**Genetic Algorithm을 이용한 TPS 문제 해결**

한양대학교 산업공학과

주 우 종

**1. 개요**

Genetic Algorithm(GA)를 이용해 TPS문제를 풀어보았다. 오픈소스로 제공된 기존의 코드를 학습하고 이를 바탕으로 새로운 코드를 실행하여 성능과 결과를 비교하였다. 유전알고리즘의 성능은 Selection Method와 Crossover Method에 의해 결정된다. 오픈소스로 제공된 방법 이외에 다양한 Selection Method와 Crossover를 이용하여 성능비교가 진행될 것이다.

구성한 알고리즘의 작동원리는 다음과 같다

* 거쳐야할 도시들의 좌표 리스트가 입력된다
* 초기 Initial Population이 리스트를 기반으로 임의로 정해진다
* 각 Route의 총 거리의 합을 이용하여 적합도를 기반으로 Mating Pool을 형성한다
* 번식을 통한 새로운 Population을 구성하고 일정 확률로 돌연변이가 일어난다
* 위 과정을 반복하며 각 세대별 Population마다 최고의 Route를 구하며 성능을 비교한다.

위의 원리를 기반으로 랜덤하게 설정된 임의의 35개 도시에 대해 알고리즘별 성능을 비교할 것이다. 각각의 알고리즘은 서로 다른 Selection Method 또는 Crossever Method를 가질 것이다.

**2. 성능 비교 및 결과 해석**

각 알고리즘 별 임의의 35개 도시에 대하여 나타난 결과와 원인을 살펴볼 것이다.

실행할 모든 알고리즘의 기본 조건은 다음과 같다

Population Size : 50

Mutationrate : 0.01

Generation Number : 1000

도시의 위치는 100\*100 좌표평면 위의 임의의 한점으로 랜덤하게 지정되고, 도시 사이의 거리는 Euclidian Distance를 이용한다. route상의 연결된 모든 Euclidian의 합의 역수형태로 fitness를 지정하여 distance의 합이 작아질수록 좋은 fitness가 적용되는 형식이다.

1)

기본 코드

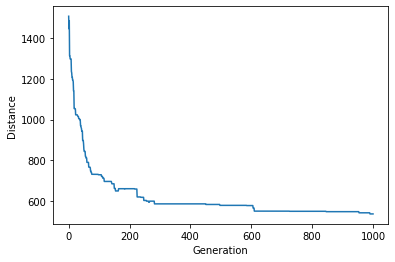
Selection Method : 룰렛 휠 + Elitism, Crossover : Ordered Crossover

Elite Size : 10

First Distance :1507.378359879382

Final Distance :537.3768753232741

Time :23.37152647972107



룰렛휠 방법을 이용해 번식에 참여할 parents를 선택한다. 룰렛 휠 방법은 각 individual의 fitness에 비례한 확률을 부여하여 그 확률에 따라 번식에 참여하는 parent를 선택한다. 또한 elitism을 적용해 10개의 Elite size를 지정하여 50개의 population내의 성능이 가장 좋은 상위 10개의 individual은 항상 reproduction에 참여한다.

1000번의 generation후에 나온 최적해는 그래프를 보았을 때 효과적으로 감소함을 볼 수 있다.

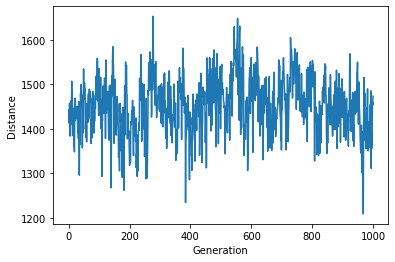
2)

Selection Method : 룰렛 휠, Crossover : Ordered Crossover

First Distance :1443.00810087244

Final Distance :1208.8498180658376

Time :25.434972524642944



1)과 같은 알고리즘에서 Elitism을 제외하고 돌려본 결과이다. 그래프를 보았을 때 최적해 값이 감감소하지 않고 진동하고 있음으로 좋은 해라고 볼 수 없다.

위와 같은 결과가 나온 이유는 룰렛휠과 Ordered Crossover만으론 부모의 우수한 유전자가 다음 세대에게 이어질 수 없기 때문이라 생각한다. 임의로 시작하는 첫 population내 individual간의 fitness는 각각의 차이가 미세할 뿐 아니라, 그중 그나마 좋은 유전자가 선택된다고 하더라도 ordered crossover를 적용하면 다음세대로 이어지는 과정에서 좋은 fitness가 파괴될 가능성이 있다. 따라서 이 문제는 Elitisim과 같은 추가적인 방법이 함께 병행되어야 좋은 성과를 낼 수 있을있을 것이다.

3)

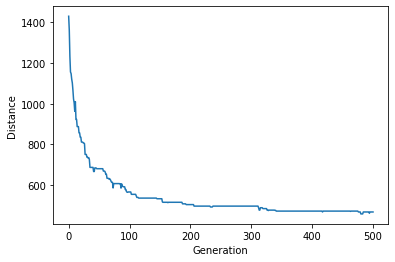
Selection Method : 토너먼트 , Crossover : Ordered Crossover

토너먼트 사이즈 : 10

First Distance :1429.1887978745367

Final Distance :459.2538269093511

Time :25.649434328079224



Crossover방법은 그대로 유지하되 selection방법을 tournament 선택 방법으로 변경한 경우이다. selection시에 population내의 10개의 individual을 임의로 고른 후 가장 fitness가 좋은 individual이 참여한다. 위와 같은 방법은 토너먼트 사이즈가 클수록 좋은 유전자가 다음 세대로 넘어갈 가능성이 룰렛휠 방법에 비해 높기 때문에 Elitism없이도 좋은 결과가 나온 것으로 보인다.

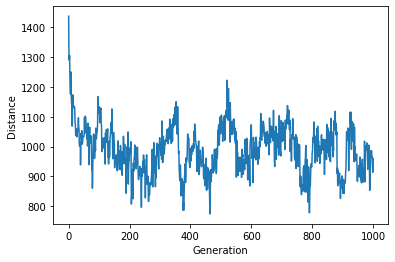
4)

Selection Method : 랭크 , Crossover : Ordered Crossover

First Distance :1437.7538632193778

Final Distance :773.6824007517994

Time :23.360480308532715



위의 방법은 Elitism없이 selection method를 Rank method로 변경한 경우이다. Rank Method는 individual들의 fitness에 따라 Rank를 부여하고 Rank에 따른 selection possibility를 부여하는 방법이다. 이러한 방법을 사사용한 경우 generation 초반 어느정도까지는 성능이 좋아지지만 이후엔 랜덤하게 진동하는 그래프가 나타났다. 룰렛 휠보단 초반에 좋은 유전자에게 부여되는 확률이 커지기 때문에 좋은 성능을 보인다. 하지만 그 이후부터 orderd crossover시에 생기는 유전자 변형 의 정도가 좋은 유전자를 선택하는 확률을 압도해버려 이러한 결과가 나온것이라 생각한다.

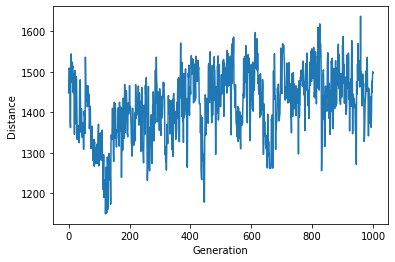
5)

Selection Method : 룰렛휠 , Crossover : PMX

First Distance :1447.849114091419

Final Distance :1148.6536907720495

Time :26.390929698944092



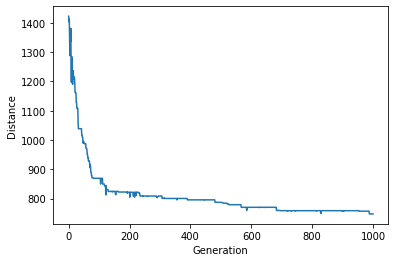
위의 경우는 룰렛휠 방법 선택을 그대로 적용하되, Crossover 방법을 PMX로 바꾼 경우이다. 그래프는 좋지 못한 성능을 보여준다. PMX를 적용하여 ordered crossover 방식에 비해 더 규칙적인 유전자를 후손에게 전달 가능하다. 하지만 위해 초기 무작위로 형성된 initial population내의 individual사이의 fitness차이가 미미하여 룰렛휠 방법을 사용할 경우 좋은 유전자가 후손에게 내려오기는 어려워 위와 같은 결과가 나타난 것이라 생각한다.

6)

Selection Method : 랭크 , Crossover : PMX  
First Distance :1422.7442862129217

Final Distance :747.6181954697454

Time :22.813005685806274



위는 crossover방식은 PMX로 유지하지만 selection방법을 rank method로 변경하였을 때의 결과이다. 이러한 방법을 적용했을 경우 효과적으로 Distance가 감소하는 것을 볼 수 있다.

**3. 결론 및 한계**

다양한 Selection method와 Crossover Method를 이용하여 TPS문제를 풀어본 결과 다양한 성능의 차이를 볼 수 있었다. 그러므로 유전알고리즘을 적용하기 위해선 문제의 성격과 특성을 파악하고 적절한 selection method와 crossover method를 결합하여야 좋은 결과를 얻을 수 있다는 결론을 얻을 수 있었다. TPS 뿐만 아니라 다른 다양한 문제에서도 같은 결론이 적용될 것이라 볼 수 있다.

현재의 한계점은 다양한 selection method와 crossover method를 모두 비교분석 해본 결과가 아니라는 점이다. 위의 적용된 방법들은 다양한 방법론들 중 일부기에 더 빠른 시간에 좋은 성능을 내는 결과가 존재할 수 있다. 또한 현실반영을 거치지 않은 Euclidian distance를 사용하였기 때문에 즉각적 적용이 어렵다. 하지만 이는 fitness와 distance계산의 일부분만 적절히 수정하여 적용하면 해결될 수 있는 문제일 것이다. 마지막으로 유전알고리즘의 한계점이다. 최종적으로 나온 결과값이 Global optimum이 아닐 수 있다. 하지만 TPS문제처럼 많은 연산량을 필요로 하는 문제의 경우[O(n!)] 효과적인 Decision making 방안이 될 수 있다고 생각한다.

참고문헌

1. Eric Stoltz, Evolution of a salesman : A complete genetic algorithm tutorial for Python

(<https://towardsdatascience.com/evolution-of-a-salesman-a-complete-genetic-algorithm-tutorial-for-python-6fe5d2b3ca35>)

2. Nisha Saini, Review of Selection Methods in Genetic Algorithms, IJECS Volume 6 Issue 12 December 2017 Page No. 23261-23263