3P71 Assignment 2

Parker TenBroeck 7376726
pt21zs@brocku.ca

I. Introduction

This document shows the performance of a GA system designed to solve the scheduling problem outlined in the assignment. The solution implementation will be tested with varying parameters and problem sets to show what configurations product the best results and how they compare to each other.

A. The Problem

A university has been tasked with creating a program to assign courses which have professors teaching them to rooms and timeslots while satisfying multiple constraints. We know that a course must be scheduled in a room with space for at least the number of students in that course, if not more. A room which is booked at a certain timeslot for one course cannot be scheduled any other course at that timeslot. And finally a professor traching one course cannot be scheduled to teach another course(or the same course) at the same time. It has been decided that a genetic algorithm is the best method for solving this problem.

II. GA Overview

A GA system is a way to represent and solve particular problems by representing potential solutions as chromosomes, who we call individuals. These chromosomes are made of genes which represent individual parts of a potential solution which we can change. Individuals exist in a population where each new generation we apply algorithms to better our potential solutions, hoping to converge on a satisfactory answer [1, p. 3].

In A GA system there are five important algorithms to consider, Fitness, Elitism, Selection, Crossover, and Mutation [1, p. 6].

Fitness determines how individuals are evaluated, and gives us a ranking of "best" to "worst". We can also use fitness to help determine if a particular individual is a satisfactory solution.

Elitism determines how many of the top k members of the population will be guaranteed in the next population. These individuals won't be mutated in any way [1, p. 7].

Selection determines how members of a previous population are selected for the new population. Selection happens after we copy the elites from the previous population filling the remaining population with individuals decided by selection [1, p. 9].

Crossover determines how two individuals will 'breed' that is, how two individuals genes are combined to create two children whos genes are some combination of their parents. There is a associated percentage for crossover that determines the chance any given chromosome will have individual applied to it [1, p. 12].

Finally mutation determines how chromosomes are mutated, specifically how individual genes are mutated inside any particular chromosome. mutation also has a probability associated with it which is the percent change a given individual will undergo mutation.

The GA implemented was based off these slides [1].

A. Fitness

Fitness determines how "fit" any individual is, that is gives us a numeric way of raking individuals against each other. In our case our fitness is determined by the reciprocal of a weighed sum over the number of different conflicts present in a schedule

$$C = \text{Over room capacity}$$
$$R = \text{Room overbooked in timeslot}$$
$$P = \text{Professor overbooked in timeslot}$$
$$\text{raw} = 3C + 2R + P \tag{1}$$
$$\text{normalized} = \frac{1}{1 + \text{raw}} \tag{2}$$

B. Selection

In this GA there is only a single algorithm for selection. It is K Tournament selection. For every remaining slot in the population that wasn't filled by elitism the selection algorithm runs. The algorithm simply selects K individuals from the previous population and gives the best individual of those K. in our case $K = 4$ so 4 individuals are selected from the old population, and the best of the 4 in terms of fitness is selected for that slot. This is repeated until every slot in the new population is filled.

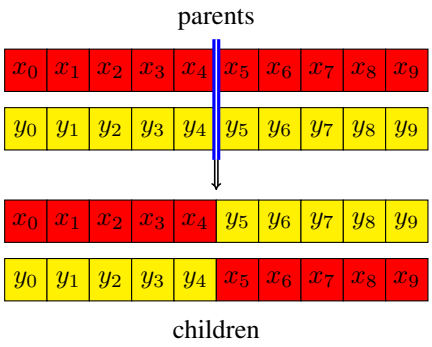
C. Mutation

Similarly to selection there is only one algorithm for mutation. The algorithm simply chooses a random gene, and replaces it with a new random, but possible gene. Mutation is attempted on all individuals in a population except elitism individuals. for every individual in a generation there is a percent chance mutation happen, this is determined by the mutation rate parameter.

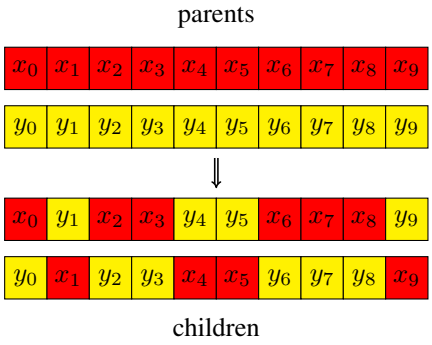
D. Crossover

Crossover can occurs between two individuals that were selected in the selection step. how often crossover happens is determined by the crossover rate parameter. If two individuals aren't selected then they are directly placed in the population. If they are then the specified crossover algorithm is ran on them and their resulting children are placed in the new population. Crossover does not mutate the elitism individuals. There are three available crossover algorithms, two are from class, one is a custom crossover operator designed specifically for this problem.

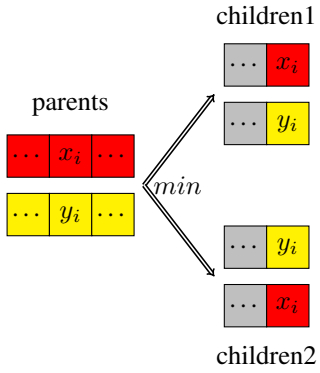
1) One Point Crossover: This crossover works by selecting a random index i and swapping all genes after i [1, p. 17].



2) Uniform Crossover: This crossover randomly swaps genes at any given point. Each swap has an equal 50% chance of occurring [1, p. 21-29].



3) Best Attempt Crossover: This crossover attempts to minimize the number of conflicts per gene swap. It starts with the first gene and at every gene i it chooses the configuration that has the least conflicts. If the number of conflicts are the same there is a 50% chance of a particular pair of genes swapping or not. This continues for all genes in the chromosome.



This method was specifically implemented for this problem as it is fairly trivial to test which set of children will have a better fitness at every step allowing us to take a greedy approach that gives significant gains. In the cases where the locally optimal choice isn't the overall optimal choice mutation and genetic diversity keeps the GA system from getting stuck in a local maximum.

III. Setup

A. Compilation

In the provided code there is a shell script `build.sh` which when ran compiles all the code and creates a jar called `assign2.jar`.

B. Running

Running the program can be done with `java -jar assign2.jar` and configuring the CLI args. `<num>` is a integer, `<rate>` is a percentage value from 0-1. The problem set and generations number must be given, and the seeds must be set using one of the methods below.

`--problem-set <path>`: the path to the problem set
`--seeds-linear <num>`: sets the seeds to 0 to <num>-1
`--seeds-rand <num>`: creates <num> random seeds
`--seeds +<num,>`: a comma separated list of seeds
`--generations <num>`: sets the max number of generations
`--gui`: show the graph GUI.

The following parameters will run tests for all permutations of the provided values. These all must have at least one value

`--elitism-rates +<num,>`: the elitism rates to test
`--crossover-rates +<rate,>`: the crossover rates to test
`--mutation-rates +<rate,>`: the mutation rates to test
`--initializer-kinds +<kind,>`: the initializer kinds to test
values: `Random`
`--fitness-kinds +<kind,>`: the fitness kinds to test
values: `WeightedConflicts`
`--mutation-kinds +<kind,>`: the mutation kinds to test
values: `SingleGene`
`--crossover-kinds +<kind,>`: the crossover kinds to test
values: `OnePoint, Uniform, BestAttempt`

There is a shell script `run_defaults.sh` which runs all of the tests ran in this document. All you provide is the path to the problem set, and optionally `--gui` if you wish to display the grahps

the results for each test will be outputed into the directory `runs` containing a file starting at `run1.json` through `run#.json` which contains all the data collected for every run. **Important** the directory `runs` in the CWD will be deleted including its contents upon running the program.

IV. Results

In all these tests the raw fitness(solid lines) is defined as (1), the closer to zero the better. And the normalized fitness(dotted lines) is defined as (2), the closer to 100 the better. Each graph shows the max/average/min of the raw and normalized fitness. Because each configuration is ran 50 times with different seeds we take the average over all runs for every generation. If a run finishes before another, the final values it produced are used as the value for all further generations. A vertical line is shown when a particular run finishes with its generation number, seed number and final normalized fitness value.

The parameters not listed for each test don't change and are as follows

elitism rate: 1 individual[1].
selection: $k = 4$ tournament selection.
seeds: 0-49.
fitness: weighted sum.
mutation: random gene mutation.
initialization: random well formed genes.

In Fig. 2 and Fig. 4 we can see that all 50 runs in nearly all configurations the GA was able to find a solution in 250 generations. That means comparing on fitness will not lead to any insights on which set of parameters are better. However, we can use the average last generation as a metric as seen in Fig. 3 and Fig. 5. This will let us compare which configurations converge on an answer in fewer generations and give us more insight into how different parameters perform against each other.

In Fig. 1 shows which config corresponds to which set of parameters.

[1]talked to Reginald and said I could use this value as I had already ran my tests using this parameter before he made the announcement

	crossover rate	mutation rate	crossover method
config1	90%	0%	OnePoint
config2	90%	10%	OnePoint
config3	100%	0%	OnePoint
config4	100%	10%	OnePoint
config5	90%	0%	Uniform
config6	90%	10%	Uniform
config7	100%	0%	Uniform
config8	100%	10%	Uniform
config9	90%	0%	BestAttempt
config10	90%	1%	BestAttempt
config11	90%	10%	BestAttempt
config12	100%	0%	BestAttempt
config13	100%	1%	BestAttempt
config14	100%	10%	BestAttempt

Fig. 1. run parameters

	min	max	mean	median	σ
config1	25	100	89.83	100.0	22.066
config2	100	100	100.00	100.0	0.000
config3	33	100	98.67	100.0	9.333
config4	100	100	100.00	100.0	0.000
config5	100	100	100.00	100.0	0.000
config6	100	100	100.00	100.0	0.000
config7	100	100	100.00	100.0	0.000
config8	100	100	100.00	100.0	0.000
config9	100	100	100.00	100.0	0.000
config10	100	100	100.00	100.0	0.000
config11	100	100	100.00	100.0	0.000
config12	100	100	100.00	100.0	0.000
config13	100	100	100.00	100.0	0.000
config14	100	100	100.00	100.0	0.000

Fig. 2. Test data 1 solution fitness results

	min	max	mean	median	σ
config1	15	250	61.18	20.0	88.490
config2	16	28	20.60	20.0	2.538
config3	15	250	23.60	19.0	32.407
config4	15	28	20.28	20.0	2.623
config5	16	23	19.08	19.0	1.809
config6	15	25	19.78	19.5	2.091
config7	13	22	18.80	19.0	1.833
config8	13	24	19.68	20.0	1.994
config9	8	12	10.32	10.0	0.882
config10	8	12	10.30	10.0	0.943
config11	8	12	10.54	11.0	0.943
config12	9	12	10.16	10.0	0.857
config13	8	11	10.36	11.0	0.843
config14	9	12	10.32	10.0	0.859

Fig. 3. Test data 1 generation completed results

	min	max	mean	median	σ
config1	50	100	98.00	100.0	9.798
config2	100	100	100.00	100.0	0.000
config3	100	100	100.00	100.0	0.000
config4	100	100	100.00	100.0	0.000
config5	100	100	100.00	100.0	0.000
config6	100	100	100.00	100.0	0.000
config7	100	100	100.00	100.0	0.000
config8	100	100	100.00	100.0	0.000
config9	100	100	100.00	100.0	0.000
config10	100	100	100.00	100.0	0.000
config11	100	100	100.00	100.0	0.000
config12	100	100	100.00	100.0	0.000
config13	100	100	100.00	100.0	0.000
config14	100	100	100.00	100.0	0.000

Fig. 4. Test data 2 solution fitness results

	min	max	mean	median	σ
config1	10	250	23.16	14.0	46.324
config2	9	18	14.02	14.0	1.944
config3	10	16	13.14	13.0	1.549
config4	10	16	13.76	14.0	1.556
config5	11	17	13.86	14.0	1.249
config6	11	19	14.02	14.0	1.679
config7	9	17	13.16	13.0	1.666
config8	10	18	13.82	14.0	1.584
config9	6	8	7.04	7.0	0.720
config10	5	8	6.90	7.0	0.728
config11	5	9	7.16	7.0	0.946
config12	5	9	6.64	7.0	0.794
config13	5	8	6.88	7.0	0.840
config14	5	8	6.90	7.0	0.728

Fig. 5. Test data 2 generation completed results

Starting with the worst performing configuration Fig. 7 we can immediately see that 9 of the 50 runs don't complete in the 250 generations.

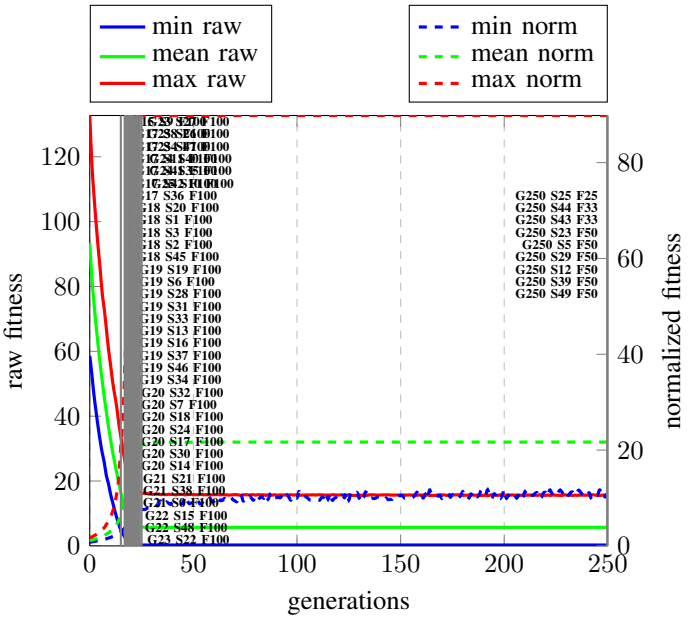


Fig. 6. Test data 1 config 1

We can see Fig. 7 exhibits some of the same behavior as Fig. 6 but with only two outliers.

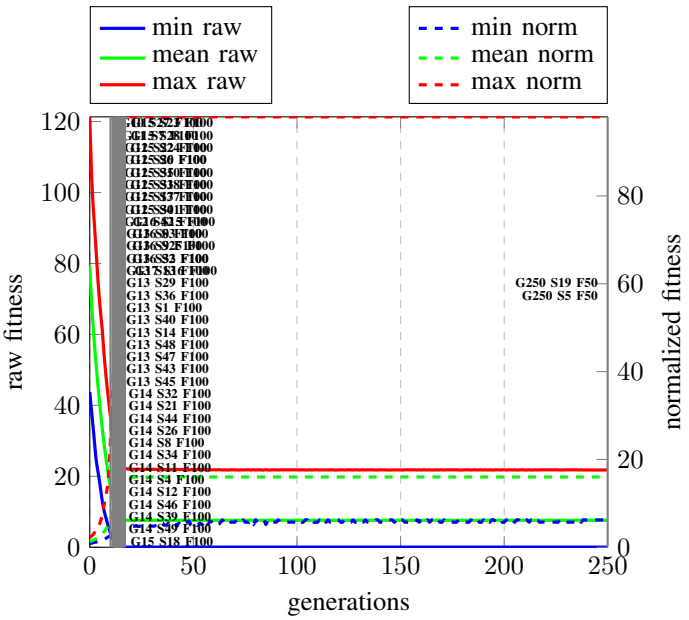


Fig. 7. Test data 2 config 1

Looking at a different configuration we can see that there aren't any runs which didn't finish. But now we can see more closely that while both Fig. 8 and Fig. 9 use the same configuration of parameters Fig. 8 finishes on average at a much later generation than Fig. 9. We can interpret this as the problem set 1 being potentially harder to solve than problem set 2. If we look at Fig. 5 and Fig. 3 we can compare the

average ending generation and notice that yes, for any given configuration the average is lower for problem set 2 compared to problem set 1.

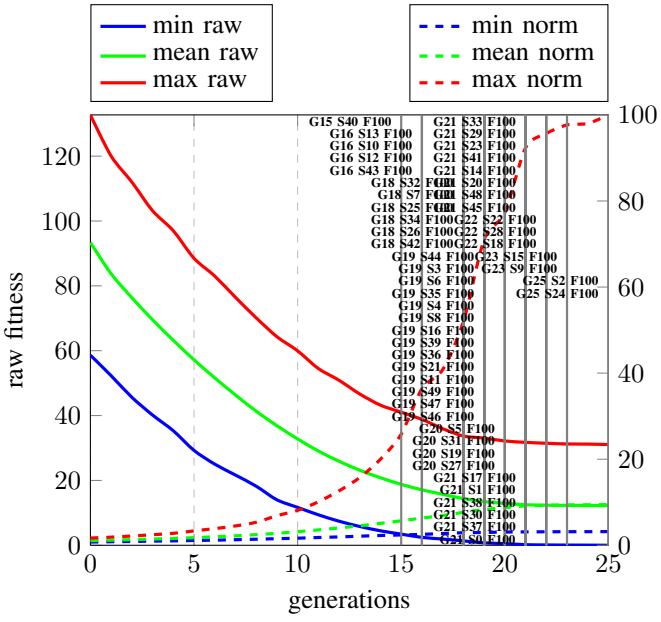


Fig. 8. Test data 1 config 6

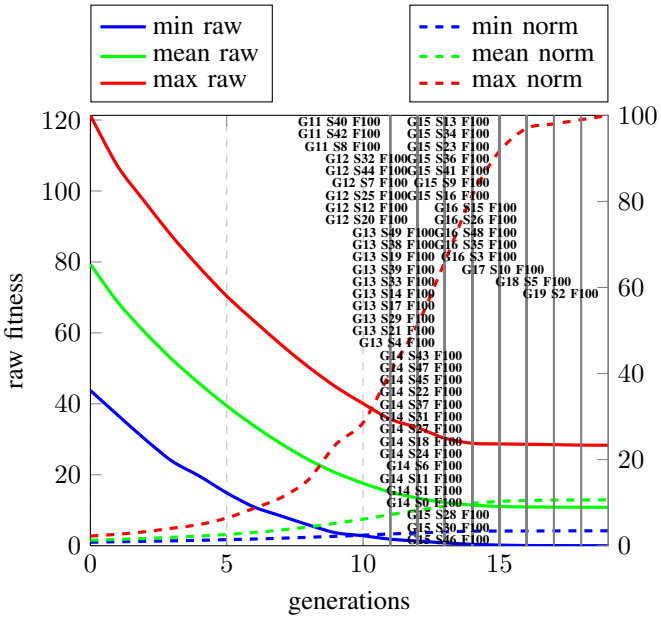


Fig. 9. Test data 2 config 6

If we take the same parameters over the same problem set and only change the crossover method we can see something interesting. Here in Fig. 10 and Fig. 11 we have just that. The configuration using `BestAttempt` crossover has a significantly lower average completed generation compared to Fig. 10. and if we look further we can actually see for all configurations on problem set one using `BestAttempt` Fig. 11, Fig. 27, Fig. 26, Fig. 25, Fig. 24, and Fig. 23 we get a very similar average completed generation.

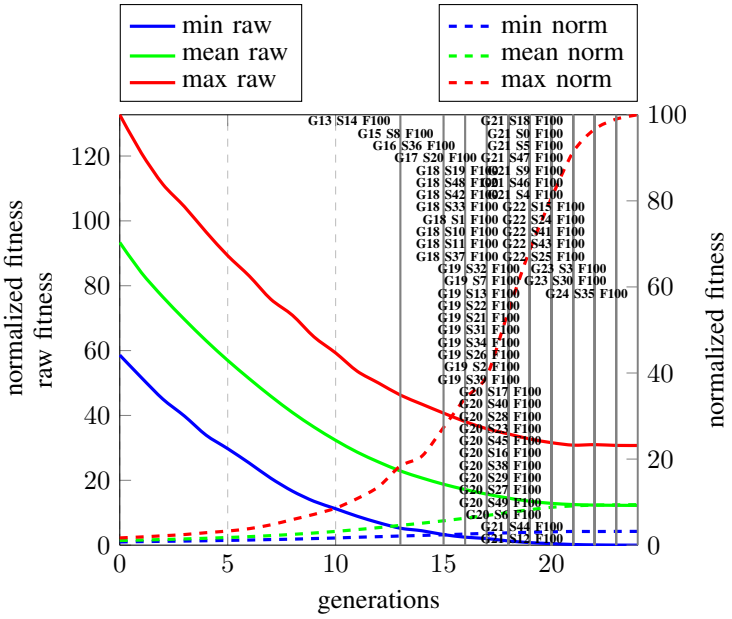


Fig. 10. Test data 1 config 8

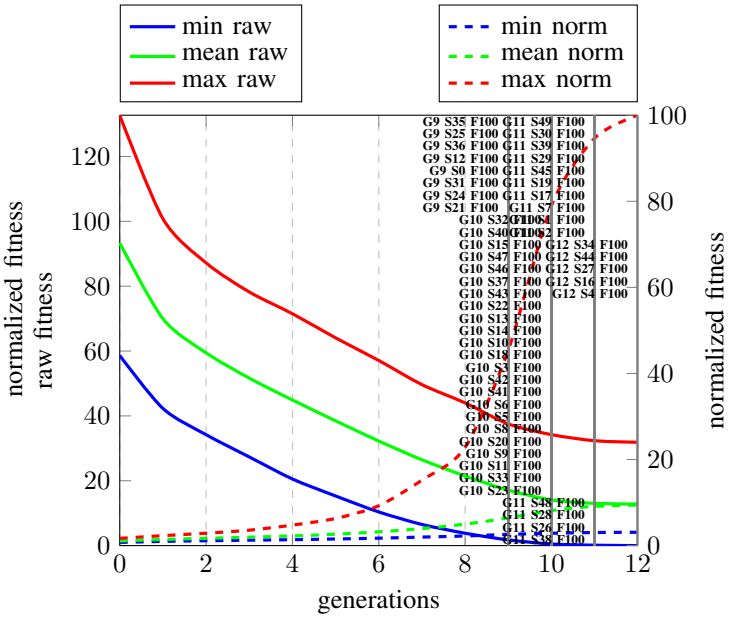


Fig. 11. Test data 1 config 14

These points can be proved using statistical analysis. We will be using the two tailed z-test with a level of significance of 0.05 Our null hypotheses is $H_0 : \mu_0 = \mu_1$ vs alternative hypothesis $H_1 : \mu_0 \neq \mu_1$. We can reject H_0 when $p < 0.05$.

What this means is if we reject the null hypothesis H_0 we can say the difference in mean generations to complete is statistically significant and one configuration will on average find a solution is less generations.

$$Z = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \tag{3}$$

$$p = 1 - erf(\frac{|Z|}{\sqrt{2}}) \tag{4}$$

We first calculate the Z value using (3) for every combination of listed parameter configurations. the mean and standard deviation can be found in tables Fig. 3 and Fig. 5 for problem set 1 and problem set 2 respectively. n is simply the number of tests ran per configuration, which in our case is always 50.

We then find p for every respective Z calculated simply by using (4).

Finally once we have p we can test if we can reject our null hypothesis H_0.

The table cells are colored in according to the following. If we cannot reject the null hypothesis, that is the average completed generation of the two configurations is not statistically significantly different, we color in the cell yellow. If we can reject the null hypotheses we color the cell green if the column configuration has a lower average completed generation and red otherwise.

This intends to show us which configurations are equivalent and if not which configuration is better. To simplify, green means column # configuration is better than row # configuration. yellow means they're the same, and red means it is worse.

These values can be found in tables Fig. 12, Fig. 13, and Fig. 14 for problem set one and Fig. 15, Fig. 16, and Fig. 17 for problem set two.

We can see that in both Fig. 14 and Fig. 17 the configurations who use `BestAttempt` crossover perform consistently better than other configurations using other crossover methods, and interestingly perform about the same as each other when configured with different crossover rates and mutation rates. We can also see that in problem set 1 configuration 1 performs significantly worse than any other configuration. Also to note configurations 7, 5, and 3 all seemed to have slight improvements over other configurations in both problem sets. To all three configurations had a mutation value of 0%.

	config1	config2	config3	config4	config5	config6	config7	config8	config9	config10	config11	config12	config13	config14
config1	0.00	3.24	2.82	3.27	3.36	3.31	3.39	3.32	4.06	4.07	4.05	4.08	4.06	4.06
config2	−3.24	0.00	−0.65	0.62	3.45	1.76	4.07	2.02	27.06	26.90	26.28	27.56	27.08	27.13
config3	−2.82	0.65	0.00	0.72	0.98	0.83	1.05	0.85	2.90	2.90	2.85	2.93	2.89	2.90
config4	−3.27	−0.62	−0.72	0.00	2.66	1.05	3.27	1.29	25.45	25.31	24.71	25.93	25.46	25.51
config5	−3.36	−3.45	−0.98	−2.66	0.00	−1.79	0.77	−1.58	30.77	30.43	29.60	31.51	30.89	30.93
config6	−3.31	−1.76	−0.83	−1.05	1.79	0.00	2.49	0.24	29.48	29.22	28.49	30.10	29.55	29.59
config7	−3.39	−4.07	−1.05	−3.27	−0.77	−2.49	0.00	−2.30	29.48	29.15	28.34	30.19	29.58	29.62
config8	−3.32	−2.02	−0.85	−1.29	1.58	−0.24	2.30	0.00	30.35	30.06	29.30	31.01	30.44	30.48
config9	−4.06	−27.06	−2.90	−25.45	−30.77	−29.48	−29.48	−30.35	0.00	0.11	−1.21	0.92	−0.23	0.00
config10	−4.07	−26.90	−2.90	−25.31	−30.43	−29.22	−29.15	−30.06	−0.11	0.00	−1.27	0.78	−0.34	−0.11
config11	−4.05	−26.28	−2.85	−24.71	−29.60	−28.49	−28.34	−29.30	1.21	1.27	0.00	2.11	1.01	1.22
config12	−4.08	−27.56	−2.93	−25.93	−31.51	−30.10	−30.19	−31.01	−0.92	−0.78	−2.11	0.00	−1.18	−0.93
config13	−4.06	−27.08	−2.89	−25.46	−30.89	−29.55	−29.58	−30.44	0.23	0.34	−1.01	1.18	0.00	0.24
config14	−4.06	−27.13	−2.90	−25.51	−30.93	−29.59	−29.62	−30.48	0.00	0.11	−1.22	0.93	−0.24	0.00

Fig. 12. Test data 1 two tailed z-test chart

	config1	config2	config3	config4	config5	config6	config7	config8	config9	config10	config11	config12	config13	config14
config1	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
config2	0.00	1.00	0.51	0.54	0.00	0.08	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00
config3	0.00	0.51	1.00	0.47	0.32	0.41	0.30	0.39	0.00	0.00	0.00	0.00	0.00	0.00
config4	0.00	0.54	0.47	1.00	0.01	0.29	0.00	0.20	0.00	0.00	0.00	0.00	0.00	0.00
config5	0.00	0.00	0.32	0.01	1.00	0.07	0.44	0.12	0.00	0.00	0.00	0.00	0.00	0.00
config6	0.00	0.08	0.41	0.29	0.07	1.00	0.01	0.81	0.00	0.00	0.00	0.00	0.00	0.00
config7	0.00	0.00	0.30	0.00	0.44	0.01	1.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00
config8	0.00	0.04	0.39	0.20	0.12	0.81	0.02	1.00	0.00	0.00	0.00	0.00	0.00	0.00
config9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.91	0.23	0.36	0.82	1.00
config10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.91	1.00	0.20	0.44	0.74	0.91
config11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.23	0.20	1.00	0.03	0.31	0.22
config12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.36	0.44	0.03	1.00	0.24	0.35
config13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.82	0.74	0.31	0.24	1.00	0.81
config14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.91	0.22	0.35	0.81	1.00

Fig. 13. Test data 1 p value chart

	config1	config2	config3	config4	config5	config6	config7	config8	config9	config10	config11	config12	config13	config14
config1	NA	>	>	>	>	>	>	>	>	>	>	>	>	>
config2	<	NA	NA	NA	>	NA	>	>	>	>	>	>	>	>
config3	<	NA	NA	NA	NA	NA	NA	NA	>	>	>	>	>	>
config4	<	NA	NA	NA	>	NA	>	NA	>	>	>	>	>	>
config5	<	<	NA	<	NA	NA	NA	NA	>	>	>	>	>	>
config6	<	NA	NA	NA	NA	NA	>	NA	>	>	>	>	>	>
config7	<	<	NA	<	NA	<	NA	<	>	>	>	>	>	>
config8	<	<	NA	NA	NA	NA	>	NA	>	>	>	>	>	>
config9	<	<	<	<	<	<	<	<	NA	NA	NA	NA	NA	NA
config10	<	<	<	<	<	<	<	<	NA	NA	NA	NA	NA	NA
config11	<	<	<	<	<	<	<	<	NA	NA	NA	>	NA	NA
config12	<	<	<	<	<	<	<	<	NA	NA	<	NA	NA	NA
config13	<	<	<	<	<	<	<	<	NA	NA	NA	NA	NA	NA
config14	<	<	<	<	<	<	<	<	NA	NA	NA	NA	NA	NA

Fig. 14. Test data 1 comparison chart

	config1	config2	config3	config4	config5	config6	config7	config8	config9	config10	config11	config12	config13	config14
config1	0.00	1.39	1.53	1.43	1.42	1.39	1.53	1.42	2.46	2.48	2.44	2.52	2.48	2.48
config2	−1.39	0.00	2.50	0.74	0.49	0.00	2.38	0.56	23.81	24.25	22.44	24.85	23.84	24.25
config3	−1.53	−2.50	0.00	−2.00	−2.56	−2.72	−0.06	−2.17	25.25	25.78	23.30	26.40	25.12	25.78
config4	−1.43	−0.74	2.00	0.00	−0.35	−0.80	1.86	−0.19	27.71	28.23	25.63	28.81	27.51	28.23
config5	−1.42	−0.49	2.56	0.35	0.00	−0.54	2.38	0.14	33.45	34.04	30.24	34.49	32.79	34.04
config6	−1.39	0.00	2.72	0.80	0.54	0.00	2.57	0.61	27.01	27.51	25.17	28.10	26.89	27.51
config7	−1.53	−2.38	0.06	−1.86	−2.38	−2.57	0.00	−2.03	23.85	24.35	22.15	24.99	23.80	24.35
config8	−1.42	−0.56	2.17	0.19	−0.14	−0.61	2.03	0.00	27.56	28.08	25.53	28.66	27.38	28.08
config9	−2.46	−23.81	−25.25	−27.71	−33.45	−27.01	−23.85	−27.56	0.00	0.97	−0.71	2.64	1.02	0.97
config10	−2.48	−24.25	−25.78	−28.23	−34.04	−27.51	−24.35	−28.08	−0.97	0.00	−1.54	1.71	0.13	0.00
config11	−2.44	−22.44	−23.30	−25.63	−30.24	−25.17	−22.15	−25.53	0.71	1.54	0.00	2.98	1.57	1.54
config12	−2.52	−24.85	−26.40	−28.81	−34.49	−28.10	−24.99	−28.66	−2.64	−1.71	−2.98	0.00	−1.47	−1.71
config13	−2.48	−23.84	−25.12	−27.51	−32.79	−26.89	−23.80	−27.38	−1.02	−0.13	−1.57	1.47	0.00	−0.13
config14	−2.48	−24.25	−25.78	−28.23	−34.04	−27.51	−24.35	−28.08	−0.97	0.00	−1.54	1.71	0.13	0.00

Fig. 15. Test data 2 two tailed z-test chart

	config1	config2	config3	config4	config5	config6	config7	config8	config9	config10	config11	config12	config13	config14
config1	1.00	0.16	0.13	0.15	0.16	0.16	0.13	0.15	0.01	0.01	0.01	0.01	0.01	0.01
config2	0.16	1.00	0.01	0.46	0.62	1.00	0.02	0.57	0.00	0.00	0.00	0.00	0.00	0.00
config3	0.13	0.01	1.00	0.05	0.01	0.01	0.95	0.03	0.00	0.00	0.00	0.00	0.00	0.00
config4	0.15	0.46	0.05	1.00	0.72	0.42	0.06	0.85	0.00	0.00	0.00	0.00	0.00	0.00
config5	0.16	0.62	0.01	0.72	1.00	0.59	0.02	0.89	0.00	0.00	0.00	0.00	0.00	0.00
config6	0.16	1.00	0.01	0.42	0.59	1.00	0.01	0.54	0.00	0.00	0.00	0.00	0.00	0.00
config7	0.13	0.02	0.95	0.06	0.02	0.01	1.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00
config8	0.15	0.57	0.03	0.85	0.89	0.54	0.04	1.00	0.00	0.00	0.00	0.00	0.00	0.00
config9	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.33	0.48	0.01	0.31	0.33
config10	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	1.00	0.12	0.09	0.90	1.00
config11	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.48	0.12	1.00	0.00	0.12	0.12
config12	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.09	0.00	1.00	0.14	0.09
config13	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.31	0.90	0.12	0.14	1.00	0.90
config14	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	1.00	0.12	0.09	0.90	1.00

Fig. 16. Test data 2 p value chart

	config1	config2	config3	config4	config5	config6	config7	config8	config9	config10	config11	config12	config13	config14
config1	NA	NA	NA	NA	NA	NA	NA	NA	>	>	>	>	>	>
config2	NA	NA	>	NA	NA	NA	>	NA	>	>	>	>	>	>
config3	NA	<	NA	<	<	<	NA	<	>	>	>	>	>	>
config4	NA	NA	>	NA	NA	NA	NA	NA	>	>	>	>	>	>
config5	NA	NA	>	NA	NA	NA	>	NA	>	>	>	>	>	>
config6	NA	NA	>	NA	NA	NA	>	NA	>	>	>	>	>	>
config7	NA	<	NA	NA	<	<	NA	<	>	>	>	>	>	>
config8	NA	NA	>	NA	NA	NA	>	NA	>	>	>	>	>	>
config9	<	<	<	<	<	<	<	<	NA	NA	NA	>	NA	NA
config10	<	<	<	<	<	<	<	<	NA	NA	NA	NA	NA	NA
config11	<	<	<	<	<	<	<	<	NA	NA	NA	>	NA	NA
config12	<	<	<	<	<	<	<	<	<	NA	<	NA	NA	NA
config13	<	<	<	<	<	<	<	<	NA	NA	NA	NA	NA	NA
config14	<	<	<	<	<	<	<	<	NA	NA	NA	NA	NA	NA

Fig. 17. Test data 2 comparison chart

V. Conclusion

To conclude we ran and collected data on 1400 individual GA runs over a wide array of different parameters, input data, and seeds. We used the results collected to analyze and show with statistical significance which configurations performed better than others.

The best crossover is `BestAttempt` which outperforms every other configuration in all tests. Interestingly aside from a few exceptions when using `BestAttempt` crossover the crossover rate and mutation rate parameters tested didn't have an effect on the average completed generation.

References

[1] Z. McGovarin and N. Aksamit, "Genetic algorithms tutorial," [Online], 2024.

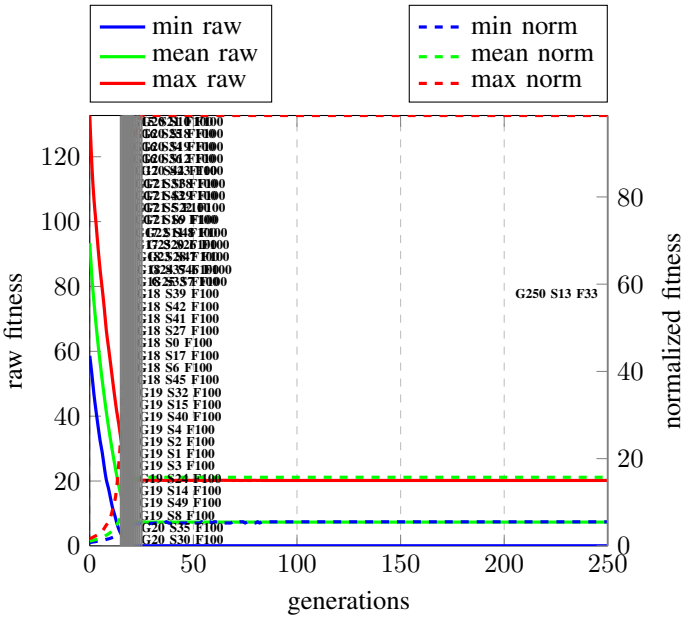


Fig. 19. Test data 1 config 3

Appendix A
Graphs

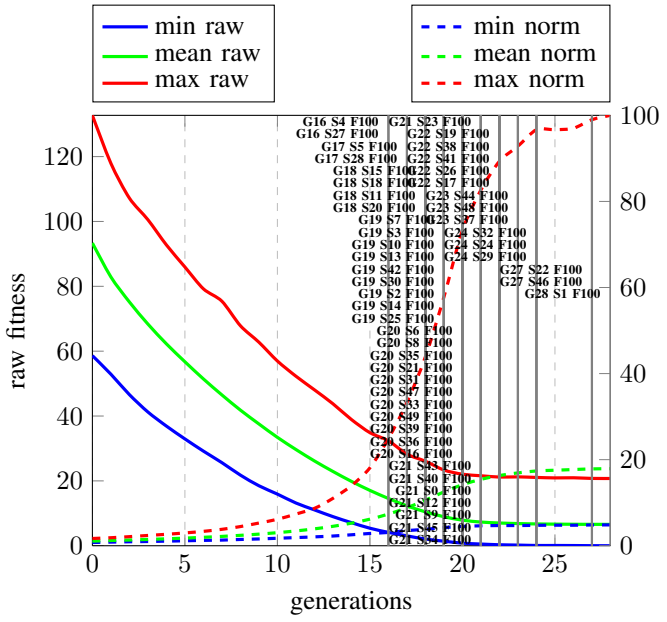


Fig. 18. Test data 1 config 2

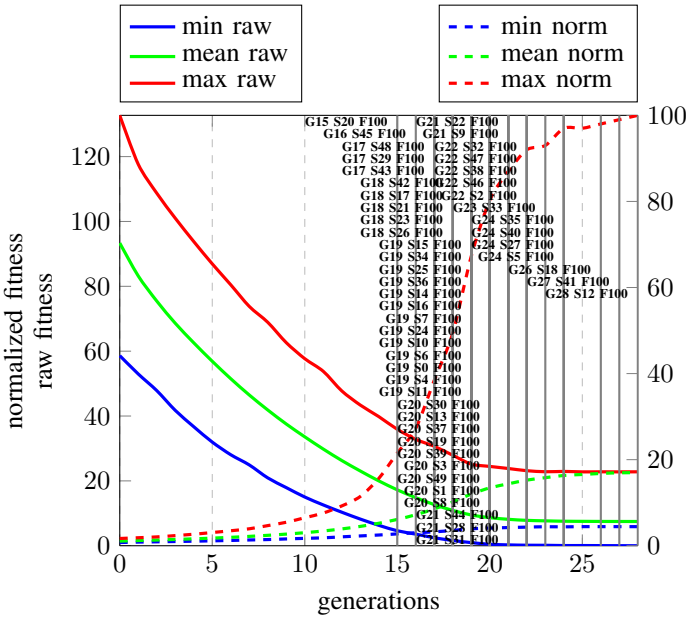


Fig. 20. Test data 1 config 4

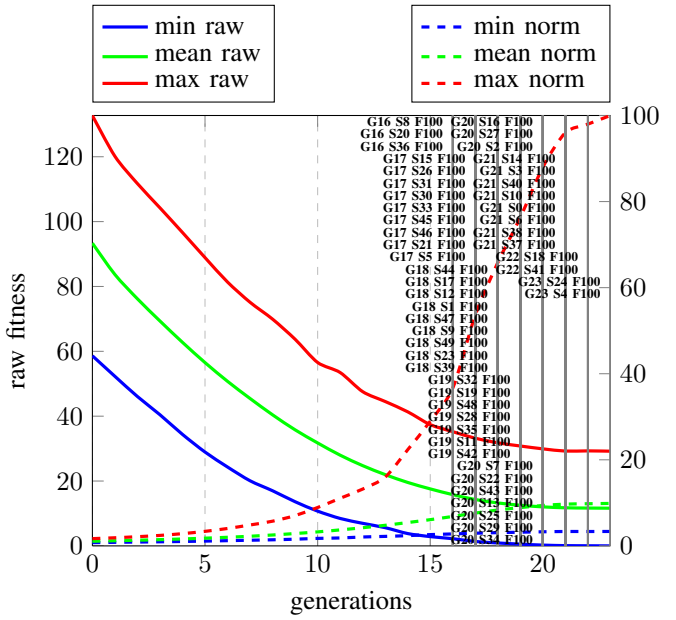


Fig. 21. Test data 1 config 5

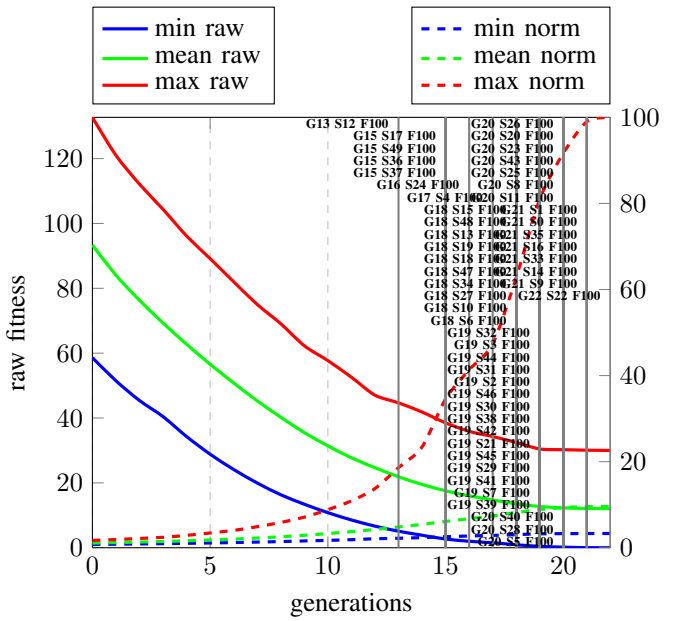


Fig. 22. Test data 1 config 7

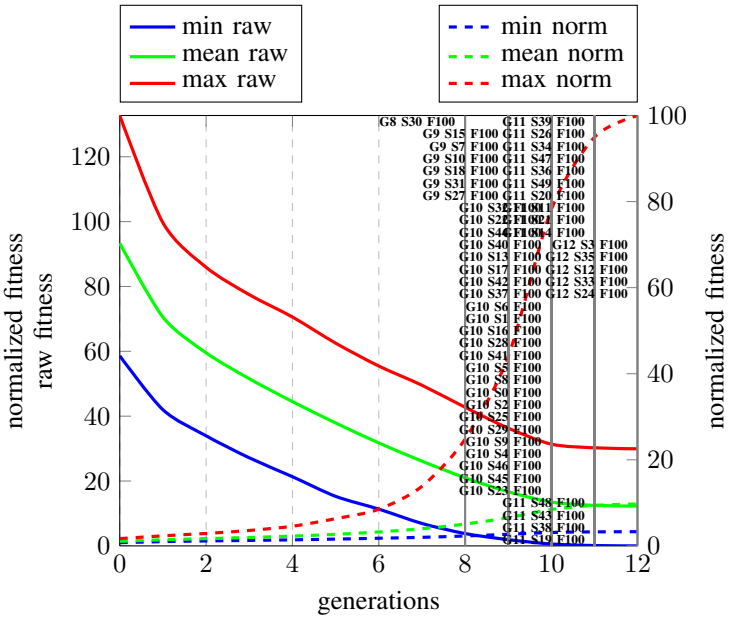


Fig. 23. Test data 1 config 9

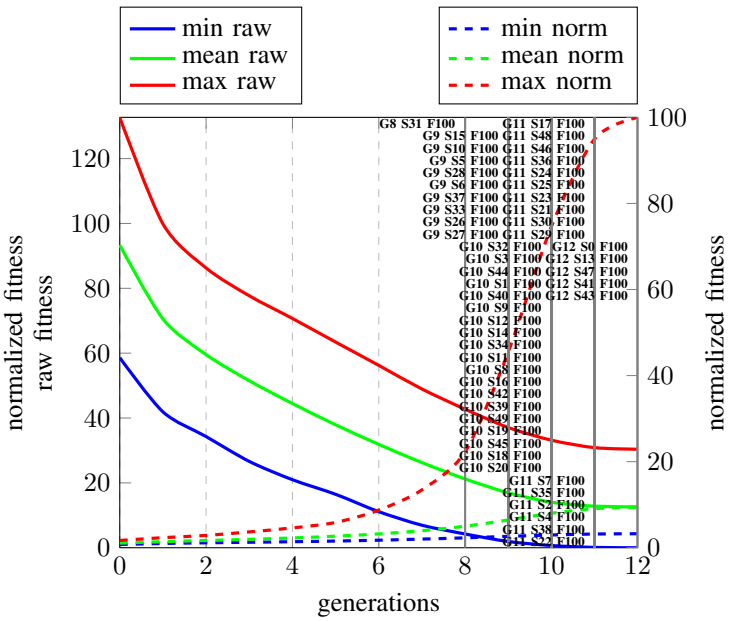


Fig. 24. Test data 1 config 10

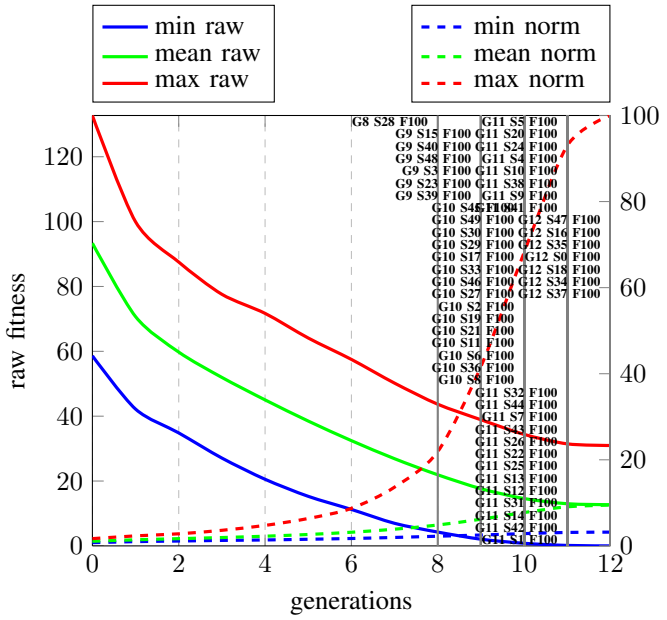


Fig. 25. Test data 1 config 11

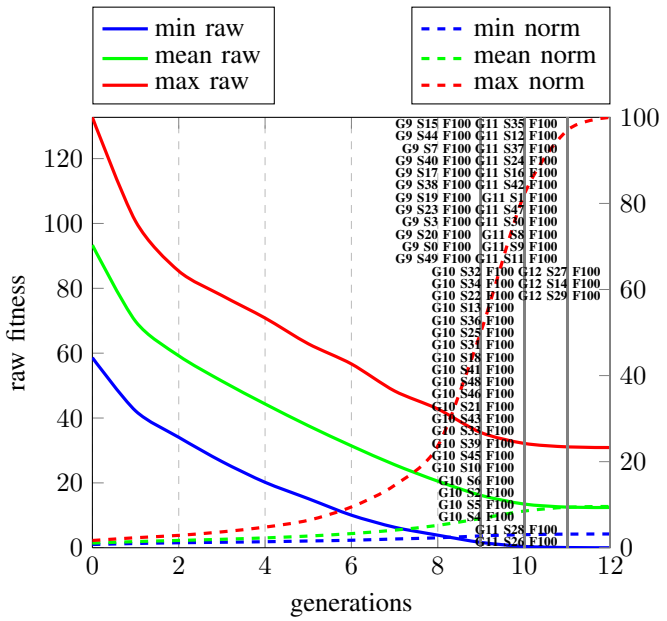


Fig. 26. Test data 1 config 12

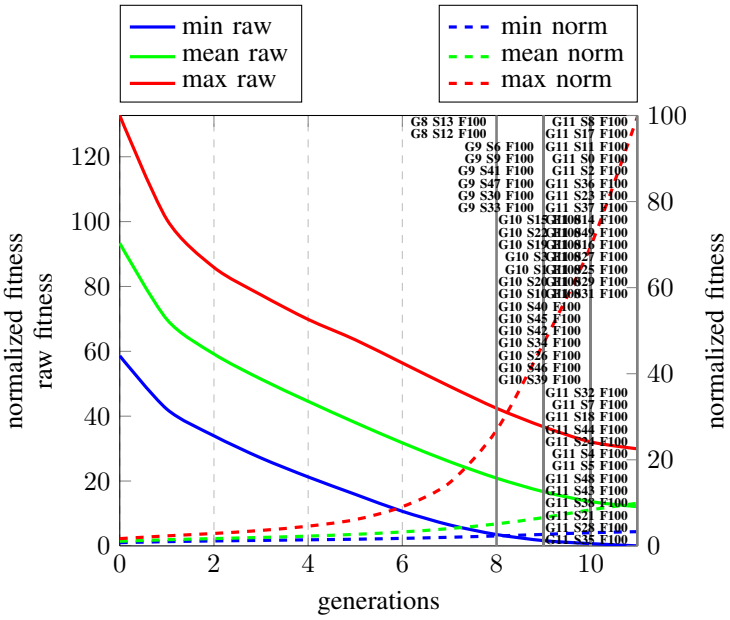


Fig. 27. Test data 1 config 13

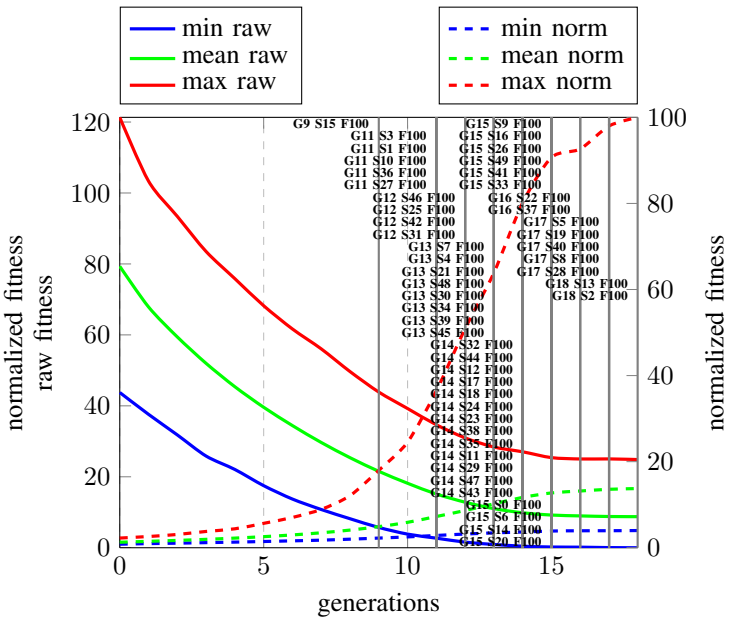


Fig. 28. Test data 2 config 2

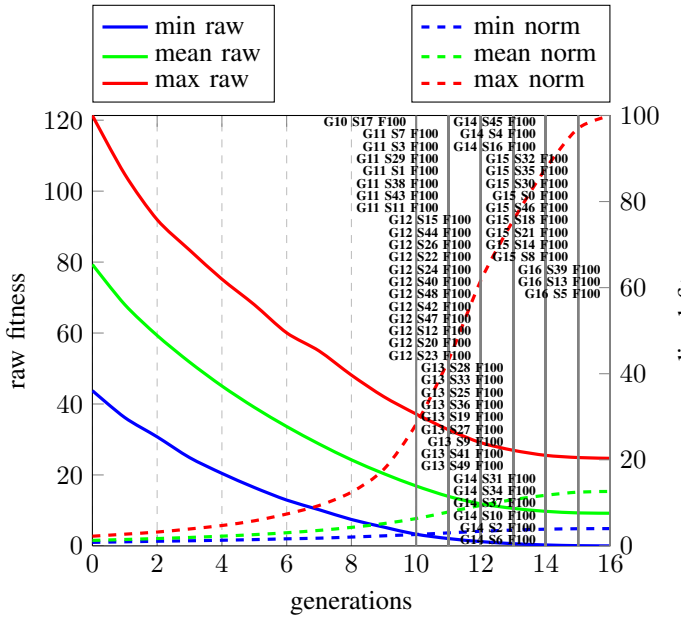


Fig. 29. Test data 2 config 3

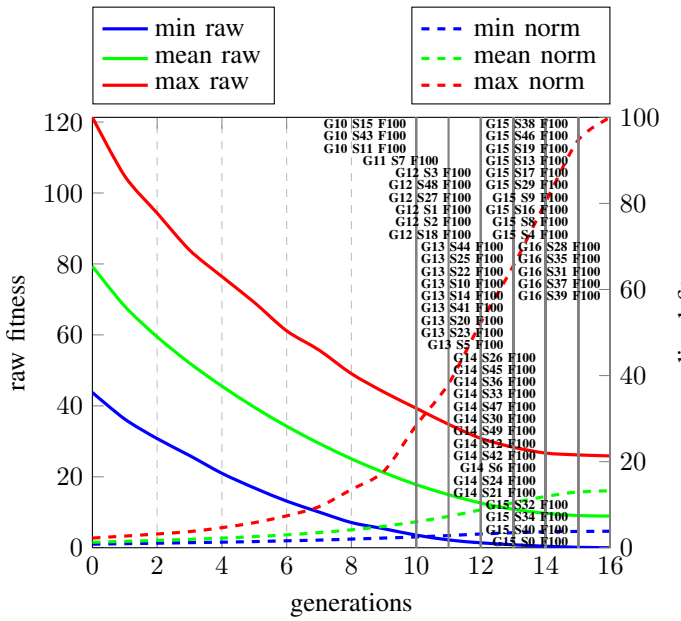


Fig. 30. Test data 2 config 4

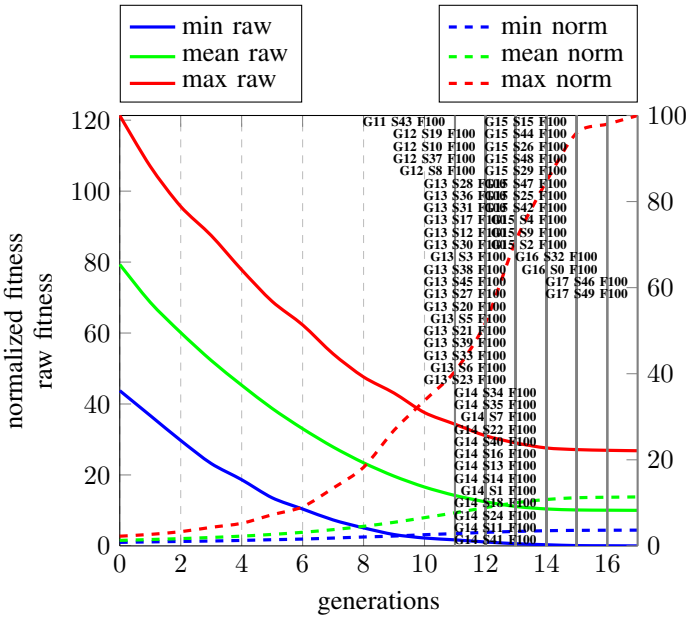


Fig. 31. Test data 2 config 5

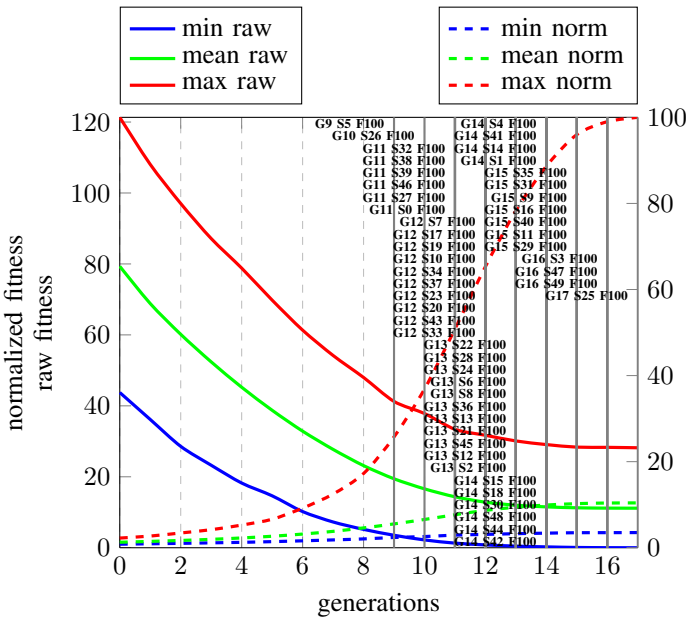


Fig. 32. Test data 2 config 7

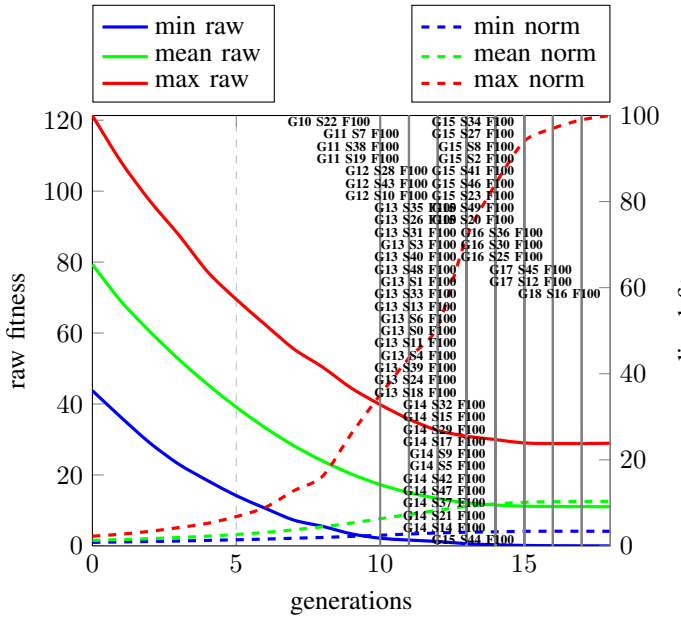


Fig. 33. Test data 2 config 8

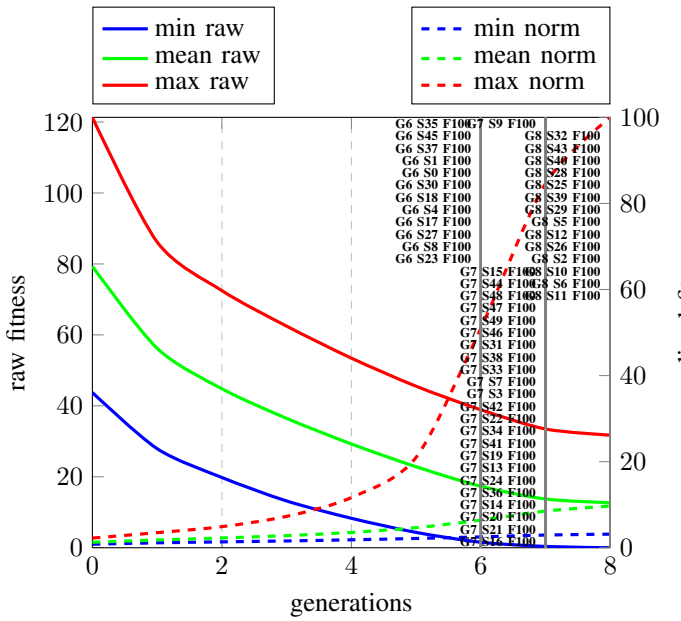


Fig. 34. Test data 2 config 9

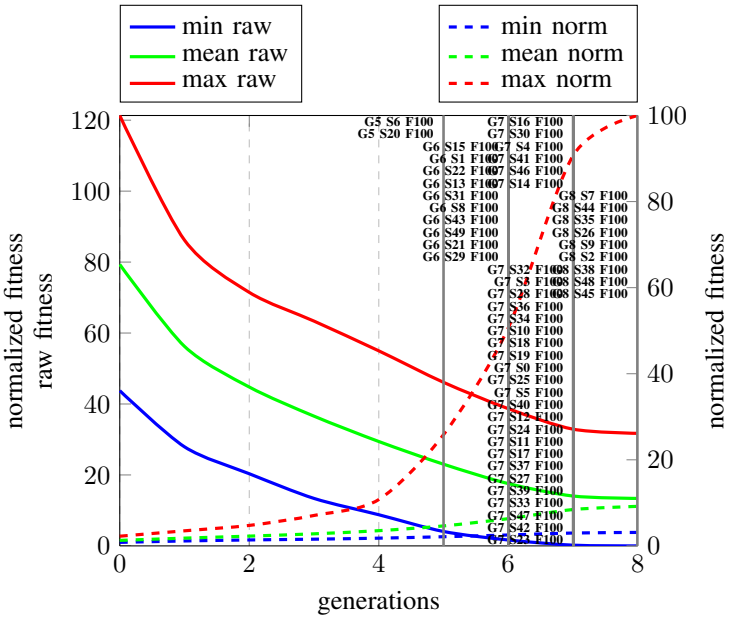


Fig. 35. Test data 2 config 10

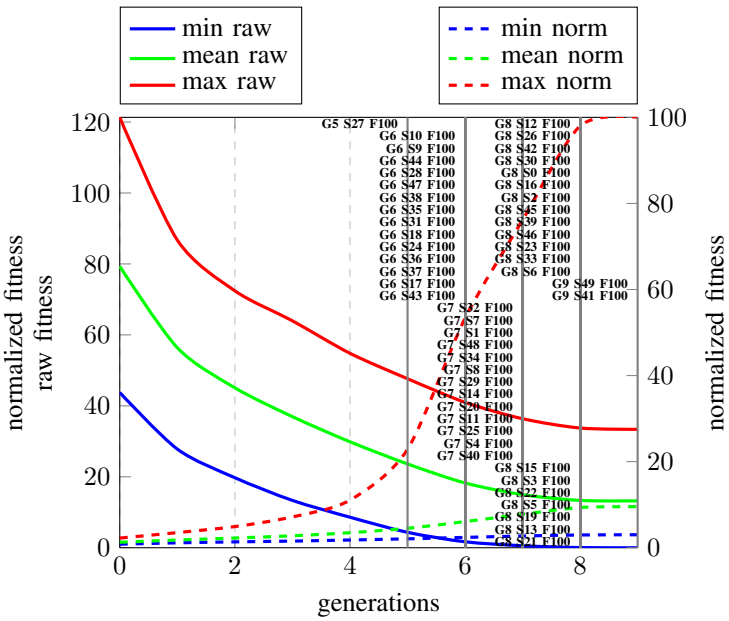


Fig. 36. Test data 2 config 11

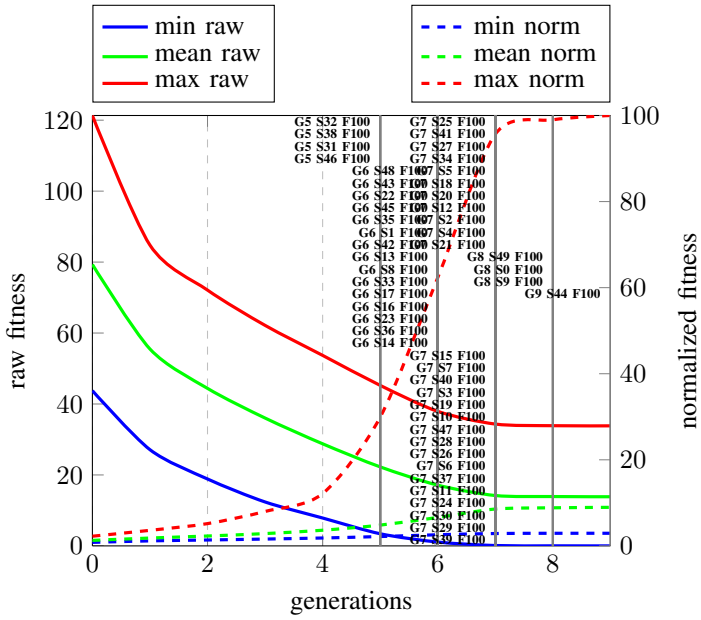


Fig. 37. Test data 2 config 12

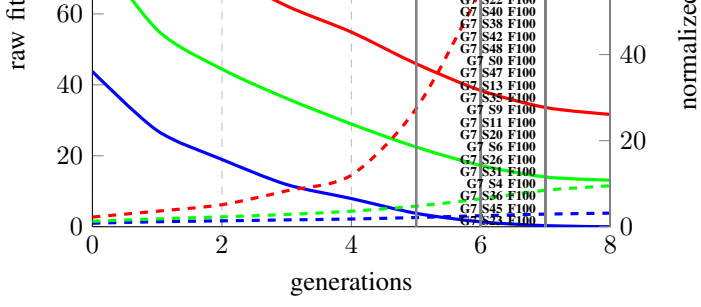


Fig. 38. Test data 2 config 13

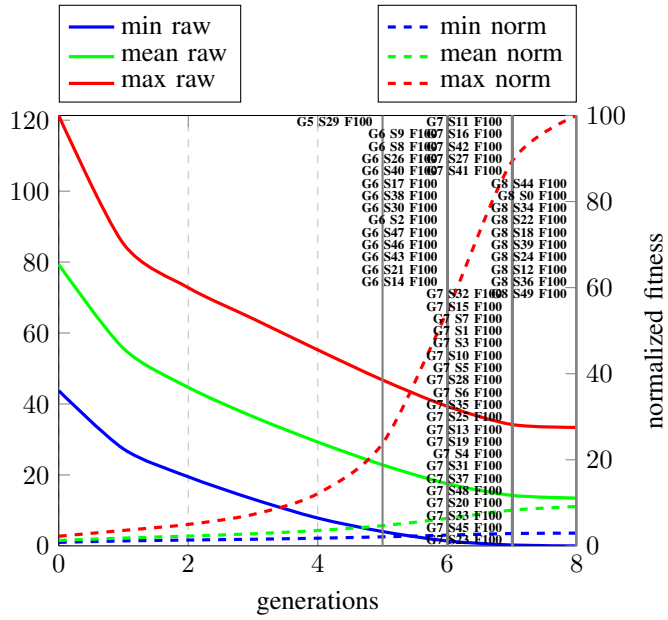


Fig. 39. Test data 2 config 14