유전 알고리즘을 이용한 최적 경로 탐색

김경남 [1] · 조민석 [2] · 이현경 [3]
한국과학기술원 해양시스템공학과 [1], 한국과학기술원 기계공학과 [2,] 한국기술교육대학교 메카트로닉스 공학부 [3]

Searching optimal path using genetic algorithm

Kyungnam Kim[1], Minseok cho[2†], and Hyunkyung Lee[3]
[1] Dept. of Ocean systems Engineering, KAIST,
[2] Dept. of Mechanical Engineering, KAIST, [3] Dept. of Mechatronics, Koreatech University

ABSTRACT:

In case of the big city, choosing the adequate root of which we can reach the destination can affect the driver's condition and driving time. so it is quite important to find the optimal routes for arriving the destination as considering the factors, such as driving conditions or travel time and so on.

In this paper, we develop route choice model with considering driving conditions and travel time, and it can search the optimal path which make drivers reduce their fatigues using genetic algorithm.

Key Words: Genetic algorithm, Optimal path

1. 서 론

최적경로 문제는 특정 목적지에 효율적으로 도달하는데 가장 기본적이고 중요한 문제 중의 하나이며, 운전자의 정신적, 신체적 부담을 줄이는데 필수적이다.

최적 경로는 정해진 기점과 종점을 연결하는 다수의 경로들 중 통행자의 목적을 최적으로 하는 경로이다. 일반적으로 최적경로를 탐색하는 기준에는 통행시간, 통행비용 등 하나의 속성만을 고려하며, 통행자는 자신의 목적에 맞는 경로를 최적경로로 선택하여 이동한다. 그러나 실제로 통행자가 단일 속성만을 고려하여 경로를 선택하는 경우, 통행자의 인식범위를 넘어선 도로불량, 차량사고 등과 같은 요인들이 통행시간 및 운전자의 피로도에 영향을 미치게 된다. 그러므로 최적경로를 탐색하기 위해서는 통행에 영향을 미치는 속성을 최대한 파악하여야 하며, 경로탐색 과정에서 이들 속성이 고려되어야 한다.

기존의 최단경로 탐색 알고리즘으로는 데이크스트라 알고리즘, Moore 알고리즘 등 다양한 기법들이 개발되어있다.

본 논문에서는 16 개의 블록이 4x4 형태로 배치되어 있고, 이 블록들을 연결하는 40 개의 도로들의 통행 시간과 도로 상태 및 도로 특성을 고려하여 경로선택모형을 만들었고, 이를 기반으로 유전 알고리즘을 이용하여 운전자의 부담을 최소화할 수 있는 최적경로를 탐색한다.

2. 최적 경로 탐색

2.1 경로 탐색 모형 구축

경로 선택에 영향을 미치는 요소들은 여러 가지가 있으나, 이들 가운데 대부분은 계량화가 어렵거나 현실적으로 자료수집에 많은 시간이 요구된다. 따라서 이러한 애로사항이 있는 요소

들을 통합하여 하나의 요소로 단순하게 나타내기로 하였다. 본 논문에서는 경로 선택에 영향을 미치는 요소를 통행 시간과 도로 상태 두 가지로 선정하였다.

통행시간은 통행자가 특정 도로를 통과할 때 걸리는 시간을 의미하고, 도로 상태는 통행자가 특정 도로를 통과함으로 인해 받게 되는 피로도의 정도를 나타낸다. 도로 상태에 영향을 주는 것으로는 도로 불량, 도로에서의 차량 사고 여부 등이라고 할 수 있다.

본 논문의 경로 탐색 모형은 16 개의 블록이 4x4 형태로 배치되어 있고, 이 블록들을 연결하는 40 개의 도로로 구성되어 있다.

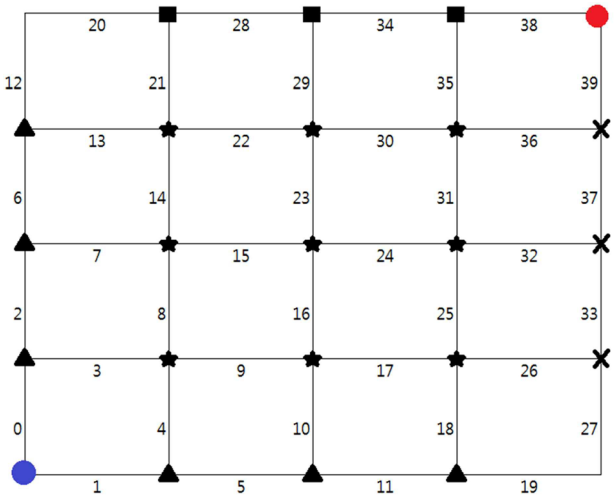


Figure 1 Path search model

도로에 0 부터 39 까지 번호를 매겼고, 번호와 가장 가까운 도로가 해당 번호의 도로이다.

출발지점은 왼쪽 하단의 파란 점의 위치이고, 목적지점은 오른쪽 상단의 빨간 점의 위치이다. 여기서 Road0, Road2, Road6, Road12, Road27, Road33, Road37, Road39 는 ↑방향으로만 통행하고, Road1, Road5, Road11, Road19, Road20, Road28, Road34, Road38 은 →방향으로만 통행하는 것으로 가정하였다. 만약 위에서 명시한 도로들에서 가정한 방향과 반대로 통행을 한다면, 통행자는 아무런 의미 없이 특정 블록을 한바퀴 돌게 되는 것이므로 해당 도로의 통행 방향을 정하였다. 나머지 도로에서는 양방향 통행이 가능하다.

그리고 Road12, Road19 를 통행 시 자동으로 Road20, Road 27 을 각각 통행하는 것으로 가정하였다. 또한 Road35 를 ↑방향으로 통행하거나 Road 36 을 →방향으로 통행 시, 각각 Road 38 과 Road39 를 자동으로 통행하도록 가정하였다.

이전에 서술한 것처럼, 각 도로의 특성을 나타내는 요소들인 통행 시간과 도로 상태는 따로 자료를 수집하기 어렵거나 많은 시간이 요구되므로 임의로 그 값을 정하였다. 그리고 이 두 요소의 관계를 고려하였을 때, 통행시간에 따른 통행자의 피로도는 도로의 상태의 좋고 나쁜 정도에 비례하여 선형적으로 증가한다고 가정하였으므로, 도로의 특성을 통행 시간과 도로 상태를 곱으로 정하였다.

2.2 최적 경로 탐색 알고리즘

최적 경로 탐색을 위해 Binary Coded Genetic Algorithm 을 사용하였다.

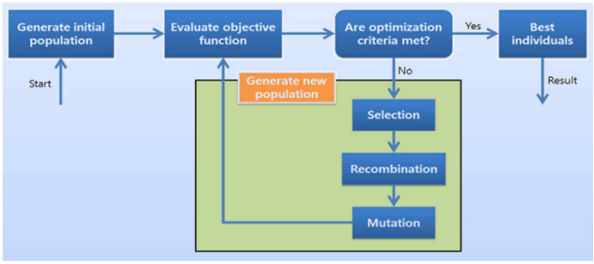


Figure 2 Flow char of genetic algorithm

Figure 2 는 전형적인 genetic algorithm 의 구조이다. Genetic algorithm 은 대부분 정해진 수의 해로 구성되는 해집단을 갖는다. 먼저 설정된 개수만큼의 해를 생성하여 해집단을 만든다. 이 해집단의 해를 objective function 에 대입시켜 여러 해 중에 최적의 해를 선택(selection)한다. 선택된 해를 복수 개의 해를 결합하여 새로운 해를 만드는 교차(crossover)와 해를 설정된 확률만큼 임의로 변형시키는 변이(mutation)의 단계를 거쳐 다음 세대의 해집단을 만들고, 이 해집단을 다시 objective function 에 대입하여 그 세대의 최적의 해를 선택하게 된다. 이러한 과정을 임의의 정지조건이 만족될 때까지 수행한 후 해집단에 남은 해 중 가장 좋은 해를 답으로 삼는다.

	Case 1	Case 2	Case 3	Case 4
00	←	↑	→	←
01	↑			
10	→	→	↓	↑
11	↓			

Table 4 Meaning of chromosome

Table 1 은 genetic algorithm 의 chromosome 이 의미하는 방향을 보여주고 있다. 교차점에서 다음 도로로 진입하는 경우의 수는 최대 4 가지이므로 이를 나타내기 위해 최소 2 개의 chromosome 이 하나의 방향을 의미하는 것으로 정하였다. 그리고 어느 한 도로를 통행한 후 도로와 도로

의 교차지점에 도달하였을 때, 다음 도로로 진행할 수 있는 방향의 경우의 수는 어떤 교차점에 도착하였느냐에 따라 다르게 된다. 따라서 교차점들을 4 가지 경우로 나누어 각 경우에 따라 chromosome 이 의미하는 방향을 각기 다르게 적용시켰다. Figure 1 에서 Case 1 에 해당하는 교차점은 별 모양에 해당하는 9 개의 교차점, Case 2 에 해당하는 교차점은 세모 모양에 해당하는 6 개의 교차점, Case 3 에 해당하는 교차점은 네모 모양에 해당하는 3 개의 교차점, Case 4 에 해당하는 교차점은 X 모양에 해당하는 3 개의 교차점이다.

```
mode=binary
population_size=100
generation_number=1000
mutation=0.01
crossover=0.8
score_frequency=10
width=2
height=40
```

Figure 3 input.txt

Figure 3 는 Binary coded genetic algorithm 의 설정 값들의 정보를 저장하고 있는 input.txt 의 내용이다.

먼저 chromosome 의 개수를 나타내는 width 와 height 는 각각 2 와 40 으로 chromosome 은 총 80 개가 된다. 왜냐하면 도로의 개수가 40 개 이므로 최적 경로를 찾는다는 가정 아래에서는 통행자가 40 개 이상의 도로를 통과하지 않는다. 따라서 chromosome 의 크기는 통행자가 모든 도로를 통행한다는 최악의 경우까지 고려하여 80 개로 정하였다. 그리고 mutation 과 crossover 에 대해서 이들 값들의 설정에 대한 명확한 기준은 찾을 수 없었고, mutation 은 0.5%~1%, crossover 는 80%~95%의 값이 주로 사용되므로, mutation 은 0.01, crossover 는 0.8 로 설정하였다. 또한 population size 와 generation number 는 본 논문에서 수행하고자 하는 genetic algorithm 이 많은 시간이 요구하는 복잡한 것이 아니므로, 여러 번 실행시켜 일정한 결과가 나오도록 설정하였다.

Genetic algorithm 의 objective function 은 아래와 같이 나타내었다.

```
For(int k=0; k<40; k++)
{
    result = result – (timeary[k] * hardary[k]) * (road_count1[k] + road_count2[k]);
}
```

result 값은 초기에 0 으로 설정되어 있다. 본 논문에서 사용한 Genetic algorithm 은 최대값을 찾도록 되어 있기 때문에, 찾아야 하는 최적경로는 가장 작은 피로도를 가지고 있어야 하므로 objective function 값이 음수를 가지도록 하였다. 만약 도로들을 통과한 횟수가 40 번을 초과하였을 때는 result 값을 -9999 로 리턴하도록 설정하여 다음 세대로 해당 population 이 전달되지 못하도록 하였다.

여기서 timeary 와 hardary 는 각각 k 번 도로의 특성을 나타내는 요소인 통행 시간과 도로의 상태를 의미하는 배열이다. 그리고 road_count1 와 road_count2 는 통행자가 생성된 chromosome 에 의해 도로를 통행할 때, k 번 도로를 몇 번 통행하였는지 저장하는 배열이다.

2.3 결과

본 논문에서 만든 genetic algorithm 을 이용한 최적경로찾기 알고리즘의 몇 가지 결과를 제시한다.

2.3.1 Example 1

도로의 특성을 저장하는 배열인 timeary 와 hardary 는 임의로 다음과 같이 설정하였다.

timeary[40]={1,2,3,4,5,6,7,8,9,10, 1,2,3,4,5,6,7,8,9,10, 1,2,3,4,5,6,7,8,9,10, 1,2,3,4,5,6,7,8,9,10}

hardary[40]={1,2,3,4,5,6,7,8,9,10, 1,2,3,4,5,6,7,8,9,10, 1,2,3,4,5,6,7,8,9,10, 1,2,3,4,5,6,7,8,9,10}

output.txt

0	-1210.3	-264	-9999
40	-518.25	-200	-9999
80	-320.05	-200	-1478
120	-269.34	-200	-918
160	-507.86	-200	-9999
200	-280.24	-200	-884
240	-454.6	-200	-9999
280	-273.96	-200	-926
320	-254.44	-200	-722
360	-263.64	-200	-1504
400	-293.79	-200	-1042
440	-367.03	-200	-9999
480	-378	-200	-9999
520	-410.55	-200	-9999
560	-231.98	-200	-994
600	-396.98	-200	-9999
640	-248.75	-200	-1406
680	-385.98	-200	-9999
720	-292.58	-200	-1258
760	-427.54	-200	-9999
800	-250.2	-200	-936
840	-325.84	-200	-9999
880	-332.25	-200	-9999
920	-321.84	-200	-1656
960	-229.27	-200	-1391
1000	-312.85	-200	-9999

Figure 4 output.txt of first example

Figure 4 는 첫번째 결과의 objective function 의 평균값, 최고값, 최저값을 보여주고 있다. 처음

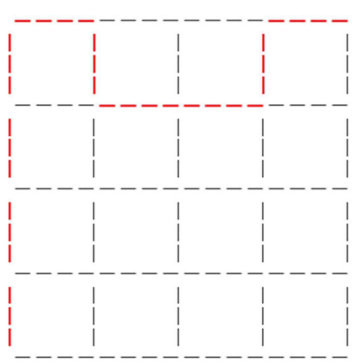
세대에서는 평균값이 -1210.3 으로 평균적으로 어려운 도로로 통행을 했다. 하지만 세대가 지나갈수록 평균값이 -500~-300 으로, 해들이 평균적으로 통행이 편한 도로를 선택하였고, 최고값이 -200 으로 최적경로를 찾은 것을 알 수 있다.

```
[kkn@localhost 0226]$ ./exe

mode=binary
population_size=100
generation_number=1000
mutation=0.01
crossover=0.8
score_frequency=10
width=2
height=40


Finding optimal path using Binary coded genetic algorithm


Number of passed road : 10
```



```
The GA found:
01
00
01
00
10
10
10
01

best of generation data are in 'output.txt'
```

Figure 5 result of first example

Figure 5 는 command 창의 결과를 보여준다. 최적 경로를 선택하였을 때 통행하는 도로의 수는 10 개이고, 해당 도로들은 빨간색으로 표시하였다. 그리고 최적 경로를 도출한 genetic algorithm 의 chromosome 도 command 창에서 확인할 수 있다.

2.3.1 Example 2

두번째 예제의 도로 특성을 저장하는 배열인 timeary 와 hardary 는 임의로 다음과 같이 설정하였다.

timeary[40]={10,9,8,7,6,5,4,3,2,1,10,9,8,7,6,5,4,3,2,1,10,9,8,7,6,5,4,3,2,1,10,9,8,7,6,5,4,3,2,1}

hardary[40]={ 10,9,8,7,6,5,4,3,2,1,10,9,8,7,6,5,4,3,2,1,10,9,8,7,6,5,4,3,2,1,10,9,8,7,6,5,4,3,2,1}

output.txt

0	-965.76	-256	-9999
40	-381.71	-200	-9999
80	-319.84	-200	-1904
120	-377.27	-200	-9999
160	-317.09	-196	-9999
200	-284.1	-196	-1430
240	-248.5	-196	-924
280	-337.07	-196	-9999
320	-229.54	-196	-834
360	-289.03	-196	-1232
400	-220.26	-194	-458
440	-247.96	-194	-756
480	-226.05	-194	-641
520	-239.8	-194	-566
560	-396.85	-194	-9999
600	-212.04	-194	-430
640	-252.8	-194	-1170
680	-232.72	-194	-988
720	-238.71	-194	-703
760	-277.78	-194	-924
800	-235.16	-194	-622
840	-360.51	-194	-9999
880	-365.51	-194	-9999
920	-355.53	-194	-9999
960	-226.42	-194	-708
1000	-247.62	-194	-1104

Figure 6 output.txt of second example

Figure 6 는 두번째 결과의 objective function 의 평균값, 최고값, 최저값을 보여주고 있다. 처음 세대에서는 평균값이 -965.76 으로 평균적으로 어려운 도로로 통행을 했다. 하지만 세대가 지나갈수록 평균값이 -400~-200 으로, 해들이 평균적으로 통행이 편한 도로를 선택하였고, 최고값이 -194 로 약 400 번째 세대에서 최적경로를 찾은 것을 알 수 있다.

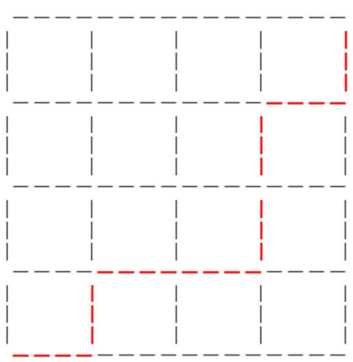
Figure 7 는 command 창의 결과를 보여준다. 최적 경로를 선택하였을 때 통행하는 도로의 수는 8 개이고, 해당 도로들은 빨간색으로 표시하였다. 그리고 최적 경로를 도출한 genetic algorithm 의 chromosome 도 command 창에서 확인할 수 있다.

```
[kkn@localhost 0226]$ ./exe

mode=binary
population_size=100
generation_number=1000
mutation=0.01
crossover=0.8
score_frequency=40
width=2
height=40

Finding optimal path using Binary coded genetic algorithm

Number of passed road : 8
```



```
The GA found:
10
00
10
10
01
01
10

best of generation data are in 'output.txt'
```

Figure 7 result of second example

3. 결 론

본 논문에서는 통행에 영향을 주는 도로의 속성을 고려하여 만든 경로 탐색 모델에서, genetic algorithm 을 이용하여 최적 경로를 탐색하였다. 2.3 의 두 가지 예제에서도 확인할 수 있듯이, 주로 통행시간과 도로상태를 의미하는 두 속성의 곱이 작은 도로들로 구성된 경로를 최적 경로로 선택하며 성공적으로 최적 경로를 찾아낼 수 있었다.

하지만 최적경로탐색을 위한 알고리즘에는 genetic algorithm 보다 더 빠르고 뛰어난 알고리즘이 존재하기 때문에, 경로 탐색 모델을 이보다 더 복잡하게 확장할 경우 보완이 필요할 것으로 판단된다.

감사의글

Genetic algorithm 으로 최적 경로 탐색 알고리즘을 만들면서, 이전에는 어렴풋하게만 알며 사용했던 genetic algorithm 의 개념을 책을 빌려 공부하면서 더 확실하게 이해할 수 있었다. 그리고 무엇보다 genetic algorithm 의 open source 중에 하나인 Galib 을 기반으로 하여 코드를 만들었기 때문에, C 및 C++을 더 자유롭게 다룰 수 있게 된 계기가 되었다.

참고문헌

1. 문병로, 2003, 유전 알고리즘, *두양사*.
2. 문병로, 2008, 쉽게 배우는 유전 알고리즘 : 진화적 접근법, *한빛 미디어*
3. Galib manual, http://lancet.mit.edu/ga/.