GENETIC EVOLUTIONARY AND LEARNING ALGORITHMS ASSIGNMENT 1#

Ram Gopal Srikar Katakam ID:40106010

Three versions of implementation have been implemented to maximize the fitness function. The pseudo-code of the three implementation have been discussed below:

Genetic Algorithm version 1:

```
#population initialization

For i in range(population size):
        population[i].values=random.rand([varmin1,varmin2],[varmax1,varmax2])
        population[i].fitness=fitness function(population[i].values)
        If population[i].fitness > best.fitness:
                Best.fitness=population[i].fitness


for i in range(generation):

   for j in range(population//2):
        #parent selection

      parent1,parent2=random selection(population)

        #offspring generation
      child1,child2=crossover(parent1,parent2)
      child1_mutate=mutate(child1)
      child2_mutate=mutate(child2)
      child1_mutate,child2_mutate=apply bounds(child1_mutate,child2_mutate)
      child1_mutate.fitness=fitness function(child1_mutate.value)
      child2_mutate.fitness=fitness function(child2_mutate.value)

      if  child1_mutate.fitness>best.fitness:
          best=child1_mutate
      if  child1_mutate.fitness>best.fitness:
          best=child2_mutate

       child population.append(child1_mutate)
       child population.append(child2_mutate)
       parent population.append(parent1)
       parent population.append(parent2)

  #population survival selection
   population=parent population+child population
   population=sorting(population.fintess)
   population=population(population size-1:end)
   average value=average(population.fitness)
```

Genetic Algorithm version 2

```
 For i in range(population size):
          population[i].values=random.rand([varmin1,varmin2],[varmax1,varmax2])
          population[i].fitness=fitness function(population[i].values)
          If population[i].fitness > best.fitness:
                     Best.fitness=population[i].fitness


for i in range(generation):

    for j in range(population size//2):
          #parent selection

        parent1,parent2=fitness based selection (population size, population,cumulative
distribution)

          #offspring generation
        child1,child2=crossover(parent1,parent2,0.2)
        child1_mutate=mutate(child1,0.3)
        child2_mutate=mutate(child2,0.3)
        child1_mutate,child2_mutate=apply bounds(child1_mutate,child2_mutate)
        child1_mutate.fitness=fitness function(child1_mutate.value)
        child2_mutate.fitness=fitness function(child2_mutate.value)

        if  child1_mutate.fitness>best.fitness:
            best=child1_mutate
        if  child1_mutate.fitness>best.fitness:
            best=child2_mutate

         child population.append(child1_mutate)
         child population.append(child2_mutate)
         parent population.append(parent1)
         parent population.append(parent2)

   #population survival selection
    population=parent population+child population
    population=sorting(population.fintess)
    population=population(population size-1:end)
    average value=average(population.fitness)

Function fitness based selection (population size,population,cumulative distribution value)
   rand1=random value(0,1)
   rand2=random value(0,1)
   for i in range(population size):
      if rand1<cummulative distribution value[i]:
         parent1=population[i]
         break
```

```
   for j in range(population size):
      if rand2<cummulative distribution value[i]:
         parent2=population[j]
         break
   return parent1,parent2
```

```
Function cross over(parent1,parent2,gamma=0):
        Alpha=uniform random distribution(-gamma,1+gamma)
        Child1=parent1*alpha+(1-alpha)*parent2
        Child2=parent2*alpha+(1-alpha)*parent1
        Return child1,child2
```

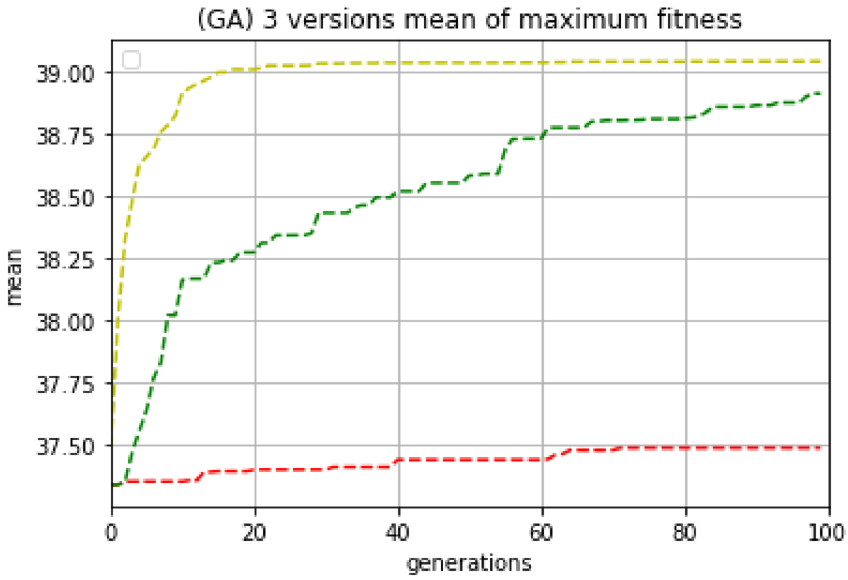
```
Function mutation(child1,mutation rate)
        Mutated child=child1+mutation rate* random normal distribution(1)
        Return mutated child
```

------GA version 3---

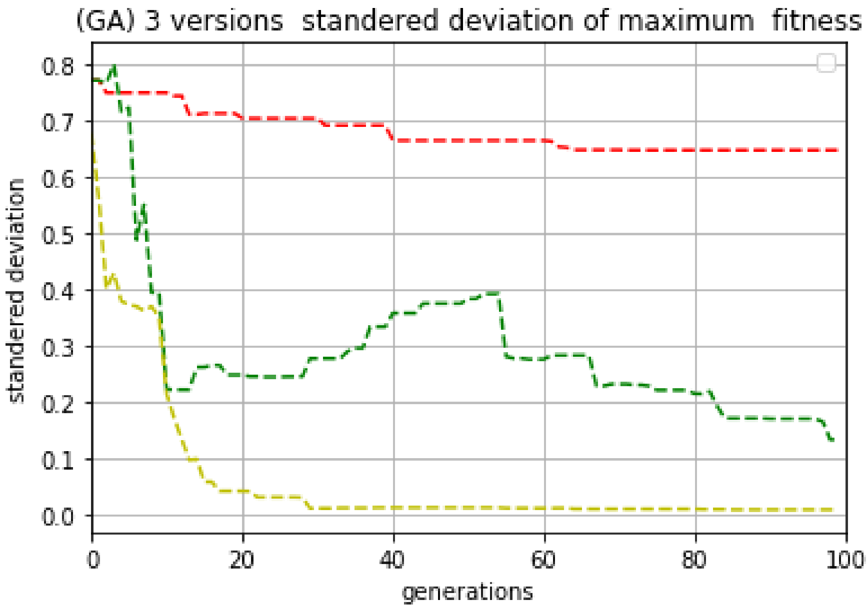
The difference between GA2 and GA3 is parent selection technique used:
Hence pseudo code of parent selection used

```
Function Random selection tournament and fitness based(population size,
population,cumulative distribution):

        Selector=uniform random distribution(0,1)
        K=no.of members in group
        If selector <threshold
        #tournament based selection
        G1=permutation(population size)
        G2=permuation(population size)
        Group1=G1(0:k)
        Group2=G2(0:k)
         Parent1=maximum(group1)#based on fitness value
        Parent2=maximum(group2)

        Else:
        Parent1,parent2=Fitness based selction(population size,population,cumulative
distribution)

Return parent1,parent2
```

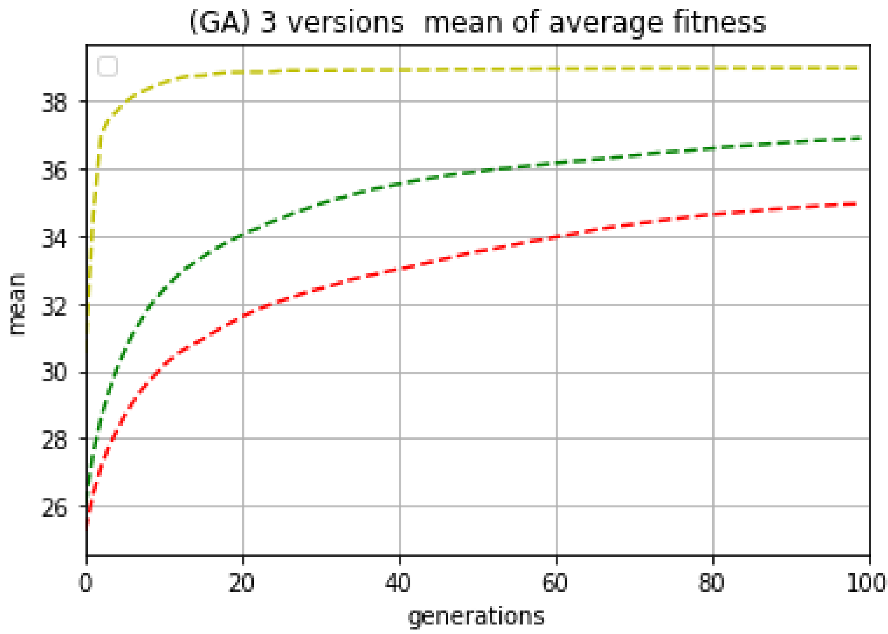
Results:

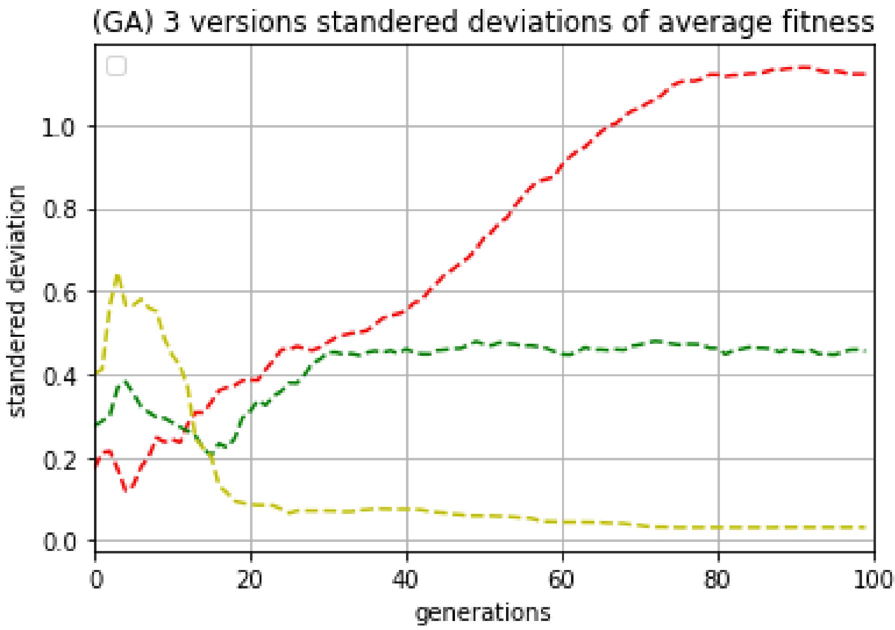


(GA) 3 versions mean of maximum fitness
mean
39.00
38.75
38.50
38.25
38.00
37.75
37.50
0
20
40
60
80
100
generations



(GA) 3 versions standered deviation of maximum fitness
standered deviation
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
20
40
60
80
100
generations





Among the lines shown, the red line corresponds to output of GA1, Green corresponds to GA2 and Yellow corresponds to GA3.

As can be observed GA 3 is outperforming GA2 due to superiority in parent selection, where as GA2 is superior over GA1 due to performance parmeters like mutation rate increase, gamma in cross over and parent selection based on fitness rather than random in nature which GA1 is implemented on.

Due to the inherent flaw in fitness based selection to be trapped in local optima, tournament based selection has been implemented which indeed excels in performance when compared to only fitness based selection, to futher increase the performance the selection between tournamnet and fitness based selection is random in nature. This increase the performance of mean of maximum and average figures as shown above. When stadered deviation is seen due to random nature of GA1 the standard deviation is unusally high compared to GA2 and GA3. The standard deviation is reduced using probablity based selection used in GA2 which

iteration	GA1 maximum value	&	GA1 X value
0	37.96748638315932	&	[-11.65109435 5.91853687]
1	38.27097943834607	&	[11.14339906 -5.92476306]
2	36.8924283763652	&	[11.15618397 -5.12711478]
3	37.007760453665774	&	[-10.62537044 5.21925162]
4	36.334945251963546	&	[-10.61585524 -4.71816008]
5	38.18859787393021	&	[11.63681797 -5.73204317]
6	36.99000441703785	&	[-9.62508605 5.92731119]
7	37.87750669850659	&	[-11.63781128 -5.02135422]
8	37.835209357068806	&	[-11.62240804 4.9204299]
9	37.48885403996839	&	[-11.12412548 4.92753004]

iteration	GA2 maximum value	&	GA2 X value
0	39.01759826006463	&	[11.63118342 5.92449481]
1	38.98344898673817	&	[11.62995176 5.9270911]
2	39.04845877530025	&	[11.62565813 5.92464791]
3	38.876425013481	&	[11.63829312 5.92650185]
4	38.59315608341425	&	[-11.62967029 -5.52699787]
5	38.97023232956454	&	[-11.6296232 5.92739826]
6	39.010319316469214	&	[-11.63208318 5.92529371]
7	38.88576537615887	&	[-11.62721601 -5.82736666]
8	38.83801514728558	&	[-11.62405902 5.72599686]
9	38.90692717967361	&	[-11.63195686 5.82574325]

iteration	GA3 maximum value	GA3 X value
0	39.035663450122925	[-11.62229908 5.92569421]
1	39.042322037526745	[11.6272161 -5.92436273]
2	39.04692419230269	[11.62391525 -5.92532512]
3	39.04872192839223	[-11.62600272 5.92469933]
4	39.03250898758491	[-11.62938858 5.92444195]
5	39.046386834067164	[11.62600859 -5.92447954]
6	39.04787717279859	[11.62714102 -5.92512399]
7	39.049813666673245	[-11.62529791 -5.92485231]
8	39.04690676794988	[-11.625891 -5.92557208]
9	39.02724846876741	[-11.62067353 -5.92537305]

Conclusion:

From the three version implementation, following conclusions can be made:

During implementation of first version, random selection of parents led to high standard deviation due to its inherent random nature and no parameter tunning limiting the performance of finding the maximum value.

During implementation of version two, to decrease the standard deviation, fitness based selection used which reduced irregularity in output values as can be seen from graphs of standard deviation. Fitness based selection has a setback in its implementation which is there is a good probability of the selection method to get stuck in local optima, hence to over come it we try to increase mutation rate parameter and gamma for cross over which could help to algorithm to perform better.

During implementation three, tournament based selection is used which increased the maximum value compared to version 2. To further enhance the algorithm there is random probability of selection between tournament based and fitness based selection which further increased the maximum value attained by the algorithm in minimum gnerations possible.