[2019113013 - Pahulpreet Singh]
[2019115008 - Dhruvee Birla]

Team 70

Report on Genetic Algorithms Project

Summary of the Algorithm

Our algorithm started with first getting an initial population of a definitive size. We then found out the train error, validation error and the total error (= val error + train error) of the population we generated.

After sorting the population based on this error, we assign weights to each member of the population based on their rank after sorting. And based on these weighted probabilities, we select 6 out of these 10 vectors, which would act as parents to participate in the cross-over.

We use these 6 parents to generate 8 children for the next generation. The remaining two members come directly from the parent generation, the vectors with minimum error are chosen. This gives us a population of 10 vectors (2 best parents + 8 children). After going through mutation, this gives rise to the final population of a given generation.

```
pop = get_initial_population(POPULATION_SIZE)

for g in range(NUM_GEN):

     # FITNESS AND SELECTION

    train_err, val_err, total_err = get_population_error(pop)
    sorted_pop = [x for _, x in sorted(zip(total_err, pop), key=lambda pair: pair[0])
    parents = get_parents(sorted_pop, 6)

    # CROSSOVER

    children = get_children(parents, 8)

    new_pop = np.concatenate((sorted_pop[:2], children))

    # MUTATION

    pop = get_mutated_pop(new_pop)
```

Simplified version of the main function

Initial Population

v1	v2	v3	v4	v5	v6	v7	v8	v9	v10
2.50E-23	2.73E-23	2.62E-23	2.42E-23	3.00E-23	2.42E-23	2.31E-23	2.73E-23	3.17E-23	1.86E-23
-3.58E-17	-7.13E-17	-5.68E-17	-9.87E-17	-7.50E-17	-8.94E-17	-7.60E-17	-7.46E-17	-8.65E-17	-7.82E-17
-5.00E-14	-1.37E-14	-2.56E-14	-3.18E-14	-3.98E-14	-2.49E-14	-2.64E-14	-2.39E-14	-2.48E-14	-3.11E-14
-2.90E-16	-2.29E-16	-3.80E-16	-1.70E-16	-2.47E-16	-5.76E-16	-1.55E-16	-2.56E-16	-4.05E-16	-2.01E-16
-2.49E-12	-2.94E-12	-2.31E-12	-1.71E-12	-1.86E-12	-1.69E-12	-4.14E-12	-2.50E-12	-3.09E-12	-1.70E-12
-7.03E-20	-8.00E-20	-9.17E-20	-8.10E-20	-7.51E-20	-2.08E-20	-1.42E-20	-8.31E-20	-1.32E-20	-1.89E-20
1.06E-20	9.57E-21	8.64E-21	6.59E-21	1.18E-20	8.47E-21	1.01E-20	6.59E-21	5.11E-21	6.90E-21
2.84E-10	1.67E-10	1.38E-10	1.59E-10	1.41E-10	1.29E-10	1.78E-10	1.88E-10	1.64E-10	1.57E-10
-5.45E-08	-5.49E-08	-3.61E-08	-3.60E-08	-3.86E-08	-5.58E-08	-4.01E-08	-5.09E-08	-4.54E-08	-3.78E-08
-9.05E-25	-5.36E-25	-7.11E-25	-4.93E-25	-3.33E-25	-6.59E-25	-4.87E-25	-5.96E-25	-4.87E-25	-5.45E-25
2.67E-12	4.61E-12	2.31E-12	2.84E-12	2.54E-12	3.48E-12	4.50E-12	4.02E-12	3.63E-12	3.44E-12

Selection

PASSED ON PASSED ON

Crossover

v1	v6	v10 x v2	v5 x v6	v2 x v9	v6 x v10	v2 x v9	v5 x v8	v10 x v8	v5 x v10
2.50E-23	2.42E-23	1.86E-23	3.00E-23	2.73E-23	2.42E-23	2.73E-23	3.00E-23	1.86E-23	3.00E-23
-3.58E-17	-8.94E-17	-7.13E-17	-8.94E-17	-8.65E-17	-7.82E-17	-8.65E-17	-7.50E-17	-7.82E-17	-7.82E-17
-5.00E-14	-2.49E-14	-1.37E-14	-2.49E-14	-2.48E-14	-2.49E-14	-1.37E-14	-2.39E-14	-3.11E-14	-3.11E-14
-2.90E-16	-5.76E-16	-2.01E-16	-2.47E-16	-4.05E-16	-2.01E-16	-2.29E-16	-2.47E-16	-2.56E-16	-2.01E-16
-2.49E-12	-1.69E-12	-2.94E-12	-1.69E-12	-2.94E-12	-1.69E-12	-3.09E-12	-1.86E-12	-1.70E-12	-1.70E-12
-7.03E-20	-2.08E-20	-8.00E-20	-2.08E-20	-1.32E-20	-1.89E-20	-1.32E-20	-7.51E-20	-1.89E-20	-7.51E-20
1.06E-20	8.47E-21	9.57E-21	8.47E-21	5.11E-21	6.90E-21	9.57E-21	6.59E-21	6.90E-21	1.18E-20
2.84E-10	1.29E-10	1.67E-10	1.29E-10	1.64E-10	1.29E-10	1.64E-10	1.88E-10	1.88E-10	1.57E-10
-5.45E-08	-5.58E-08	-3.78E-08	-3.86E-08	-5.49E-08	-5.58E-08	-5.49E-08	-5.09E-08	-5.09E-08	-3.78E-08
-9.05E-25	-6.59E-25	-5.36E-25	-6.59E-25	-4.87E-25	-5.45E-25	-4.87E-25	-5.96E-25	-5.45E-25	-3.33E-25
2.67E-12	3.48E-12	4.61E-12	3.48E-12	4.61E-12	3.48E-12	4.61E-12	2.54E-12	3.44E-12	3.44E-12

Mutation

v1*	v2*	v3*	v4*	v5*	v6*	v7*	v8*	v9*	v10*
3.15E-23	**2.23E-23**	**1.42E-23**	**3.17E-23**	**2.28E-23**	**3.03E-23**	**3.29E-23**	3.00E-23	**1.46E-23**	3.00E-23
-2.95E-17	**-8.05E-17**	**-8.60E-17**	**-1.06E-16**	**-6.27E-17**	**-5.61E-17**	**-7.66E-17**	-7.50E-17	**-9.70E-17**	**-8.47E-17**
-4.48E-14	**-2.99E-14**	**-1.56E-14**	-2.65E-14	**-1.91E-14**	**-3.20E-14**	**-1.02E-14**	**-2.87E-14**	**-2.64E-14**	-3.11E-14
-2.69E-16	**-6.88E-16**	-2.26E-16	-1.86E-16	**-3.26E-16**	**-1.60E-16**	-2.29E-16	**-2.71E-16**	**-2.12E-16**	**-1.49E-16**
-2.71E-12	**-1.77E-12**	**-2.22E-12**	**-1.31E-12**	**-2.08E-12**	**-1.92E-12**	**-3.50E-12**	**-1.56E-12**	**-1.49E-12**	**-2.14E-12**
-5.79E-20	**-1.92E-20**	**-8.56E-20**	**-1.66E-20**	**-1.23E-20**	**-1.39E-20**	**-1.56E-20**	**-5.43E-20**	**-2.28E-20**	**-5.37E-20**
1.17E-20	8.47E-21	9.57E-21	**7.92E-21**	**4.66E-21**	**7.44E-21**	**9.57E-21**	**4.81E-21**	6.90E-21	**8.86E-21**
3.42E-10	**1.44E-10**	**2.05E-10**	**1.45E-10**	**1.47E-10**	**1.38E-10**	**1.49E-10**	**2.00E-10**	**2.29E-10**	**1.76E-10**
-4.36E-08	**-4.06E-08**	**-4.32E-08**	-2.86E-08	-5.49E-08	**-4.44E-08**	**-5.49E-08**	**-4.83E-08**	**-4.04E-08**	**-2.78E-08**
-1.09E-24	**-8.38E-25**	**-4.61E-25**	**-4.89E-25**	**-5.33E-25**	**-4.30E-25**	**-5.45E-25**	**-4.28E-25**	-5.45E-25	**-2.80E-25**
3.18E-12	**2.51E-12**	**3.43E-12**	**3.06E-12**	**4.11E-12**	**3.24E-12**	**5.16E-12**	**3.03E-12**	**3.66E-12**	**4.38E-12**

Generation - 1

Initial Population

v1	v2	v3	v4	v5	v6	v7	v8	v9	v10
3.15E-23	2.23E-23	1.42E-23	3.17E-23	2.28E-23	3.03E-23	3.29E-23	3.00E-23	1.46E-23	3.00E-23
-2.95E-17	-8.05E-17	-8.60E-17	-1.06E-16	-6.27E-17	-5.61E-17	-7.66E-17	-7.50E-17	-9.70E-17	-8.47E-17
-4.48E-14	-2.99E-14	-1.56E-14	-2.65E-14	-1.91E-14	-3.20E-14	-1.02E-14	-2.87E-14	-2.64E-14	-3.11E-14
-2.69E-16	-6.88E-16	-2.26E-16	-1.86E-16	-3.26E-16	-1.60E-16	-2.29E-16	-2.71E-16	-2.12E-16	-1.49E-16
-2.71E-12	-1.77E-12	-2.22E-12	-1.31E-12	-2.08E-12	-1.92E-12	-3.50E-12	-1.56E-12	-1.49E-12	-2.14E-12
-5.79E-20	-1.92E-20	-8.56E-20	-1.66E-20	-1.23E-20	-1.39E-20	-1.56E-20	-5.43E-20	-2.28E-20	-5.37E-20
1.17E-20	8.47E-21	9.57E-21	7.92E-21	4.66E-21	7.44E-21	9.57E-21	4.81E-21	6.90E-21	8.86E-21
3.42E-10	1.44E-10	2.05E-10	1.45E-10	1.47E-10	1.38E-10	1.49E-10	2.00E-10	2.29E-10	1.76E-10
-4.36E-08	-4.06E-08	-4.32E-08	-2.86E-08	-5.49E-08	-4.44E-08	-5.49E-08	-4.83E-08	-4.04E-08	-2.78E-08
-1.09E-24	-8.38E-25	-4.61E-25	-4.89E-25	-5.33E-25	-4.30E-25	-5.45E-25	-4.28E-25	-5.45E-25	-2.80E-25
3.18E-12	2.51E-12	3.43E-12	3.06E-12	4.11E-12	3.24E-12	5.16E-12	3.03E-12	3.66E-12	4.38E-12

Selection

				PASSED ON		PASSED ON			

Crossover

v5	v7	v7 x v6	v9 x v2	v5 x v9	v2 x v9	v9 x v2	v7 x v6	v2 x v5	v1 x v9
2.28E-23	3.29E-23	3.29E-23	1.46E-23	2.28E-23	2.23E-23	1.46E-23	3.29E-23	2.23E-23	3.15E-23
-6.27E-17	-7.66E-17	-7.66E-17	-9.70E-17	-6.27E-17	-8.05E-17	-9.70E-17	-7.66E-17	-6.27E-17	-9.70E-17
-1.91E-14	-1.02E-14	-3.20E-14	-2.99E-14	-1.91E-14	-2.99E-14	-2.64E-14	-3.20E-14	-1.91E-14	-2.64E-14
-3.26E-16	-2.29E-16	-1.60E-16	-6.88E-16	-2.12E-16	-2.12E-16	-6.88E-16	-2.29E-16	-3.26E-16	-2.69E-16
-2.08E-12	-3.50E-12	-1.92E-12	-1.49E-12	-1.49E-12	-1.77E-12	-1.77E-12	-1.92E-12	-2.08E-12	-2.71E-12
-1.23E-20	-1.56E-20	-1.39E-20	-2.28E-20	-1.23E-20	-1.92E-20	-1.92E-20	-1.56E-20	-1.92E-20	-2.28E-20
4.66E-21	9.57E-21	7.44E-21	8.47E-21	4.66E-21	8.47E-21	6.90E-21	9.57E-21	8.47E-21	6.90E-21
1.47E-10	1.49E-10	1.49E-10	1.44E-10	1.47E-10	1.44E-10	2.29E-10	1.49E-10	1.47E-10	3.42E-10
-5.49E-08	-5.49E-08	-4.44E-08	-4.06E-08	-5.49E-08	-4.04E-08	-4.04E-08	-4.44E-08	-5.49E-08	-4.36E-08
-5.33E-25	-5.45E-25	-4.30E-25	-5.45E-25	-5.33E-25	-5.45E-25	-5.45E-25	-5.45E-25	-8.38E-25	-1.09E-24
4.11E-12	5.16E-12	5.16E-12	2.51E-12	3.66E-12	2.51E-12	2.51E-12	3.24E-12	4.11E-12	3.66E-12

Mutation

v1*	v2*	v3*	v4*	v5*	v6*	v7*	v8*	v9*	v10*
2.95E-23	**4.03E-23**	**3.88E-23**	**1.54E-23**	**1.97E-23**	**1.70E-23**	1.46E-23	**3.84E-23**	**2.70E-23**	**2.91E-23**
-4.44E-17	**-9.58E-17**	**-9.11E-17**	**-8.32E-17**	**-6.87E-17**	**-6.62E-17**	**-6.82E-17**	**-5.89E-17**	**-7.35E-17**	**-1.04E-16**
-2.18E-14	**-1.31E-14**	**-2.61E-14**	-2.99E-14	**-2.37E-14**	-2.99E-14	**-3.04E-14**	**-2.57E-14**	**-1.34E-14**	**-2.42E-14**
-2.34E-16	**-2.81E-16**	-1.60E-16	**-7.36E-16**	**-2.50E-16**	**-1.63E-16**	**-7.46E-16**	**-2.67E-16**	**-2.85E-16**	**-3.22E-16**
-1.75E-12	**-3.20E-12**	**-1.66E-12**	**-1.18E-12**	**-1.64E-12**	**-1.45E-12**	**-2.13E-12**	**-1.73E-12**	**-2.65E-12**	**-3.45E-12**
-1.39E-20	**-1.20E-20**	**-1.79E-20**	**-1.60E-20**	**-1.47E-20**	**-1.74E-20**	**-2.37E-20**	**-1.26E-20**	**-2.10E-20**	**-2.09E-20**
3.64E-21	**8.06E-21**	**6.03E-21**	**9.88E-21**	**5.94E-21**	**1.04E-20**	**8.09E-21**	**1.11E-20**	**6.38E-21**	**8.78E-21**
1.90E-10	**1.09E-10**	**1.76E-10**	**1.15E-10**	**1.10E-10**	**1.74E-10**	**2.87E-10**	**1.91E-10**	**1.71E-10**	**3.03E-10**
-5.86E-08	**-6.00E-08**	**-3.71E-08**	**-3.84E-08**	**-4.64E-08**	**-4.62E-08**	**-3.67E-08**	**-5.71E-08**	-5.49E-08	**-5.22E-08**
-3.89E-25	**-6.44E-25**	**-3.11E-25**	**-4.82E-25**	**-5.71E-25**	**-6.74E-25**	**-5.99E-25**	**-4.02E-25**	**-9.79E-25**	**-1.16E-24**
4.11E-12	**5.55E-12**	**4.23E-12**	**2.24E-12**	**4.00E-12**	**2.10E-12**	**2.17E-12**	**3.52E-12**	**3.23E-12**	**3.85E-12**

Generation - 2

Initial Population

v1	v2	v3	v4	v5	v6	v7	v8	v9	v10
2.95E-23	4.03E-23	3.88E-23	1.54E-23	1.97E-23	1.70E-23	1.46E-23	3.84E-23	2.70E-23	2.91E-23
-4.44E-17	-9.58E-17	-9.11E-17	-8.32E-17	-6.87E-17	-6.62E-17	-6.82E-17	-5.89E-17	-7.35E-17	-1.04E-16
-2.18E-14	-1.31E-14	-2.61E-14	-2.99E-14	-2.37E-14	-2.99E-14	-3.04E-14	-2.57E-14	-1.34E-14	-2.42E-14
-2.34E-16	-2.81E-16	-1.60E-16	-7.36E-16	-2.50E-16	-1.63E-16	-7.46E-16	-2.67E-16	-2.85E-16	-3.22E-16
-1.75E-12	-3.20E-12	-1.66E-12	-1.18E-12	-1.64E-12	-1.45E-12	-2.13E-12	-1.73E-12	-2.65E-12	-3.45E-12
-1.39E-20	-1.20E-20	-1.79E-20	-1.60E-20	-1.47E-20	-1.74E-20	-2.37E-20	-1.26E-20	-2.10E-20	-2.09E-20
3.64E-21	8.06E-21	6.03E-21	9.88E-21	5.94E-21	1.04E-20	8.09E-21	1.11E-20	6.38E-21	8.78E-21
1.90E-10	1.09E-10	1.76E-10	1.15E-10	1.10E-10	1.74E-10	2.87E-10	1.91E-10	1.71E-10	3.03E-10
-5.86E-08	-6.00E-08	-3.71E-08	-3.84E-08	-4.64E-08	-4.62E-08	-3.67E-08	-5.71E-08	-5.49E-08	-5.22E-08
-3.89E-25	-6.44E-25	-3.11E-25	-4.82E-25	-5.71E-25	-6.74E-25	-5.99E-25	-4.02E-25	-9.79E-25	-1.16E-24
4.11E-12	5.55E-12	4.23E-12	2.24E-12	4.00E-12	2.10E-12	2.17E-12	3.52E-12	3.23E-12	3.85E-12

Selection — **PASSED ON** — **PASSED ON**

Crossover

v1	v8	v4 x v9	v7 x v9	v9 x v5	v5 x v10	v2 x v9	v10 x v5	v5 x v10	v2 x v10
2.95E-23	3.84E-23	1.54E-23	1.46E-23	2.70E-23	1.97E-23	4.03E-23	2.91E-23	1.97E-23	4.03E-23
-4.44E-17	-5.89E-17	-7.35E-17	-6.82E-17	-6.87E-17	-6.87E-17	-9.58E-17	-1.04E-16	-6.87E-17	-1.04E-16
-2.18E-14	-2.57E-14	-1.34E-14	-1.34E-14	-1.34E-14	-2.42E-14	-1.31E-14	-2.37E-14	-2.37E-14	-2.42E-14
-2.34E-16	-2.67E-16	-7.36E-16	-2.85E-16	-2.85E-16	-3.22E-16	-2.81E-16	-2.50E-16	-3.22E-16	-2.81E-16
-1.75E-12	-1.73E-12	-2.65E-12	-2.13E-12	-1.64E-12	-3.45E-12	-2.65E-12	-3.45E-12	-3.45E-12	-3.45E-12
-1.39E-20	-1.26E-20	-1.60E-20	-2.10E-20	-1.47E-20	-2.09E-20	-1.20E-20	-2.09E-20	-1.47E-20	-2.09E-20
3.64E-21	1.11E-20	6.38E-21	8.09E-21	5.94E-21	8.78E-21	8.06E-21	5.94E-21	8.78E-21	8.78E-21
1.90E-10	1.91E-10	1.15E-10	1.71E-10	1.10E-10	3.03E-10	1.71E-10	1.10E-10	3.03E-10	3.03E-10
-5.86E-08	-5.71E-08	-5.49E-08	-3.67E-08	-4.64E-08	-5.22E-08	-5.49E-08	-5.22E-08	-5.22E-08	-6.00E-08
-3.89E-25	-4.02E-25	-9.79E-25	-5.99E-25	-5.71E-25	-5.71E-25	-9.79E-25	-5.71E-25	-1.16E-24	-1.16E-24
4.11E-12	3.52E-12	2.24E-12	3.23E-12	3.23E-12	3.85E-12	3.23E-12	4.00E-12	4.00E-12	3.85E-12

Mutation

v1*	v2*	v3*	v4*	v5*	v6*	v7*	v8*	v9*	v10*
3.63E-23	3.84E-23	1.54E-23	**1.08E-23**	**1.89E-23**	**2.20E-23**	**4.56E-23**	**2.55E-23**	**1.62E-23**	**2.94E-23**
-3.43E-17	**-6.28E-17**	**-6.09E-17**	**-4.87E-17**	**-5.91E-17**	**-5.28E-17**	**-1.02E-16**	**-1.27E-16**	**-8.53E-17**	**-1.20E-16**
-1.68E-14	**-3.26E-14**	**-1.01E-14**	**-1.53E-14**	**-1.47E-14**	**-1.73E-14**	**-1.50E-14**	**-2.72E-14**	**-2.37E-14**	**-3.03E-14**
-2.34E-16	**-3.37E-16**	**-5.18E-16**	**-2.20E-16**	**-3.10E-16**	**-3.81E-16**	**-3.13E-16**	**-3.23E-16**	**-2.52E-16**	-2.81E-16
-1.75E-12	**-1.87E-12**	**-2.34E-12**	**-1.74E-12**	**-1.22E-12**	**-2.94E-12**	**-2.98E-12**	**-3.92E-12**	**-2.68E-12**	-3.45E-12
-1.59E-20	**-1.63E-20**	**-1.83E-20**	**-1.99E-20**	**-1.30E-20**	**-2.21E-20**	**-8.63E-21**	**-1.94E-20**	**-1.91E-20**	-2.09E-20
3.64E-21	**7.92E-21**	**8.04E-21**	**8.71E-21**	5.94E-21	**7.90E-21**	**9.85E-21**	**6.63E-21**	**7.23E-21**	**6.86E-21**
1.64E-10	**2.36E-10**	1.15E-10	**1.26E-10**	**1.29E-10**	**2.48E-10**	**2.17E-10**	**1.24E-10**	**3.03E-10**	**3.78E-10**
-7.14E-08	**-5.07E-08**	**-4.34E-08**	**-4.71E-08**	**-4.19E-08**	**-4.72E-08**	**-6.20E-08**	**-4.72E-08**	**-4.07E-08**	**-7.74E-08**
-3.00E-25	**-3.38E-25**	**-1.04E-24**	-5.99E-25	**-5.06E-25**	**-4.63E-25**	**-1.26E-24**	**-4.25E-25**	**-9.70E-25**	**-9.24E-25**
3.76E-12	**4.45E-12**	**1.63E-12**	**3.43E-12**	**2.61E-12**	**3.48E-12**	**2.66E-12**	**3.18E-12**	**4.49E-12**	**4.12E-12**

Generation - 3

Fitness Function

First we get the total error of each vector of the population. We then arrange them in ascending order of errors and assign probabilities in a descending order. (So lower the error, higher the rank in the population, higher the probability to get selected as a parent). These parents are then used for cross-over.

```
def get_probs(num):
    prob = np.ndarray(num)
    for i in range(num):
        prob[i] = num - i*0.5
    prob /= np.sum(prob)
    return prob

def get_parents(pop, num):
    prob = get_probs(len(pop))
    indices = np.random.choice(np.arange(0, len(pop)), num, False, prob)
    parents = []
    for i in indices:
        parents.append(pop[i])
    return parents
```

Fitness and Selection functions

Crossover Function

In this function, we pass two parameters which are two parents. Then on a 50% probability, either parent 1 is selected or parent2 is selected. For a particular element in the vector, the corresponding element in the child vector either becomes the same element as that of parent1 or of parent2 depending on the randomly generated integer. Then we run this function for the entire population.

```python
def crossover(p1, p2):
    child = np.ndarray(MAX_DEG)
    for i in range(MAX_DEG):
        if random.getrandbits(1):
            child[i] = p1[i]
        else:
            child[i] = p2[i]
    return child
```

Crossover function for parents p1 and p2

Mutation Function

For mutation, first we decide whether or not to mutate a given chromosome by a certain probability. Then we decide the mutation factor by generating a random number between 1-m% and 1+m% (here m% is the maximum mutation percentage) and then we multiply the numbers with this factor. (We are not using a set mutation percentage, a different factor is generated randomly each time a chromosome is mutated. m% just denotes the upper/lower limit of mutation)

```python
def mutation(member, prob, pc):
    mem = member
    for i in range(len(mem)):
        if np.random.random_sample() <= prob:
                factor = np.random.uniform(1 - pc, 1 + pc)
                mem[i] *= factor
    return mem
```

Simplified version of the Mutation Function

We kept changing the mutation probability and the mutation percentage throughout the project. Initially we kept these numbers low, but when we realised that we hit a local minima and our populations started to converge but with high errors, we increased these numbers. And once we started getting better results, we again changed the mutation probability because we thought we were getting closer to the results so we don't want too many changes in the vectors.

Hyperparameters

We chose our pool size to be 10 since less than 10 would be too small of a sample space and would not be enough data for observations and conclusions whereas more than 10 seemed like a lot of generations keeping in mind the limited number of API calls. Therefore, a pool size of 10 seemed like a reasonable amount to iterate through maximum generations within the given API calls.

We chose six parents from every generation and created eight children from those parents. The next generation included those eight children and two parents with the least total error. This way of choosing our splitting point for creating new genes ensures that a variety of vectors is introduced in every generation. This variety takes place due to the selection of six parents. Selecting less than six parents might mean less variation or no variation at all whereas selecting all the ten parents will lead to poor crossover causing the algorithm to slow down by producing children complements of each other and causing the population to converge.

In other words, there is a trade-off between the number of parents selected and variety. More selected parents leads to bad crossover and hence low variety whereas less selected parents leads to more variety. But since we do not want a lot of variation and want the generations to produce more or less similar errors, there should be a good number of selected parents for enough variation.

Statistical Information

Around hundred iterations after the first vector was introduced, convergence was observed. Our errors kept increasing and remained high for a number of future iterations therefore, we believed that we had reached a local minima and started over by writing a new random population generator function which turned out to be better. The reason why we think our new function is better than our previous one is because, with our first convergence, the errors were of the order 10^{16} but around 100-150 iterations after starting over with the new function the errors are now of the order 10^{11} - 10^{12}.

Heuristics

The very first population generated gave errors of order 10^{40} - 10^{42} which after a lot of iterations were reduced to the order 10^{20}. Since this error took many iterators to get better, we decided to start over. We wrote a new population generator function and used all the ten parents from the

new population after starting over thus this led to convergence since by using all the parents, pairs of children were complements of each other implying the randomness/variation went down. This led to an infinite loop.

Therefore, we changed the mutation function by ensuring there is a certain amount of mutation but this led to convergence as well. Hence, we started with a new population again with a new mutation function.

This new mutation function with the new population led to less errors and therefore, we decided to make this our final algorithm. With this method, after around hundred generations, we reached a population with the best errors. We handpicked the ten best vectors for mid-evals which is now our generation 0. The descendants of this generation, that is, the ten handpicked vectors, became our future generations that we worked on.

Train and validation error

In the beginning, the train and validation error had a magnitude difference of around 100. And with our present method (changed mutation and population generator function) the validation error is around $4*10^{11}$ - $6*10^{11}$ and the train error is around $5*10^{11}$ - $7*10^{11}$.

We picked vectors with almost the same train error and validation error therefore even on unseen data we expected equal performance of train error and validation error. We did not choose vectors with even less errors because this would lead to overfit vectors and on unseen data, that would mean poor performance.

Miscellaneous

After every few generations, we checked out our errors for the same. If those errors were worse than the previous generation we would manually replace a vector in the population with that of some previous generation which was giving lesser error. We reached a population which gave us the best errors and used it to create future generations. From those generations, we looked for specific vectors giving lesser errors and used those specific vectors to create a new population and made that population our starting population.

For mid-evals, we handpicked and submitted the 10 best vectors in terms of total error and created a new progeny using that. We used this as our initial population (Generation 0), carried this progeny forward, created the future generations and worked with those.