TP6 - Self-Adaptation in Evolutionary Strategies

Gabriel Fernandes 2018288117 gabrielf@student.dei.uc.pt
Miguel Rabuge 2018293728 rabuge@student.dei.uc.pt

University of Coimbra

1 Introduction

In this report we present a comparison between a normal genetic algorithm (GA) and a genetic algorithm where the parameters of the mutation are altered during run-time by the evolution process (SA). Each individual's gene is mutated taking into account the respective standard deviation (σ) that has also been mutated throughout the generations of the algorithm.

2 Algorithms' Description

2.1 Representation

In the case of the GA, the individuals are represented by a python list of floats with dimension n, with n being the dimension of the problem.

The SA individuals are also represented by a python list, but with dimension $2n$, due to the fact that besides the genes (values used to compute the individual's fitness), we also need to store the standard deviation values for each gene.

2.2 Variation operators

Crossover
For the crossover operator we decided to use two point crossover. This operator is not as destructive as a per gene implementation, because we exchange chunks of genes (sequential genes) between the two parents. So, if we already have a good sequence of genes, it might not get destroyed in the mating process.

Mutation
The GA was tested with a per gene mutation operator, where the decision to mutate or not, is being done at the level of each gene. If a gene is in fact mutated, a new random value belonging to the problem domain will be assigned.

In the SA, the mutation operator is constituted by two parts: (1) update the genes using the respective standard deviation and (2) update the standard deviations. These updates are only performed if, on a per gene and standard deviation basis, a random value is less than the probability of mutation, similar to the method used in the GA.

Below we show the rules used to on the updates of the SA mutation method:

$$\sigma' = \sigma \times e^{\tau \times N(0,1)}$$

$$x_i' = x_i + \sigma' \times N_i(0,1)$$

where τ is the learning rate.

2.3 Fitness functions

The fitness functions are the benchmark problems that we chose: *Quartic*, *Rastrigin*, *Rosenbrock*, *Sphere* and *Step*. The reason why we chose these problems was due to the similarity between their domains.

3 Experimental Setup

Our experimental setup consists in:

1. Genetic Algorithm's parameters selection
2. Genetic Algorithm's effective run for the benchmark problems, using the chosen parameters
3. Graphical and Statistical Analysis of the results

In the first step, we are unbiased-looking for a set of parameters that do not spoil one algorithm over the other, while looking for the best performing ones. For that, we are not doing a grid search, due to combinatory/time constraints, but we will fix a set of parameters, that we believe that are adequate. These will be the default parameters. Then, we will vary one of them at a time. This allows us to understand how a certain parameter influences the algorithms, which we will later show in plot figures, although neglecting the interactions between parameters. Each one of them was run for 30 runs. The seeds are fixed and the benchmark problems' dimension is 20. The parameters and values that we fixed and tested are presented in table **1**:

Parameters	**Fixed Values**	**Testing Values**
Crossover Probability	0.8	0.7, 0.8
Learning Rate	0.1	0.1, 0.5, 0.9
Mutation Probability	0.1	0.1, 0.3
Population	100	100, 150
Std Domain	[0, 1]	[0,1], [1,1]
Elitism Size	1	-
N Point Crossover	2	-
Tournament Size	3	-
Generations	300	-

Table 1. Default and Testing parameters

In the next step, we will configure the algorithms with the chosen parameters for the benchmark problems. We will log the graphical data, as best's fitness and best's generation regular plots and box plots, as well as perform a statistical analysis, descriptive and inferential. Those are the data results that we will use in order to compare both algorithms, one against the other.

4 Parameter selection

After performing the parameters search, we obtained the following results:

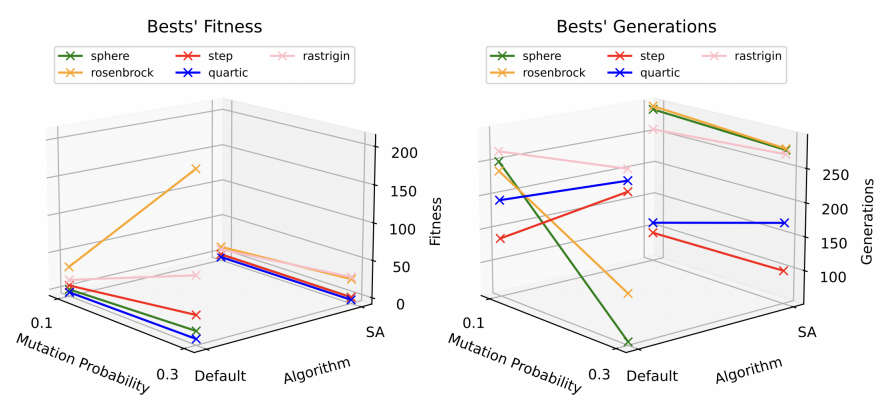

Fig. 1. Mutation Probability Testing. Best of 300 generations of 30 runs.

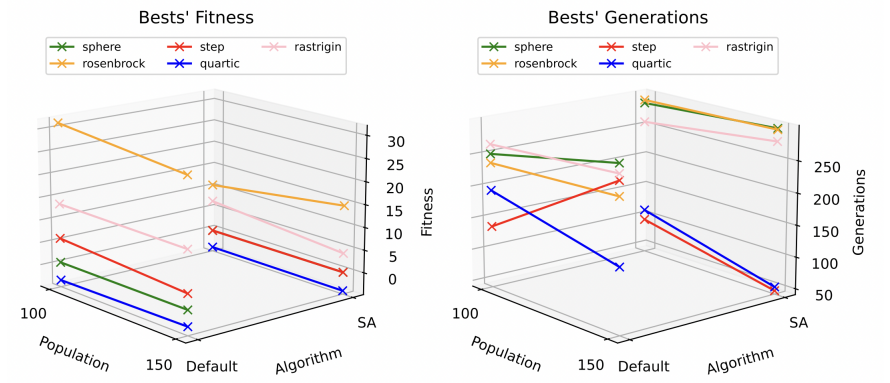

Fig. 2. Population Size Testing. Best of 300 generations of 30 runs.

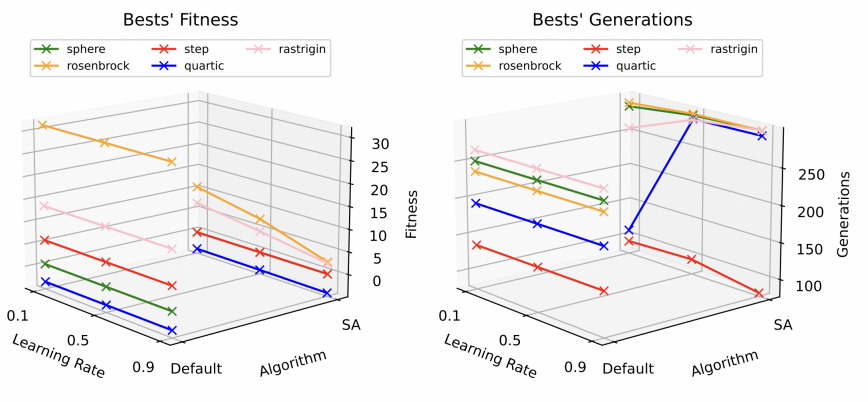

Fig. 3. Learning Rate Testing. Best of 300 generations of 30 runs.

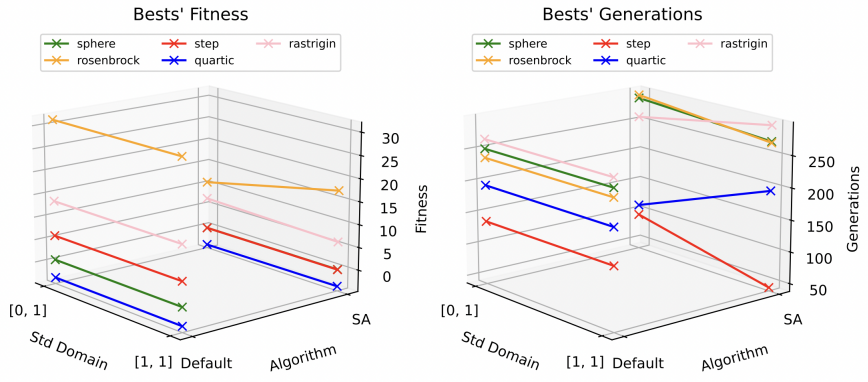

Fig. 4. Standard Deviation Domain Testing. Best of 300 generations of 30 runs.

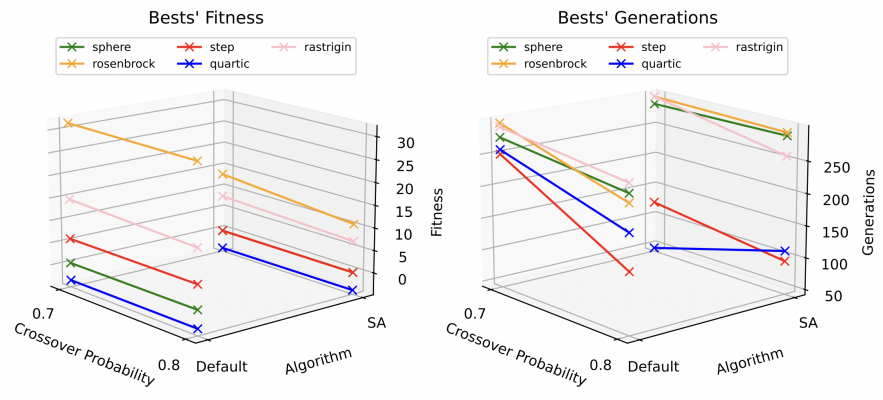

Fig. 5. Crossover Probability Testing. Best of 300 generations of 30 runs.

From figure **1**, we can see that the better mutation probability, in terms of fitness, is clearly the 0.1 value. Regarding the population size, we can observe that it slightly improved the default algorithm fitness. Overall, we concluded that the population sizes tested do not make such a difference. This way, one hundred population size seems a fair choice. Concerning the learning rate value, figure **3**, the value 0.9 stands out from the others, achieving lower values of fitness for *Rosenbrock* and *Rastrigin*, while the others remain basically unchanged. As to the standard deviation domain values, the [0, 1] obtained better results. Finally, in terms of crossover probability, the value 0.8 presents itself slightly better than the other one.

5 Results

With the parameters chosen in the previous section, we ran both algorithms, for 30 runs each, and obtained the following results:

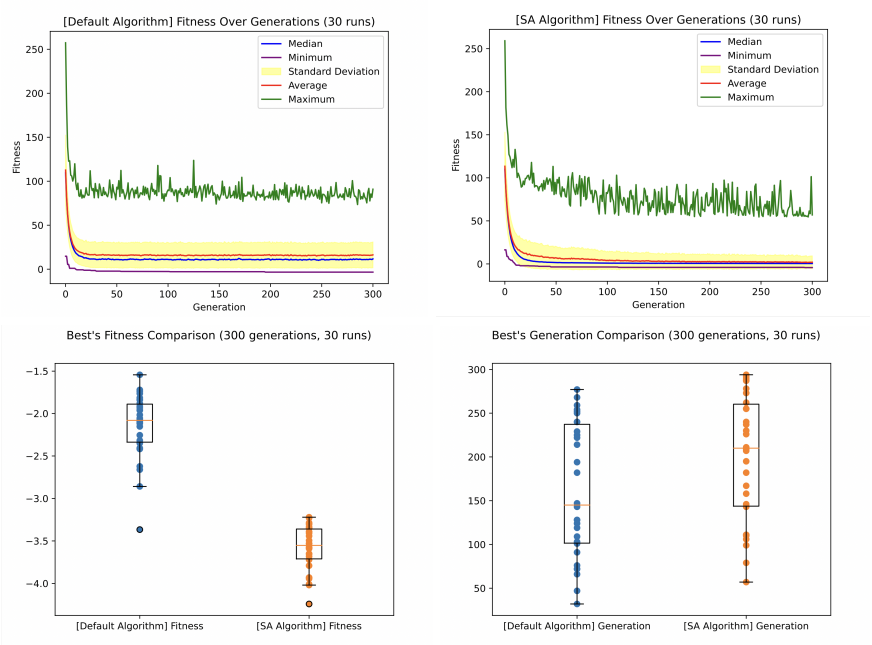

Fig. 6. Quartic Results

At a first glance, it seems that the SA algorithm has fiercely outperformed the default algorithm, since it reaches fitness values way closer to the minimum than the default, for every benchmark problem. Regarding the generation where the best was found, in most problems the SA is very high (close to the max

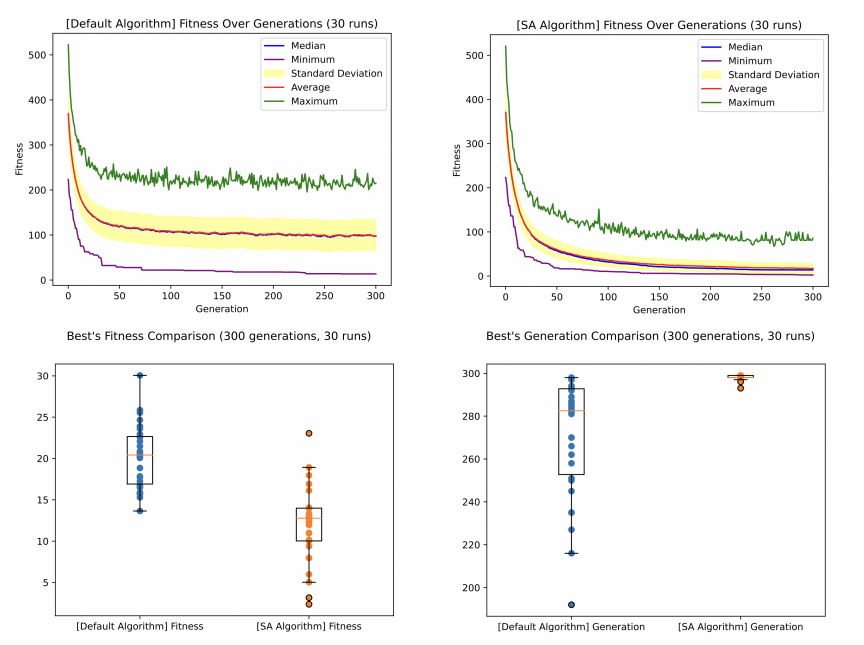

Fig. 7. Rastrigin Results

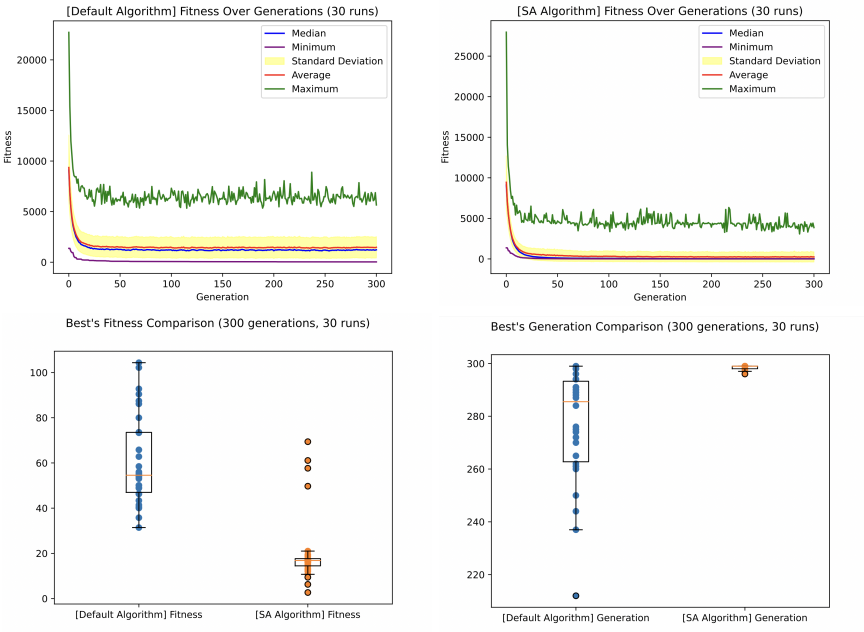

Fig. 8. Rosenbrock Results

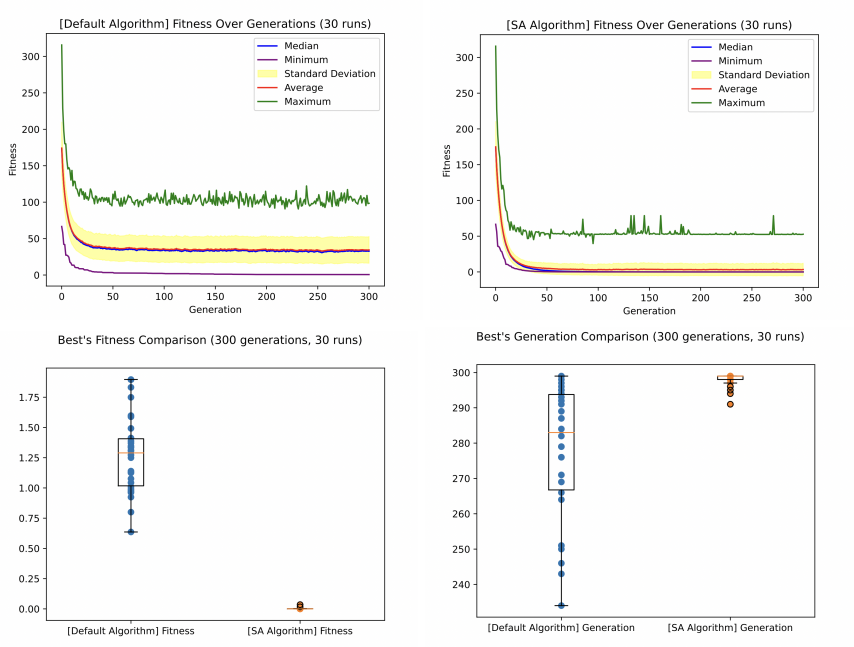

Fig. 9. Sphere Results

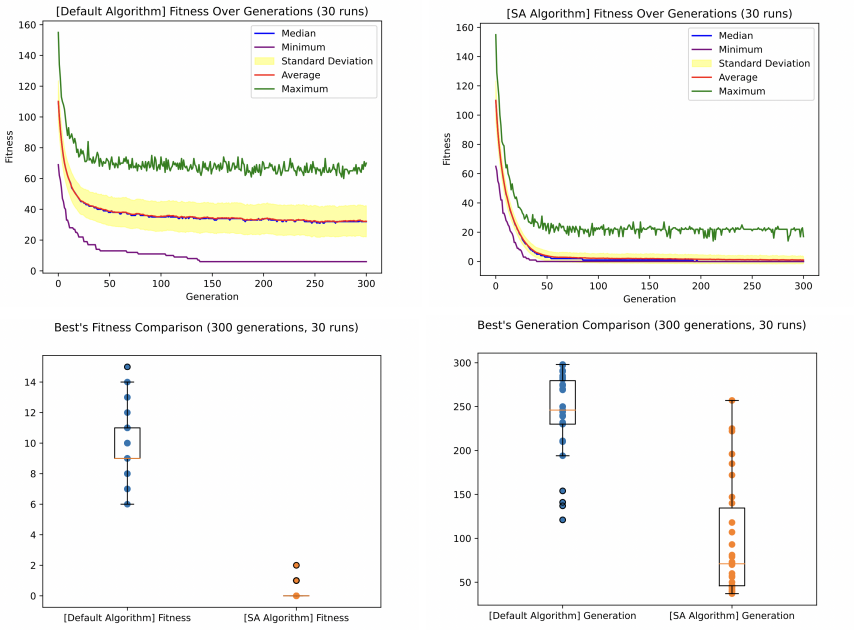

Fig. 10. Step results

generations limit), which alongside the fitness plots over generations, point out that the algorithm is continuously improving his best, until the generations limit is reached. Possibly, it might have a slow convergence rate towards the optimum.

6 Statistical analysis

Having collected the data, now we will proceed to a more formal statistical analysis. Firstly, we will make a statistical description of the data. Next, we will perform hypothesis tests to compare both algorithms. The experiences are paired, since we are comparing one algorithm against the other, and matched since they run over the same set of parameters, on each run, except for the mutation operator, since that is what we are testing. The parametric conditions are checked through a normality test (Kolmogorov-Smirnov) on both algorithms separately, and a variance homogeneity test (levene). All the tests were performed using a significance level of 5%.

Metrics	***Quartic***	***Rastrigin***	***Rosenbrock***	***Sphere***	***Step***
Minimum	-3.367	13.647	31.402	0.636	6
Maximum	-1.543	30.058	104.407	1.897	15
Mean	-2.141	20.283	60.929	1.272	9.8
Std	0.373	3.734	19.812	0.314	2.088
Variance	0.139	13.946	392.524	0.099	4.36
Median	-2.081	20.416	54.564	1.290	9
Mode	-3.367	13.647	31.402	0.636	9; 9
Skew	-1.270	0.377	0.729	0.320	0.574
Kurtosis	1.996	-0.237	-0.560	0.438	0.005
Q1	-2.337	16.908	46.971	1.017	9
Q2	-2.081	20.416	54.564	1.290	9
Q3	-1.890	22.650	73.526	1.407	11

Table 2. Descriptive Statistics of the fitness with GA

The statistical test used was Wilcoxon, because the normality of the data was rejected for every benchmark problem, both for fitness and for generations. As we can observe from tables **6** and **7**, the algorithms are significantly different, having a big effect size on the fitness data, in every benchmark problem. In terms of bests' generations, both of them are also different having a big effect size, except for the Quartic function, which has a medium effect size.

7 Conclusions

In this work two variants of a genetic algorithm were compared, a normal one (GA) and one that evolves the standard deviations of the mutation Gaussian distributions (SA). The plots presented indicate that the SA is better, getting

Metrics	***Quartic***	***Rastrigin***	***Rosenbrock***	***Sphere***	***Step***
Minimum	-4.242	2.364	2.677	0	0
Maximum	-3.221	23.0431	69.404	0.035	2
Mean	-3.589	12.134	20.980	0.002	0.167
Std	0.263	4.559	15.762	0.007	0.453
Variance	0.069	20.788	248.429	0	0.206
Median	-3.551	12.776	16.833	0	0
Mode	-4.242	2.364	2.677	0	0; 26
Skew	-0.681	-0.041	2.008	4.047	2.782
Kurtosis	-0.380	0.135	2.826	15.863	7.069
Q1	-3.711	10.028	14.475	0	0
Q2	-3.553	12.776	16.833	0	0
Q3	-3.360	13.989	17.658	0.002	0

Table 3. Descriptive Statistics of the fitness with SA

Metrics	***Quartic***	***Rastrigin***	***Rosenbrock***	***Sphere***	***Step***
Minimum	32	192	212	234	121
Maximum	277	298	299	299	298
Mean	162.133	270.4	276.8	277.767	240.833
Std	75.435	27.378	21.015	18.825	48.687
Variance	5690.382	749.573	441.627	354.379	2370.406
Median	145.0	282.5	285.5	283	246
Mode	72; 2	294; 3	298; 3	251; 2	230; 2
Skew	-0.017	-1.104	-1.169	-0.778	-1.012
Kurtosis	-1.438	0.443	1.047	-0.596	0.216
Q1	101.5	252.75	262.75	266.75	230
Q2	145	282.5	285.5	283	246
Q3	237.25	292.75	293.25	293.75	279.5

Table 4. Descriptive Statistics of the generations with GA

closer than the GA to the minimum of the benchmark problems. For the *Sphere*, *Rosenbrock* and *Rastrigin* problems, the SA found the best of each run closer to the end of the generations. By looking at the fitness plots, it can be seen that the SA algorithm, after the 200 generations mark, is just fine tuning the best that it found, explaining the high values of generation where it found the best. The statistical tests performed corroborated what we suspected by analysing the fitness plots, i.e. the performances of the algorithms are different, with SA being the better one.
To conclude, it can be said that for these benchmark problems the SA is a better method to optimize them.

References

1. Costa, E.: Estatística Aplicada à Computação Evolucionária (2018)
2. Costa, E.: Nature-Inspired Artificial Intelligence. Springer Nature (2021)

Metrics	***Quartic***	***Rastrigin***	***Rosenbrock***	***Sphere***	***Step***
Minimum	57	293	296	291	37
Maximum	294	299	299	299	257
Mean	199.933	298.133	298.467	298.033	96.167
Std	71.706	1.231	0.806	1.835	64.507
Variance	5141.730	1.520	0.649	3.366	4161.206
Median	210.0	298	299	299	71
Mode	111; 2	299; 14	299; 19	299; 20	46; 3
Skew	-0.348	-2.506	-1.422	-2.349	1.088
Kurtosis	-1.153	7.564	1.178	5.341	-0.117
Q1	143.75	298	298	298	46
Q2	210	298	299	299	71
Q3	260.25	299	299	299	134.5

Table 5. Descriptive Statistics of the generations with SA

Problems	KS_{GA}	KS_{SA}	**Levene**	**Test**	**p-value**	**effect size**
Quartic	0	0	0.308	Wilcoxon	0	-0.615
Rastrigin	0	0.002	0.615	Wilcoxon	0	-0.601
Rosenbrock	0	0	0.055	Wilcoxon	0	-0.567
Sphere	0	0	0	Wilcoxon	0	-0.617
Step	0	0	0	Wilcoxon	0	-0.617

Table 6. Inferential Statistics for the fitness

Problems	KS_{GA}	KS_{SA}	**Levene**	**Test**	**p-value**	**effect size**
Quartic	0	0	0.495	Wilcoxon	0.06	-0.243
Rastrigin	0	0	0	Wilcoxon	0	-0.597
Rosenbrock	0	0	0	Wilcoxon	0	-0.586
Sphere	0	0	0	Wilcoxon	0	-0.597
Step	0	0	0.282	Wilcoxon	0	-0.607

Table 7. Inferential Statistics for the generations