4월 1일 과제: Genetic Algorithm

학번 : 2017320208

학과 : 컴퓨터학과

이름 : 이연준

1. 두 개의 다른 mutation strategy : 'Shuffle'과 ‘Swap & Shuffle'로 명명.

(1) Shuffle 전략 (이후 1번 전략)

- 첫 번째 세대의 solution population 10개는 모두 무작위 숫자 10개를 각 solution하나에 배정을 시켰다.

- 그 세대의 자식들(offspring)을 생성하는 전략은 무작위로 부모 세대에서 3개를 선택하여 그 3개의 선택지에서 각 solution의 fitness value값에 비중(weight)를 주어서 그 비중에 맞는 확률로 2개가 선택되어서 그 2개끼리 crossover가 되도록 하였다(fitness proportionate selection).

- 2개의 solution에서의 Crossover point는 무작위로(random) 선택해서 crossover 시켰다.

- 이후 위의 과정을 10번 반복해서 생성된 10개의 자식 solution에서의 mutation 전략은 각 솔루션의 각 셀(cell)마다 0.05%의 확률로 무작위(random) 값으로 전환이 되도록 설정을 해놨다(Shuffle).

- mutation이 일어난 이후 다음 세대의 population을 선택(selection)하는 방식은 원래 있던 population과 자식 population 중에서 가장 fitness value가 높은 10개를 선택하는 elite selection 전략을 사용하였다.

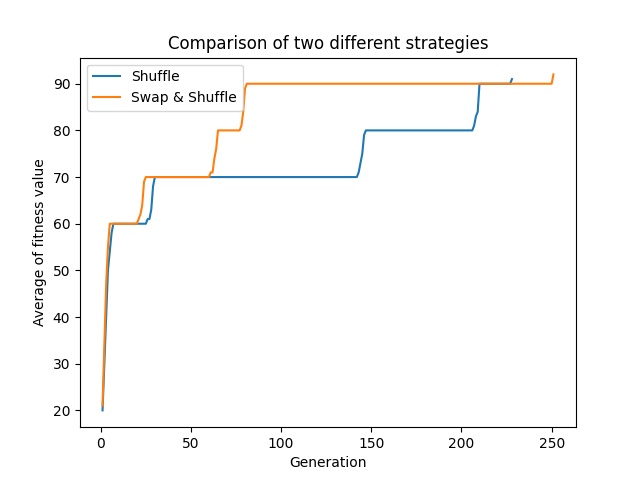
(2)Swap & Shuffle (이후 2번 전략)

- 첫 번째 세대를 생성하고 crossover하여 자식(offspring)을 생성하는 전략은 1번 전략과 동일하다.

- 이 전략에서 mutation 전략은 각 cell마다 순서대로 무작위로 다른 셀(mutation point)를 지정하여 바꾼 뒤(swap), 그 바뀌고 원래 index에 있던 값을 무작위로 바꾸는 전략이다(shuffle). 이 과정은 각 자식 solution에서 0번째부터 9번째 index까지 각각 0.5%의 확률로 일어나게 설정을 해두었다. 예를 들어 2번째 index의 요소의 차례에서 mutation이 일어나면 무작위로 6번째 index에 있는 값과 swap이 되고, swap이 일어난 상태에서 2번째 index에 있는 값이 무작위로 바뀌는(shuffle) 것이다.

- 이후에 다음 세대의 population을 selection하는 방식은 1번 전략과 동일한 방법을 사용하였다.

처음 이 두 개의 전략을 생각했을 때 mutation이 일어날 확률이 매우 낮고 1번에 비해 상대적으로 2번이 exploitation보다는 exploration에 치중을 했다고 생각하여 local-maxima에 빠지면 2번이 1번에 비해 조금 더 빨리 탈출하고 정답도 더 빠르게 찾아갈 것이라고 예상했다.



위의 그림은 generation 마다 각각의 평균 fitness value를 그래프로 표현한 것이다.

하지만 위의 두 전략을 구현하고 실행시켰을 때의 결과는 예상과 비슷한 부분도 있고 다른 부분도 있었다. 두 전략을 계속해서 실행시켜 그래프를 비교해봤을 때, 각각의 전략에서 정답을 찾기 위해 실행된 generation의 범위가 최저 10~20번에서 최대 400~500번 정도로 각각 실행시켰을 때 총 generation의 차이가 크게 나오는 경우도 있었지만 여러 차례 실행시켰을 때 범위가 대체로 비슷해서 결국 예상과는 달리 두 전략이 각각 정답을 찾는 속도는 비슷한 것으로 보인다.

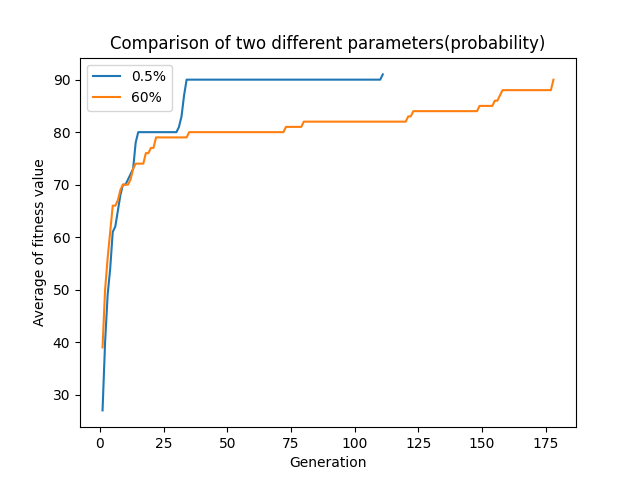
하지만 local-maxima의 경우에서 비교를 해봤을 때에는 나의 예상과 비슷하게 결과가 나왔다. 예상대로 2번 전략이 1번에 비해 exploration에 조금 더 치중한 전략이다 보니 local-maxima에서 탈출하는 속도가 빠르고 더 빨리 빠져 나온 만큼 local-maxima에 빠지는 횟수가 1번에 비해 조금 더 많았다.

2. 1번에서 사용한 (1)Shuffle 전략에서 mutation이 발생할 확률을 다르게 함으로서 parameter을 다르게 하였다.

-하나는 그대로 mutation이 발생할 확률을 0.5%로 설정해두었고 눈에 띄는 차이를 보기 위해 극단적으로 확률을 60%로 올렸다.

-0.5% 설정해 둔 것을 편하게 1번이라고 부르고 60%로 설정해 둔 것을 2번이라고 부르겠다.

처음 parameter을 이렇게 다르게 설정해두었을 때 2번은 매우 exploration에 치중하여서 1번에 비해서 정답을 매우 느리게 찾는 대신 local-maxima에 빠졌을 때 더 빠르게 탈출할 것이라고 예상하였다.



각각의 상황에서 전략(strategy)자체는 같지만 mutation이 일어날 확률을 다르게 하여서 다른 형태의 그래프가 나왔다.

각각의 상황을 여러 번 실행한 결과 예상과 비슷하게 결과가 나왔다. 대신 의외로 2번이 생각보다 더 빠르게 정답을 찾는 경우도 있었다. 진행된 generation의 범위도 비슷했지만 평균적으로 진행된 generation들을 생각해봤을 때에는 확실히 차이가 났다. 평균적으로 1번은 200번 밑으로 나오고 400번 이상 나오는 횟수가 상당히 적었지만, 2번은 3,400번 이상 나오는 경우가 1번에 비해 더 많았다.

예상한 것과 비슷하게 local-maxima에서 빠져나오는 속도는 2번이 1번에 비해 조금 더 빨랐지만, local-maxima에 빠지는 횟수만 비교했을 때에는 2번이 1번보다 더 많이 빠졌다.

따라서 mutation의 확률을 높이면 local-maxima에서 빠져나오는 것은 쉽지만 평균 fitness value값이 증가하는 속도도 더디고 정답을 찾는 속도도 확률이 낮은 것에 비해 상대적으로 좀 더 느린 것을 알 수 있다.