GA2015 Project1 Report

2015-21242 박준호

**1. Abstraction**

유전 알고리즘은 선택, 교차, 변이, 세대교체를 수없이 반복하여 문제공간을 탐색하는 방법이다. 이번 프로젝트에서는 지역 최적화 성격을 가지지 않는 순수 유전 알고리즘을 이용하여 문제에 접근한다. 최대 318개의 도시를 가지는 Traveling Salesman Problem의 문제공간을 탐색하여 최적 해를 구하는 것이 목표이다.

**2. Introduction**

유전 알고리즘은 집단 유전학의 개체 진화 원리를 이용하는 문제공간 탐색 방법이다. 기존에 확정적으로 구할 수 없었던 NP-hard 문제들이 대표적인 유전 알고리즘이 사용되는 문제군이며, 유전학에서 사용되는 여러 현상들이 operation으로 등장한다. 한 개체를 구성하는 DNA를 Chromosome으로 지칭하여, 두 부모 개체를 선택하여 각 부모의 Chromosome을 섞은 후 이전 세대와 교체한다. Selection -Crossover - Mutation - Replace 의 단순한 반복으로 평균적으로 이전보다 좋은 Chromosome 을 만든다면, 결국에는 가장 최적화된 해를 찾을 수 있을 것이라는 점이 가장 큰 뼈대라고 할 수 있다. 하지만, 각 단계에서 어떤 방법을 쓰느냐에 따라 최적화된 해를 찾을 수도, 찾지 못한 채로 세대의 모든 Chromosome들이 수렴해버릴 수도 있다. 따라서, 각 단계에서 어떤 방법으로, 어떤 Chromosome을 선택하여 교차, 변이하여 기존의 어떤 Chromosome을 교체할 것인가를 선택하는 것이 해의 품질에 지대한 영향을 끼친다.

Traveling Salesman Problem(이하 TSP)은 위치가 명시된 도시들이 있을 때, 그 도시들을 모두 연결하는 Cycle 중에 가장 짧은 경로를 찾는 문제이다. 이는 전형적인 NP-hard 문제이며, 다음에 설명되는 방법으로 최적 해를 구해보기로 한다.

**3. Methodology**

먼저, 유전 알고리즘 반복구간(2.에서 설명한 Selection ~ Replace에 이르는 일련의 과정)에 진입하기 전에 초기화 과정이 필요하다. 초기화 과정에서는 정해진 개수만큼 Chromosome을 만들며, 정해진 개수를 Population이라 하며, Chromosome들의 초기값은 임의로 정해진다. 유전 알고리즘 반복구간이 종료되면, 이제까지의 해들 중 가장 품질이 좋은 해를 출력하고 프로그램을 종료한다. Representation중 Encoding의 경우, Order-based Encoding을 사용하였다.

**3.1. Selection**

품질에 비례한 룰렛 휠 선택 알고리즘은 가장 기본적이고 대표적인 선택 방법으로서, 각 해의 품질을 선택 한 후, 가장 좋은 해의 적합도가 가장 나쁜 해의 적합도의 K배(K>1)가 되도록 하여 가상의 룰렛에서 선택될 확률이 적합도에 비례하도록 선택하는 방식이다.

**3.2. Crossover**

TSP처럼 염색체가 순열(Permutation)로 표현 가능한 경우, 사이클 교차, 순서 교차, PMX등의 방법이 있다. 이 중 PMX(Partially Matched Crossover)를 이용하여 교차 연산을 구현하였다. PMX는 자식 Chromosome이 부모(p1, p2)의 Chromosome들과 부분적으로 매치가 되도록 자식을 만드는 방법이다. 먼저 첫 번째 부모(p1)로부터 임의의 범위만큼 Gene을 복사한 후, 나머지는 기본적으로 p2로부터 복사하되, 중복이 일어나면 최초로 복사한 p1의 범위에 해당하는 p2의 값을 복사한다.

**3.3. Mutation**

변이 연산은 부모해에 없는 속성을 자식에게 부여하여 염색체의 다양성을 야기하여 탐색 공간을 넓히는 용도로 사용된다. TSP에서는 순서가 인접한 도시가 연결된 순서만이 중요하므로, 일정 구간의 도시 이동 경로를 떼어 반대쪽에 붙여주는 의미를 가지게 되는 inversion mutation을 구현하였다.

**3.4. Replacement**

교체는 GA가 더 나아지기 위한 마지막 관문이다. 만약 이 단계에서 Crossover, Mutation된 자식을 반영하지 않는다면 GA는 Steady-state가 될 것이다. 따라서 이번 프로젝트에서는 너무 만들어진 자식의 품질이 부모보다 낮더라도 새로운 세대에 반영되도록 게 나오더라도 무조건적으로 반영하도록 하였다. 일해보았는데, 현재 세대 중 가장 나쁜 품질을 가진 것과 교체하였다.

**4. Problem Representation**

도시의 개수는 N으로 문제의 입력으로 주어진다. 각 개체는 Chromosome과 Fitness를 멤버로 가지며, Chromosome은 order-based encoding으로, 방문하는 도시의 inex를 값으로 가지는 길이 N의 배열로 정하였다. 각 세대는 PSIZE = 10 의 개수로 이루어진다. PSIZE가 적으면 수렴할 가능성이 높아지나, mutation으로 이를 어느 정도 극복할 수 있으며, 세대 교체 step의 시간 간격이 적어지는 이점이 있어 기존 값인 100에서 낮추었다.

**5. Result**

**5.1. Environment Specification**

사용된 CPU의 사양은 다음과 같다.

Intel® Core™ i7-3930K CPU @ 3.20GHz 3.20 GHz

**5.2. Statistics**

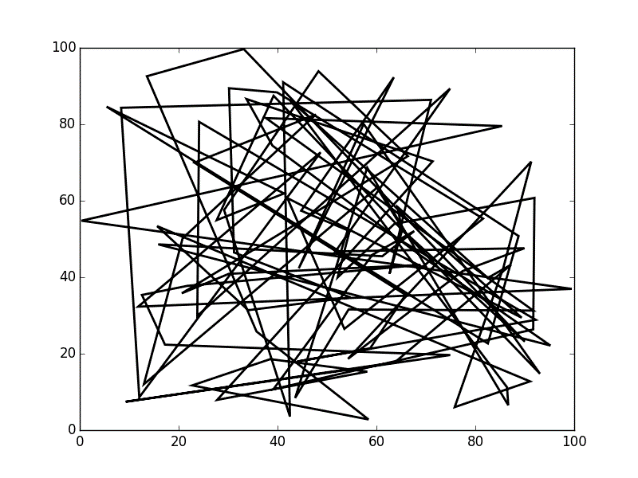
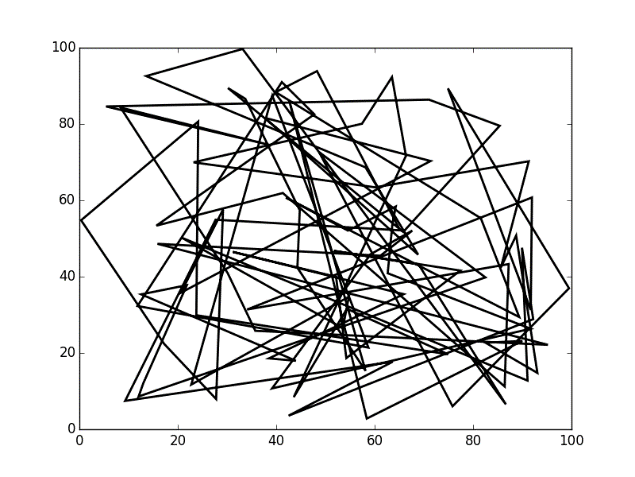
N = 11, 21, 51, 101, 318인 TSP에 대해 30번씩 알고리즘을 돌린 결과 가장 좋은 품질, 평균 품질, 품질간의 표준편차는 다음과 같다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N= | 11 | 21 | 51 | 101 | 318 |
| 최고 | 345.87524 | 399.711162 | 590.748887 | 799.920684 | 466.842214 |
| 평균 | 345.87524 | 399.711162 | 607.4774994 | 820.0599306 | 482.8114206 |
| 표준편차 | \*0 | \*0 | 8.723027172 | 10.56731052 | 8.845663934 |

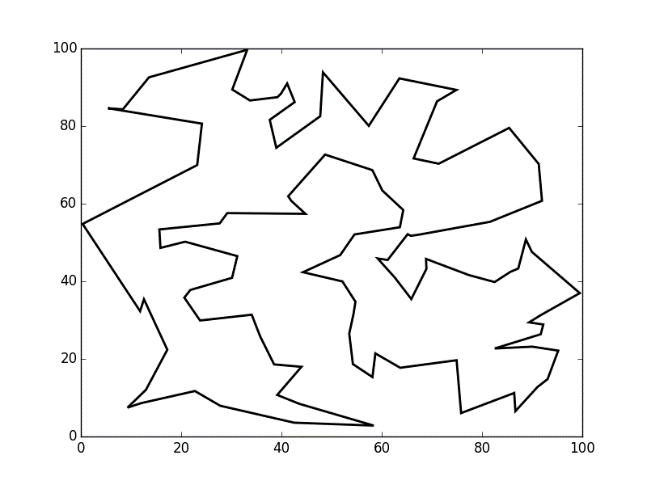
Table 1. GA를 이용하여 TSP Solving을 30회 진행한 결과. (\* : 1E-10보다 작은 경우 0으로 처리)

**5.3. Traveling Path Plot**

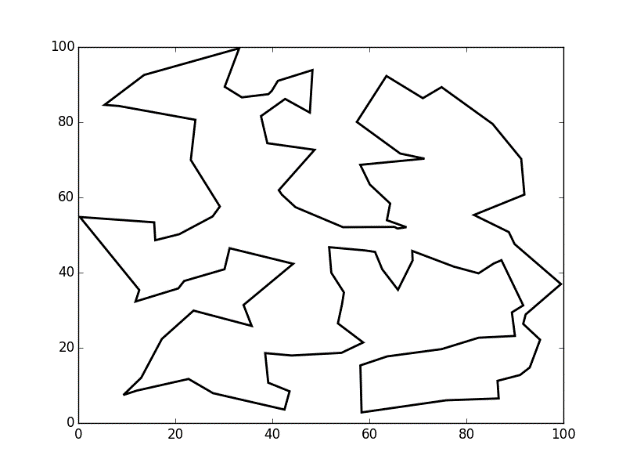
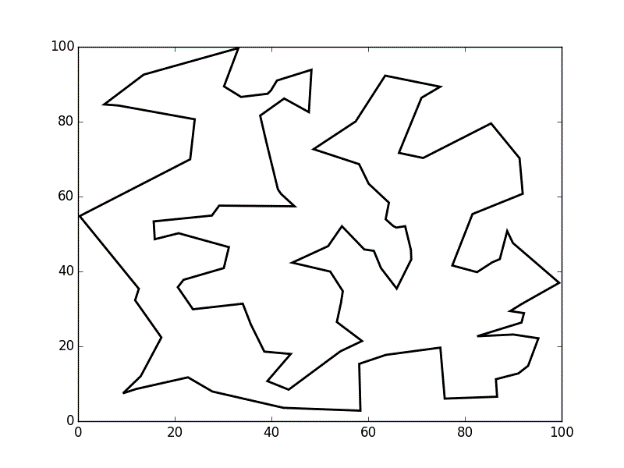
N = 101인 경우 GENERATION 1, 100, 10000, 1000000, Terminal Case(31153556)인 경우, 그 시점까지 얻어낸 최고 품질을 가진 해의 Traveling Plot은 다음과 같다.

.

**Figure 1. GENERATION = 1 Figure 2. GENERATION = 100**



**Figure 3. GENERATION = 10000 Figure 4. GENERATION = 1000000**

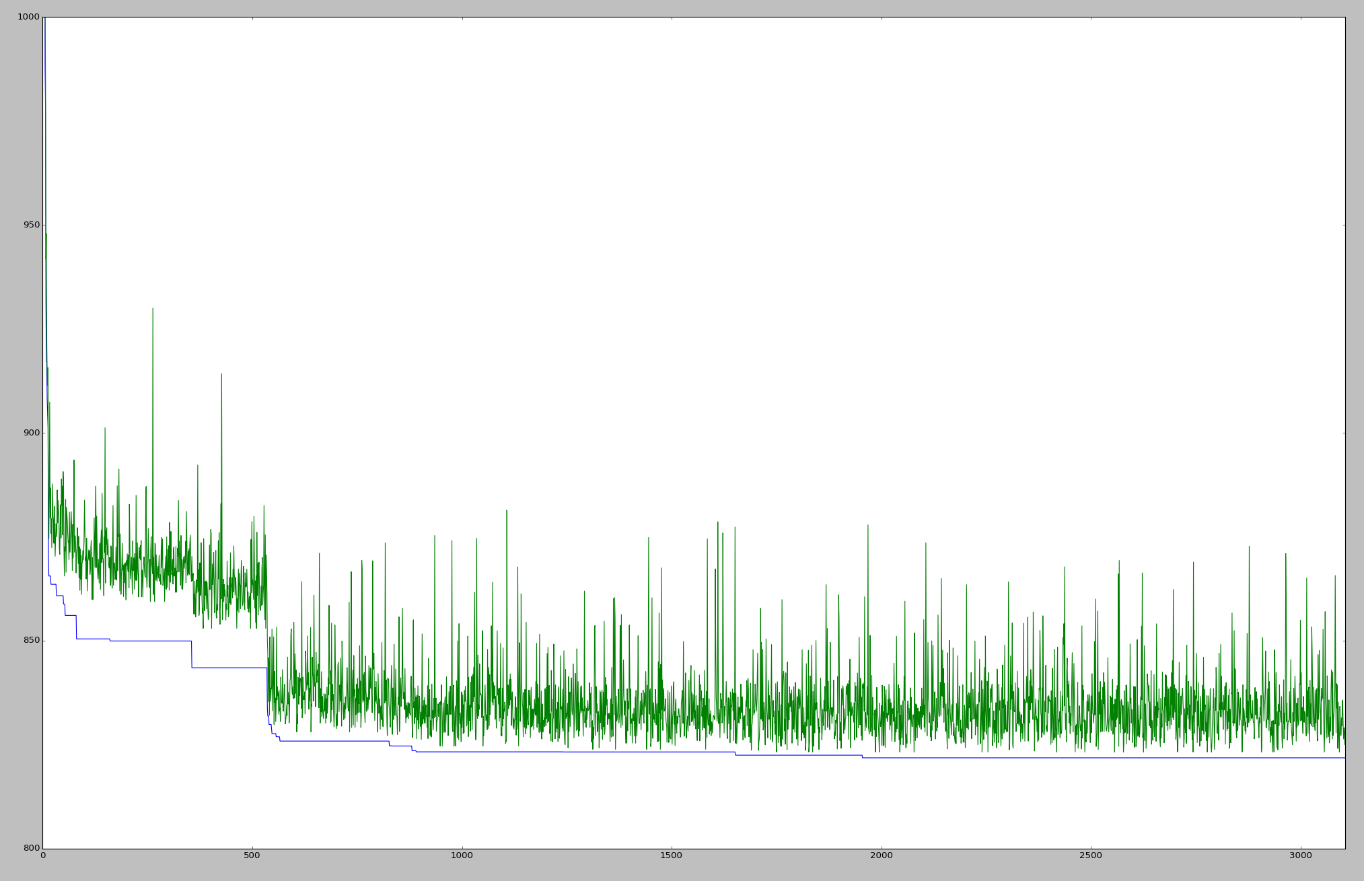


**Figure 5. Terminal (f = 808) Figure 6. Terminal (f = 799.9)**

N = 100 정도 까지는 개선이 눈에 띄는 정도로 이루어지지 않지만, 이후 10000까지 가면서 확연히 눈에 띄는 차이가 생겼고, 10000에서 1000000으로 가면 도시 이동 경로 간에 교차하는 경우도 사라지게 된다. **Figure5**와 **Figure6**은 서로 다른 fitness값으로 수렴하여 Timeout 종료된 경우의 결과인데, 부분적으로 동일한 경로를 공유함을 볼 수 있다.

**5.3. Generation – Fitness Plot**

N = 101인 경우 Generation이 거듭할수록 fitness의 변화를 10000번의 Generation마다 한번씩 측정하여 plotting한 결과는 다음과 같다. 해의 품질이 좋아지는 모양이 꾸준한 것이 아니라 Stepping하는 모양을 가지는 것을 확인할 수 있다.



**Figure 7. Generation – Fitness Plot for N = 101**

**6. Discussion**

Selection, Crossover, Mutation, Replacement 네 가지 GA 연산자마다 여러 종류의 방법이 있는데, 이 중 어느 것을 선택하느냐가 가장 중요한 문제였다. 각각 연산자들 간에 시너지를 내는 경우도 있고, 서로 도움이 되지 않는 경우도 있는데, 이번에 선택한 네 가시 연산자 Roulette-Wheel Selection / PMX / Inversion Mutation / Worst-case Replacement 중 PMX와 Inversion Mutation은 분석 결과 일정 품질 이상이 얻어지면 그 이후로는 해의 품질개선에 유효한 도움을 주지 못하는 채로 수렴하는 경우라고 볼 수 있다.

PMX는 부모가 동일하면 자식도 동일한 특징이 있는데, 이 경우 부모와 동일한 자식에 대해 시행하는 Inversion Mutation은 굉장히 높은 확률로 유효하지 않다. **Figure 4~6** 과 같이 부분적으로 self-crossing하는 형태가 아닌 자식이라면, Inversion Mutation을 한 결과는 기하학적 성질에 의하여 무조건 self-crossing하게 된다. 즉, Inversion을 시행하기 전보다 더 높은 fitness값을 가지게 되기 때문에 해의 품질 개선에 도움을 주지 못한다. 이 경우 reset을 하거나 crossover를 order-base crossover 등 부모가 동일한 경우에도 다른 자식을 만드는 방법을 사용하거나, mutation 방법을 다르게 하는 등의 연산을 통해서 극복할 수 있을 것으로 보인다.

이번 과제의 경우, 각 operator들을 구현하는 데 있어서는 생각보다 어렵지 않았으나, 연산자들의 조합에 따라 결과가 얼마나 유의미하게 바뀌는 지를 알기 위해서는 적어도 열 번 이상은 실행해 보아야 감이라도 잡을 수 있었는데, 초반에 코드를 수정할 때에는 이런 것을 알지 못하고 금방금방 바꾸어 버려 아쉬웠다. 또한, 문제공간을 결정하는 값인 N값이 적을 때는 빨리 수렴해도 찾을 수 있던 해들이 N값이 증가함에 따라 더 이상 나아지지 않는 상황도 발생했다.

지역 최적화 알고리즘을 적용할 수 있게 되면 더 좋은 결과를 얻을 수 있으리라 기대하기는 하지만, 그 전에 각 연산자들마다 어떤 효과를 가지게 되고, 어떤 변수를 조절하여 수렴과 최적 해를 컨트롤 할 수 있게 되는지 알게 되는 것도 중요할 것이다.