**HW#1 Genetic Algorithm**

**PNU Computer Science & Engineering**

**201624476 박 상 운**

**Date : 2018/04/16**

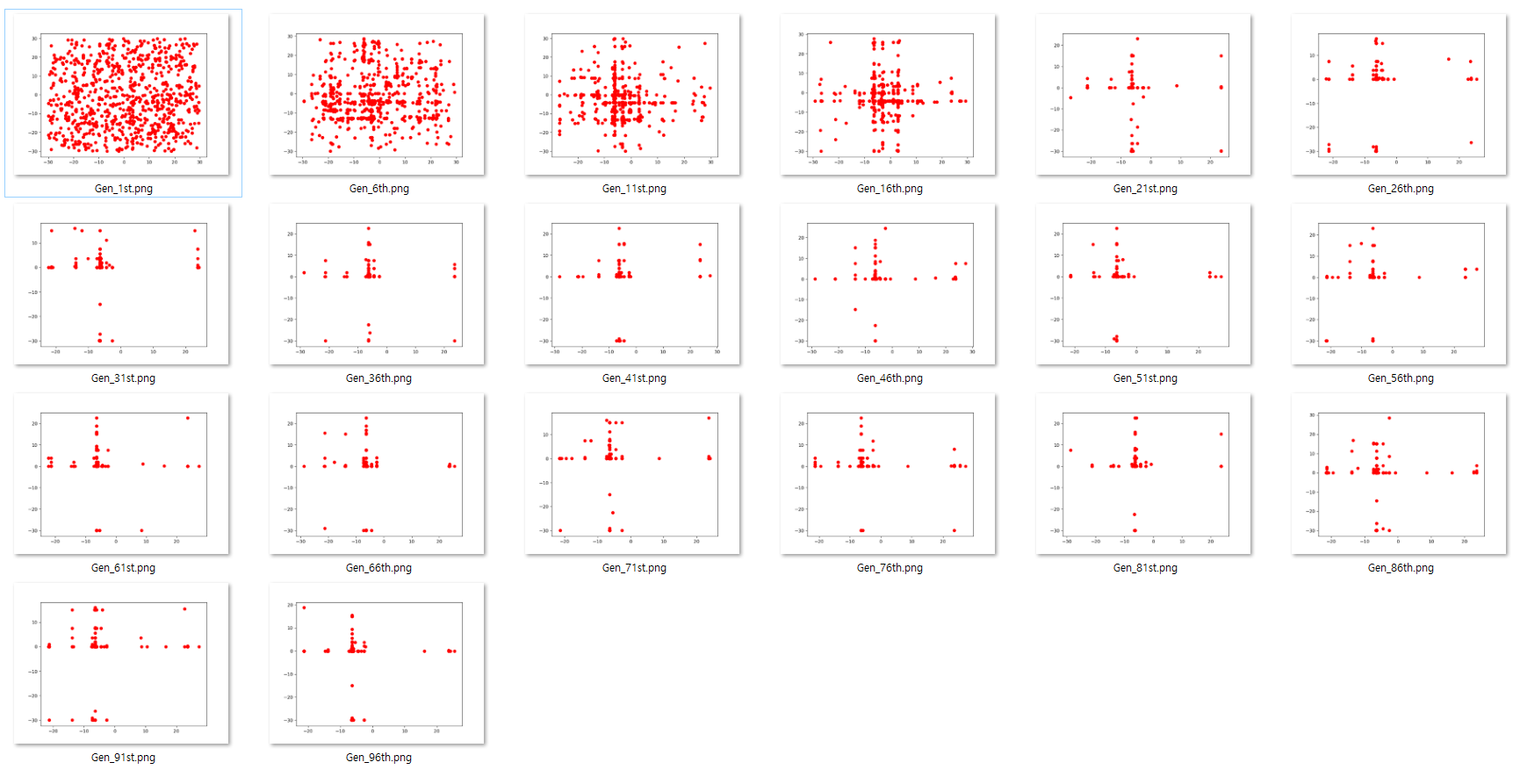
1. **최소화 문제에서는 목적함수의 값이 작을수록 좋지만 GA에서는 fitness 값이 클수록 좋기 때 문에 목적함수를 그대로 fitness로 사용할 수 없다. 어떤 다른 방법이 필요한가?**

**A:** 목적함수의 값에 -1을 곱하면 함수의 대소관계가 역전되므로 Maximum을 구하는 GA를 그대로 사용할 수 있습니다.

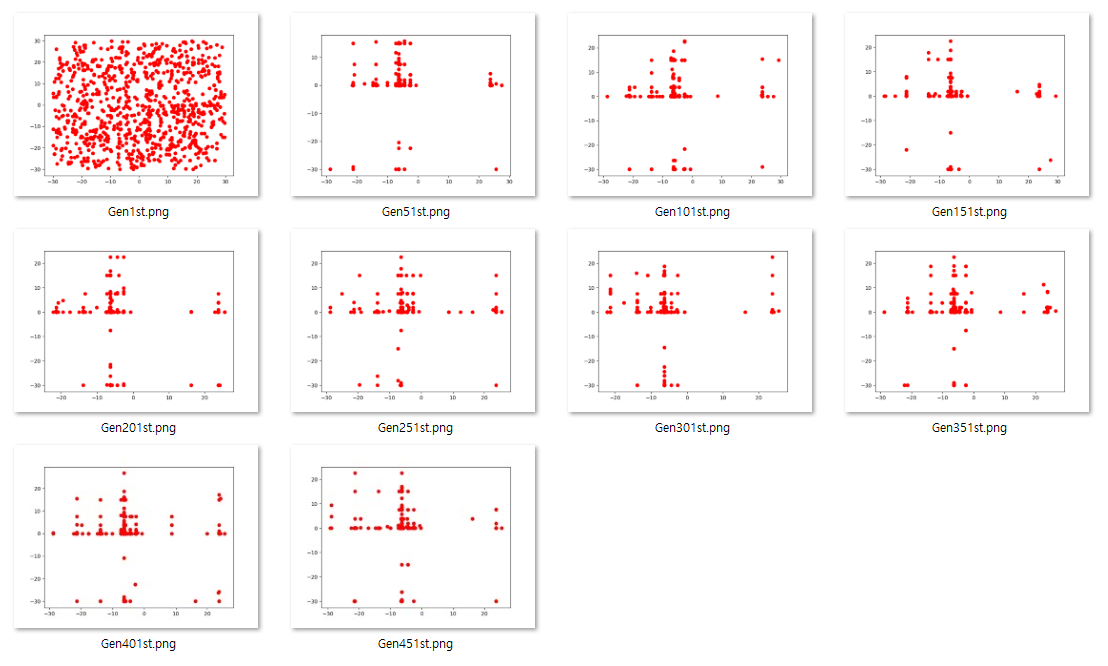
1. **Python 기반의 DEAP library를 이용하여 위의 최적화 프로그램을 구현하고, visualization 도구 matplotlib를 사용하여 2차원 평면 상에서 population 내 해의 분포가 어떻게 진화하는지를 관찰 하는 실험을 수행하라. 실험결과 보고에 포함되어야 할 내용들은 다음과 같다: (1) 해 표현은 어 떻게 했고 genetic operator로는 어떤 것들을 사용했는지와 그 이유, (2) crossover rate, mutation rate, population size 등을 결정하기 위한 실험 데이터, (3) 해 분포의 진화 모습을 보여 주는 (일정 세 대 간격으로 이루어진) 일련의 그림, (4) 탐색 결과로 얻어 낸 최소값 및 그를 실현하는 (x, y)의 값**

**A:** (1) 해의 표현은 원소가 (0,1) 중 하나의 값을 가지는 정수형 튜플(length = 16)으로 구현하였습니다. 해당 튜플 ind에 대해서 ind[0:8] 은 x의 genotype representation, ind[8:16]은 y의 genotype representation로 구현하였으며 과제의 참고 사항에 있는 decoding 작업을 이용해서 실수형 Phenotype representation으로 변환하여 사용하였습니다. operator로는 toolbox.population을 이용한 initialization, toolbox.mate를 이용한 crossover, toolbox.mutate를 이용한 mutation 등을 구현하였습니다.

(2) 각각의 조건 설정에 대한 진행 과정은 동봉된 result.zip에 첨부하였습니다. 결과적으로 해당 내용을 요약하자면, population이 길수록, Crossover가 활발할 수록, Mutation이 활발할수록, Generation이 높아질수록 더욱 더 최적화된 값을 찾아낼 수 있었습니다. 최종적으로 population = 1000, CrossoverRate = 50%, MutationRate = 50%, Generation = 500의 값이 동일 조건에서 CrossoverRate = 20%인 경우를 제외한 다른 모든 경우와 최적해가 동일하였음으로 이로 결정하였습니다.

(3) 

Figure



Figure

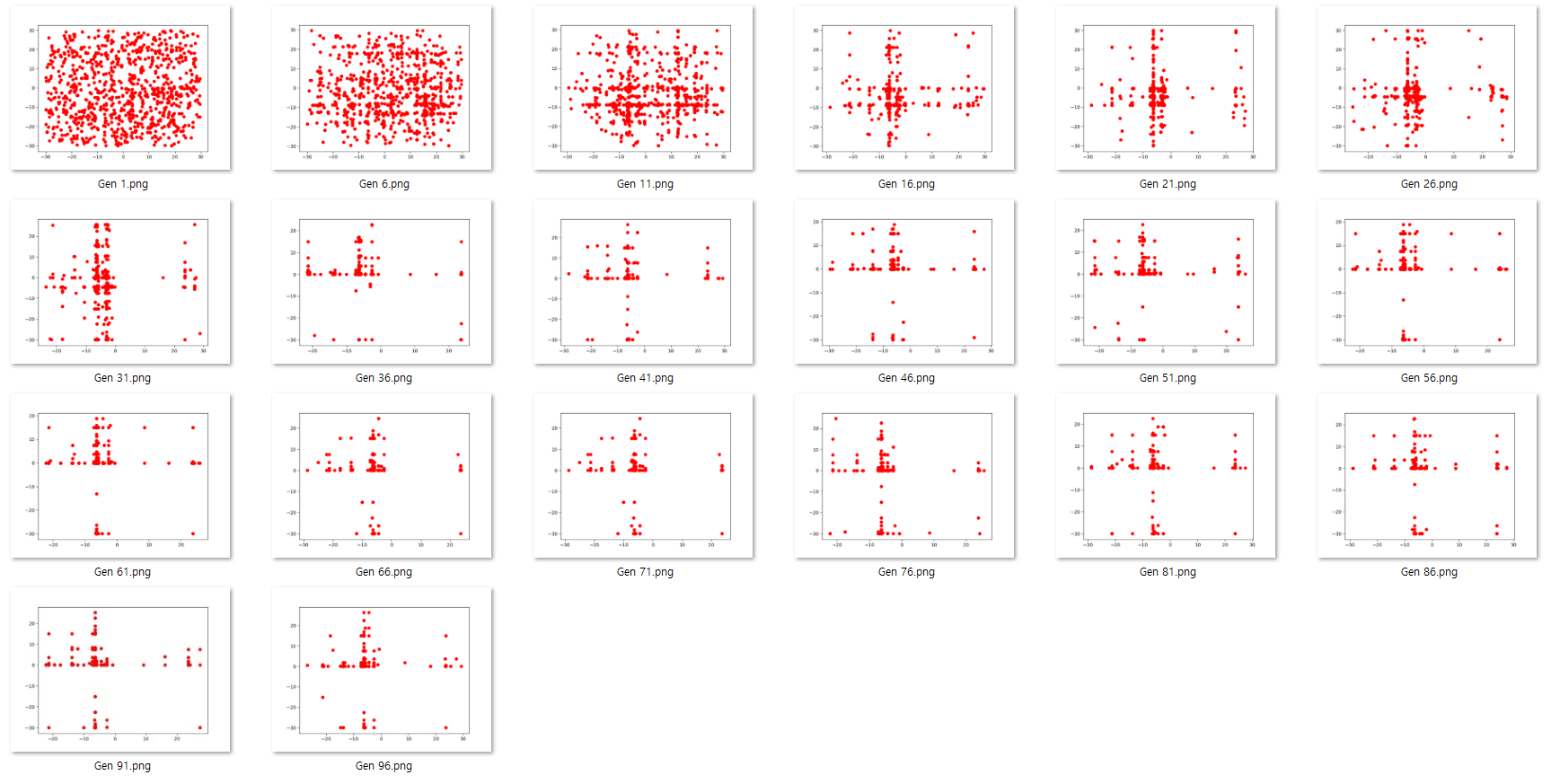
**Figure 1**은 population = 1000, CrossoverRate = 50%, MutationRate = 20%, GenerationLimit = 100 의 진행 상황을 5세대 간격으로 확인한 그림입니다. 21세대부터 분포의 퍼짐이 확연히 줄어드는 모습을 볼 수 있습니다. **Figure 2**는 population = 1000, CrossoverRate = 50%, MutationRate = 50%, GenerationLimit = 500의 경우를 50세대 간격으로 확인한 그림입니다.

(4) Best individual is ( -6.328125 , 0.000000 ). In this case, f(x,y) = 0.011021.

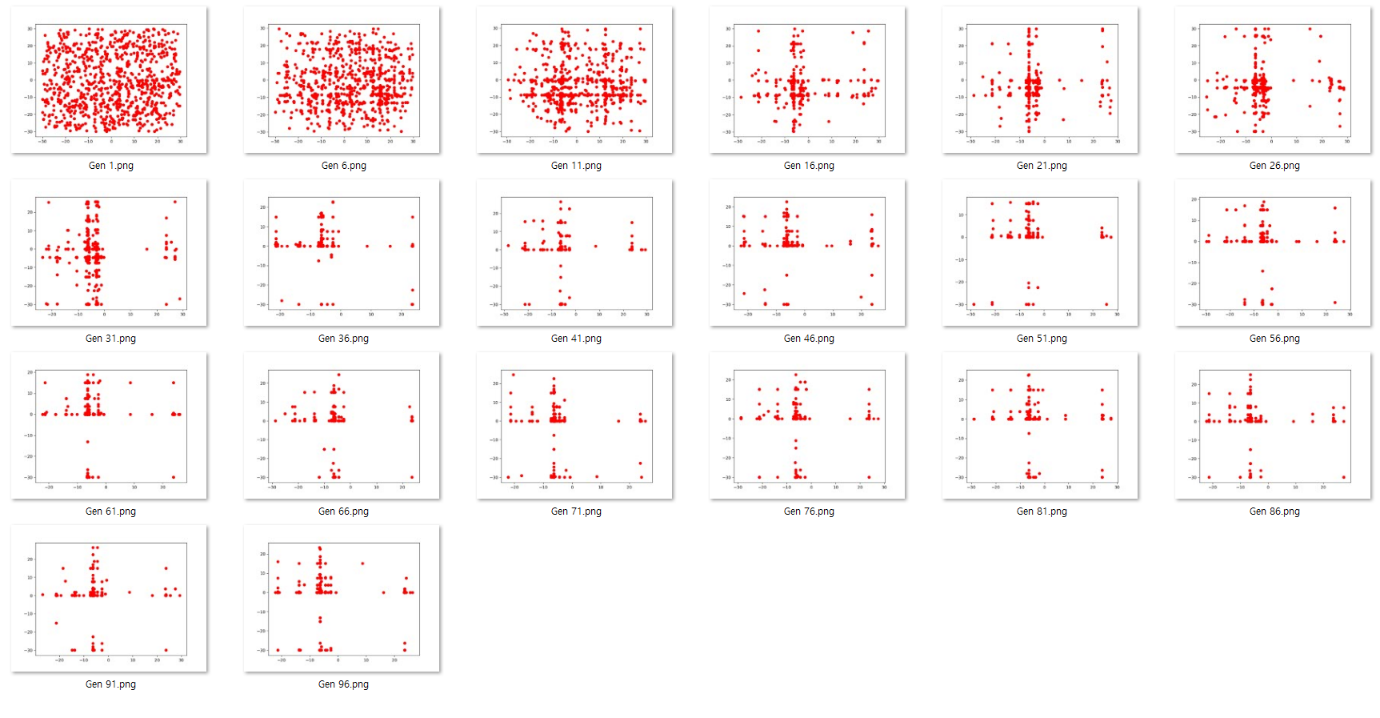
1. **전역적 최소값을 달성하는 단 하나의 해 (x, y) 만 구하는 것이 아니라 전역적 최소와 크게 차 이가 나지 않는 여러 해들을 구하고 싶을 때가 있다. 이럴 때 사용하는 대표적인 기법 중 하나 로 fitness sharing이라는 것이 있다. 이 기법은 각 개체의 fitness 값을 그대로 사용하지 않고 그 개체 근처에 있는 다른 개체들의 수가 많을수록 그만큼 fitness 값을 줄여서 사용함으로써 탐색 공간 내의 특정 지역에 개체들이 쏠리는 현상을 막는다. f(xi)가 개체 xi의 원래 fitness라 하면 sharing에 의해 줄어든 fitness의 값 f¢ (xi)는 아래와 같이 계산된다. 여기서 n은 population의 크기이고 dij는 xi와 xj 사이의 거리이며 s는 유사도 함수로서 두 개체 사이의 거리가 dij일 때 niche radius라 부르는 s 를 이용하여 아래와 같이 유사도를 계산한다. 1 ( 30 , 30) 2 cos( ) cos 4000 ( , ) 2 2 ÷ + - £ £ ø ö ç è æ - + = x y y x x y f x y å = ¢ = n j ij i i s d f f 1 ( ) (x ) (x ) î í ì - < = otherwise if 0 1 ( / ) ( ) ij s ij s ij d d s d 2 위의 Griewank 함수 문제에 fitness sharing을 적용하면 어떻게 되는지 실험하되 s 의 값 크기에 따라 결과가 어떻게 달라지는지 관찰해 보라. s 가 너무 크거나 작으면 어떻게 되는가? 이론적 으로 가장 좋은 값은 무엇이라 생각하는가?**

(다음 장에)

**A:**

****

Figure

****

Figure

Figure 3과 4는 각각 Population = 1000, CrossoverRate = 50%, MutationRate = 50% 로 설정한 상태에서 Niche Radius를 각각 1.0 / 0.5로 설정한 그림입니다. Figure 1과 비교해보면 Niche Radius 값이 커질수록 세대가 거듭되어도 각각의 individual의 분포도 비례해서 넓어짐을 볼 수 있습니다. 이는 탐색할 수 있는 상태공간의 경우의 수가 더 넓어짐을 뜻하고, 따라서 최적해를 찾을 가능성이 조금이라도 더 높아진다고 볼 수 있습니다. Niche Radius가 너무 작을 경우 fitness sharing이 제대로 작동하지 않을 수 있으며, Radius가 너무 커지게 될 경우 거리가 멀어서 영향력이 떨어지는 다른 individual까지도 영향을 주게 되므로 탄력적인 값 추출이 어려워집니다. 따라서 좋은 Niche Radius값은 각각의 individual이 일정한 거리를 가지도록 권장하는 값을 가지도록 하는 것이 좋으며, 이는 비둘기집의 원리에 따라서 individual이 존재할 수 있는 단위 면적 당 1개의 individual이 존재한다고 가정할 때의 간격인 2 \* sqrt((x값의 boundary \* y값의 boundary) / (population))이 최적이라고 생각합니다.