조사하여 매 정점(i∈[1,24])마다 succ(i)를 결정한다. 여기서 succ(i)는 1부터 정점 25로 향하는 첫 번째 정점이다. 이렇게 구한 succ(i)는 <표 4>와 같다.

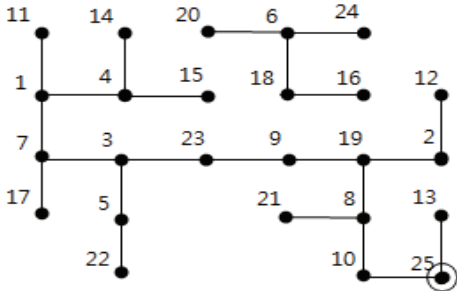


그림6. 트리 T
Figure 6. Tree T

표4. 트리 T의 succ(i)
Table 4. succ(i) of tree T

| N(정점) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| succ(i) | 7 | 19 | 23 | 1 | 3 | 18 | 3 | 10 |
| N(정점) | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| succ(i) | 9 | 25 | 1 | 2 | 25 | 4 | 4 | 18 |
| N(정점) | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| succ(i) | 7 | 9 | 8 | 6 | 8 | 5 | 9 | 6 |

<표 4>로부터 경로 Π를 구하면 경로 W=(1,7,3,23,9,19,8,10,25) 이고, Π=(7,3,23,9, 19,8,10) 이다. 스텝2에서 리스트 Π에서 사이클 우측 값부터 최소값(10,8,3)으로 구분하면 $Z_1$=(3,7), $Z_2$=(8,23,9,19), $Z_3$=(10)이 된다. b1=3, b2=8, b3=10이 되므로 사이클의 순서는 이대로지만 $Z_1$과 $Z_2$ 중에서 정점 순서는 $Z_1$=(7,3), $Z_2$=(23,9,19,8)으로 바뀌게 된다. 스텝 3의 (a)단계로서 Z(i) 로부터 Dandelion 스트링을 구하면 <표 5>의 (a)가되고, <표 5>에서 빈 칸은 표4의 i번째 값으로 채우면 <표 5>의 (b)가 되고 완성된 Dandelion 스트링은 (c)와 같다.

표5. 완성된 Dandelion 스트링
Table 5. Completed Dandelion string

| N(정점) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| (a) succ(i) | | | 7 | | | | 3 | 23 |
| (b) succ(i) | 7 | 19 | | 1 | 3 | 18 | | |
| **(c) succ(i)** | **19** | **19** | **7** | **1** | **3** | **18** | **3** | **23** |
| N(정점) | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| (a) succ(i) | 19 | 10 | | | | | | |
| (b) succ(i) | | | 1 | 2 | 25 | 4 | 4 | 18 |
| **(c) succ(i)** | **19** | **10** | **1** | **2** | **25** | **4** | **4** | **18** |
| N(정점) | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| (a) succ(i) | | | 8 | | | | 9 | |
| (b) succ(i) | 7 | 9 | | 6 | 8 | 5 | | 6 |
| **(c) succ(i)** | **7** | **9** | **8** | **6** | **8** | **5** | **9** | **6** |

## 5. 결과 및 검토

유전자 알고리즘은 확률에 기반을 둔 연산이기 때문에 우성 유전자의 특성들이 반드시 다음 세대로 유전되지는 않는다는 문제가 존재한다. 따라서 현재 세대의 엘리트 유전자를 다음 세대로 유전되도록 하는 엘리트 전략을 적용하였다. 엘리트 보존 전략을 적용한 이유는 매 세대에서 최고의 적합도를 갖는 부모 염색체를 다음 세대로 유전되도록 함으로서 매 세대마다 적합도를 정렬해야 하는 시간을 단축할 수 있으며, 적합도가 국소해에 빠지는 것을 방지할 수가 있기 때문이다.

실험 데이터로는 널리 사용되고 있는 Waxman의 임의 그래프 생성 기법을 사용하였다. 생성된 그래프는 정점수 50, 비용(랜덤값)은 94로 축소하여 적용한 결과 <표 6>과 같으며, 비용과 시간을 비교하였다. <표 6>에서 A는 경유 제약횟수를 적용하지 않은 최소비용 스패닝 트리의 비용과 시간이고, B는 제약횟수를 3일 때 비용과 시간이다. 제약횟수가 있는 경우에 비용이 큰 것을 확인하였으며 다른 실험에서도 같은 결과를 보였다. 실제 게임에서 인공지능을 적용하기가 어려운 점이 인공지능을 이용한 문제해결에 시간이 걸린다는 점이다. 인공지능 알고리즘으로는 구현이 가능하나 게임에 적용할 때는 시간지연과 시각화로 온라인 게임에 적용이 어렵다. 이런 점에서 볼 때 <표 6>의 B에서 시간 0.081은 A방법에 비해 시간이 더 걸리기는 하나 온라인 게임에서 인공지능 캐릭터의 행동에 큰 영향을 주지 않는다고 생각된다.

표6. 비용과 계산시간의 비교
Table 6. Comparison between cost and time

| 정점수 | 비용 | | 시간 | |
|---|---|---|---|---|
| | A | B | A | B |
| 50 | 285 | 413 | 0.016 | 0.081 |

<표 7>은 정점수가 60, 70, 80이고 각각에 대해서 횟수제약을 3과 5로 나누어 적용했을 때의 비용과 시간을 측정하였다. 횟수 제약이 많으면 경로를 탐색할 때 조건에 맞는 경로를 여러 번 탐색해야 하는 문제점으로 시간이 늘어남을 알 수 있다. 횟수제약을 적용하면 <표 7>에서와 같이 비용과 시간이 증가하였다. 그러나 온라인 게임에 인공지능을 적용하고, 또 한 지점에서 여러 지점으로 게임 캐릭터를 이동시킬 때 하나의 경로만 이용하지 않고 여러 경로를 통하여 이동하게 되므로 게임의 몰입도나 흥미를 유발할 수가 있다. <표 7>의 시간은 게임 전체 화면에서 경로를 탐색할 때 소요되는 시간이므로 실제 구현과정에서 캐릭터를 이동시킬 때는 다소 차이가 있을 수 있다. 여기서 비용은 랜덤 생성된 값으로 정점수 80을 먼저 실행한 후 정점을 10개씩 제거하면서 실험하였다.

표7. 경유 횟수제약 결과
Table7. Result of applying via limitation.

| 정점수 | 횟수 제약 | 비용 | 시간 [sec] |
|---|---|---|---|
| 60 | 3 | 529 | 0.097 |
| | 5 | 618 | 0.106 |
| 70 | 3 | 641 | 0.115 |
| | 5 | 765 | 0.130 |
| 80 | 3 | 784 | 0.132 |
| | 5 | 972 | 0.161 |

## 6. 결 론

현재 인공지능에서 게임 캐릭터(NPC)의 주된 역할은 캐릭터의 행동을 보다 영리하고 흥미를 더해주는 기능 위주로 구현되고 있다. 지금까지 주로 게임에 적용하고 있는 인공지능 기술은 실제 인공지능 분야에서 개발된 기술과는 거리가 멀고, 대개 스크립트와 AI 길찾기 등의 한정된 기술에만 의존

해왔다. 물론 이들 기술만 가지고도 충분히 영리해 보이는 캐릭터를 만들 수 있다. 하지만 인공지능에 지능 캐릭터를 적용하였을 때 인공지능 처리에 걸리는 시간과 결과를 캐릭터에 시각화하여 출력하는데 시간이 소요된다는 점이다.

본 연구에서는 유전자 알고리즘을 이용하여 경유횟수 제약이 있는 최소비용 스패닝 트리를 탐색하는 방법을 제안하였다. 유전자 알고리즘에서는 룰렛 선택 방식을 사용하였고, 교배는 균등교배를 적용하였다. 엘리트 전략도 이용하여 최적의 파라미터가 다음 세대에 유전되도록 하여 탐색시간을 단축하였다. 엔코딩과 디코딩 과정에서 Dandelion Code를 이용하여 염색체로의 이식이 용이하도록 하였다.

실험 결과 정점수가 50, 60, 70, 80 일 때 경유횟수 제약을 3과 5로 하여 스패닝 트리의 비용과 시간을 비교하였다. 정점수와 경유횟수가 가장 큰 (정점수 80, 경유횟수 5) 경유 경로 탐색에 0.161초가 소요되었다. 인공지능이 처리하는 시간이 짧으면 짧을수록 게임 캐릭터의 이동을 효과적으로 시각화 할 수가 있다. 실제 게임에 적용할 경우 정점수와 경유 횟수 제약을 얼마로 할지 정확한 기준은 없으나 게임 화면 크기나 이동할 캐릭터의 수에 따라 적절하게 조절할 필요가 있다고 생각된다.

## References


[1] B. Y. Wu, and K-M. Chao, *Spanning trees and optimization problems,* London, U.K.: Chapman & Hall, 2004.

[2] F. Rothlauf, and D. E. Goldberg, *Pruefernumbers and genetic algorithms: A lesson how the low locality of an encoding can harm the performance of GAs*, in Lecture Notes in Computer Science, 1917, Proc. PPSN VI, Paris, France, pp. 395-404, Sep. 2000.

[3] S. M. Elsayed, R. A. Sarker, and D. L. Essam. *A new genetic algorithm for solving optimization problems*, Eng. Appl. Artif. Intell, Vol. 27, pp. 57-69, 2014.

[4] M. Camilleri, and F. Neri. *Parameter optimization in decision tree learning by using simple genetic algorithms*, WSEAS Transactions on Computers, Vol. 13, pp. 582-591, 2014.

[5] K. L. Sadowski, P. A. N. Bosman, *On the usefulness of linkage processing for solving MAX-SAT,* Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, ACM: 853-860, 2013.

[6] E. B. Thompson, *The application of evolutionary algorithms to spanning tree Problems,* Ph.D. University of Exeter, U. K.. 2003.

[7] D. Kniazew, *A competent genetic algorithm for solving permutation and scheduling problems*, Kluwer Academic Publishers, 2002.

[8] J. E. Laird, *Using a computer game to develop advanced AI*, 2001 IEEE Computer, Jul. 2001.

[9] S. Woodcock, *Game AI : The state of the industry,* Gamasutra Magazine, Vol. 01, Nov. 2000.

[10] M. van Lent, and J. Laird, *Developing an artificial intelligence engine,* Proc. of the Game Developers Conference, pp. 577-588, 1999.

[11] D. Carmel, and S. Markovitch, *Learning models of opponent's strategy in game playing,* CIS Report #9318, 1993.

[12] D. Whitley, *An overview of evolutionary algorithms,* Journal of Information and

Software Technology Vol. 43, pp. 817-831, 2001.

[13] E. Thompson, T. Paulden, and D. K. Smith, *From the dandelion code to the rainbow code; A class of bijective spanning tree representations with linear Complexity and bounded locality,* IEEE Trans. on Evolutionary Computation, Vol. 10, No. 2, pp. 108-123, Apr. 2006.

[14] E. Thompson, T. Paulden, and D. K. Smith, *The dandelion code: a new coding of spanning trees for genetic algorithms,* IEEE Trans. Vol. Comput., Vol. 11, pp. 91-100. 2007.

[15] M. Imase, and B. M. Waxman, *Dynamic steiner tree problem,* SIAM Journal of Discreate Mathematics, Vol. 4, pp. 369-384, Aug.1991.

## 유전자 알고리즘을 이용한 경유횟수 제약이 있는 최단 경로 탐색

**이면섭**

*인천대학교 컴퓨터공학부*

**요 약**

현재 인공지능 게임에서 지능 캐릭터의 주된 역할은 캐릭터의 행동을 보다 영리하고 흥미를 더해주는 기능 위주로 구현되고 있다. 또, 지능 캐릭터의 행동과 이동에 인공지능을 적용하려면 인공지능 처리에 시간을 단축하여야 한다. 이러한 이유는 인공지능 처리 외에 지능이 적용된 캐릭터를 시각화하여 출력하는데 시간이 소요되기 때문이다. 본 연구에서는 유전자 알고리즘을 이용하여 경유횟수 제약이 있는 최소 비용 스패닝 트리를 탐색하는 방법을 제안하였다. 유전자 알고리즘에서 선택 방법은 룰렛 선택을 사용하였고, 교배는 균등교배를 적용하였다. 최적의 파라미터가 다음 세대에 유전되도록 엘리트 조존 전략을 적용하여 탐색시간을 단축하였다. 진화 알고리즘에서 트리를 인코딩하는데 효과적인 방법으로 널리 사용되고 있는 Dandelion Code를 사용하였다 실험 결과 정점수가 50, 60, 70, 80일 때 경유 횟수 제약을 3과 5로 하여 스패닝 트리의 비용과 시간을 비교하였다. 정점수가 80, 경유횟수가 5일 때 0.161초가 빠른 시간내에 경로를 탐색하였다.

## 감사의 글

본 논문은 인천대대학교의 2016학년도 학술연구조성비를 지원 받음.

**Myun Sub Lee** received the B.S. degree in Electrical Engineering from Kookmin Univ. in 1985. He received the M.S. degree in the Electrical Engineering from Inha University in 1987 and the Ph.D. degree in the Department of Computer Engineering from Kookmin University in 2005. He was a professor in Dept. of Computer Science and Engineering at Incheon University from 1990 to 2014. His current research interests include artificial intelligence, intelligent Games.

*E-mail address*: nantian@inu.ac.kr