Machine Learning
Università della Svizzera Italiana
Faculty of Informatics
December 23, 2019

Assignment 5: Evolutionary Algorithms

Giorgia Adorni (giorgia.adorni@usi.ch)

1 Introduction

In this assignment, we are going to implement a few evolutionary algorithms against test functions and analyze the characteristics of different algorithms. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a test function with n-dimensional domain.

Sphere function:

$$f(\overline{x}) = \sum_{i=1}^{n} x_i^2. \tag{1}$$

Rastrigin functionction:

$$f(\overline{x}) = An + \sum_{i=1}^{n} [x_i^2 - A\cos(2\pi x_i)] \tag{2}$$

where $A = 10$. The search domain is constraint as $x_i \in [-5, 5]$.

2 Test Functions

In Figure 1 and 2 are shown the 2D contour-plot of 2-dimensional test functions.

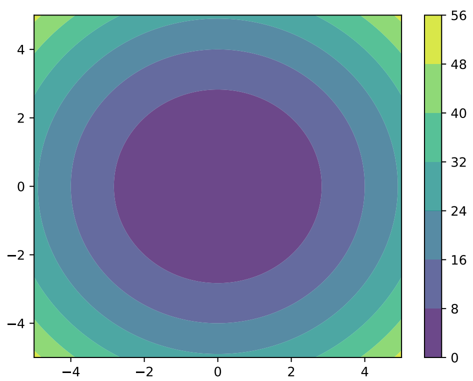


Figure 1: 2D contour plot of 2D Sphere

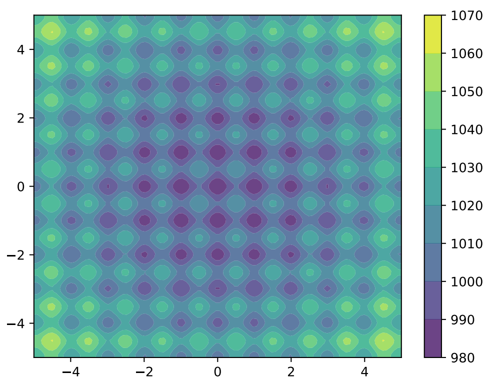


Figure 2: 2D contour plot of 2D Rastrigin

For each test function, 100 points has been uniformly sampled in the domain and evaluated with the test function. This points are shown in Figure 3 and 4. Filling the

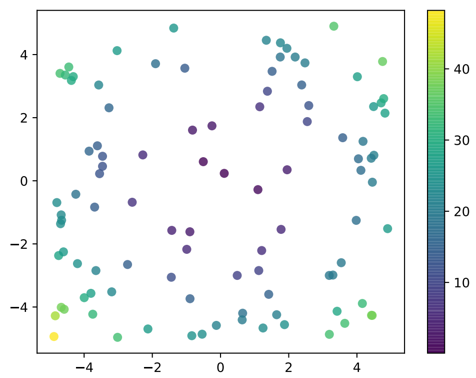


Figure 3: Sphere test function evaluation

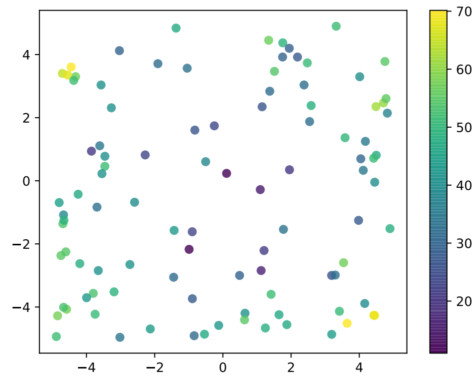


Figure 4: Rastrigin test function evaluation

contours and showing the colour-map it is easier to see which are the regions of the global optimum. All the points with value close to 0, the ones in purple, belong to the global optimal region.

For the sphere, the points are concentrated in the central region of the figure.

For the rastrigin function evaluate the samples is many more complicated, since the function has many local minima. However, the global optimum is located in the centre of plot. For this reason, it will be seen later that finding the global optimum of this function will be difficult.

3 Cross-Entropy Method (CEM)

experiment	population	elite	generations
`baseline`	100	0.20	100
`pop_size-1000`	1000	0.20	100
`pop_size-3000`	3000	0.20	100
`elite-40`	100	0.40	100
`elite-30`	100	0.30	100
`elite-10`	100	0.10	100
`pop_size-1000+elite-40`	1000	0.40	100
`pop_size-1000+elite-30`	1000	0.30	100
`pop_size-1000+elite-10`	1000	0.10	100
`pop_size-3000+elite-40`	3000	0.40	100
`pop_size-3000+elite-30`	3000	0.30	100
`pop_size-3000+elite-10`	3000	0.10	100
`iter-200`	100	0.20	200
`iter-50`	100	0.20	50
`iter-30`	100	0.20	30
`iter-200+elite-30`	100	0.30	200
`iter-200+pop_size-3000+elite-30`	3000	0.30	200

Table 1: CEM parameters

The first algorithm implemented is the **Cross-Entropy Method (CEM)**.

Different execution of the algorithm has been performed. The first experiment is called baseline, and it uses the same parameters as for the following algorithms. After that, were made some attempts of improving the performances trying different population size and elite set ratio, and also with different number of generations. In all the experiments the domain size is fixed to 100 dimensions.

For all the experiments, the initial population parameters are initialised reasonably far from the global optimum, in particular, the mean is uniformly sampled in the range [-5, 5] and the variance is uniformly sampled in the range [4, 5].

In Table 1 are shown the different combinations of parameters used.

The algorithm has been run 3 times for both test functions. In order to evaluate the performance, for each pair of experiment and test function, the best and the worse fitness for each generation (averaged over 3 runs) has been plotted.

In Tables 2 and 3 are summarised the results.

experiment	**best fitness**	**worse fitness**	**avg run time**
`baseline`	7.17	7.19	0.43 sec
`pop_size-1000`	0.0	0.0	0.94 sec
`pop_size-3000`	0.0	0.0	2.24 sec
`elite-40`	2.65	2.71	0.54 sec
`elite-30`	1.54	1.56	0.42 sec
`elite-10`	93.74	93.74	0.42 sec
`pop_size-1000+elite-40`	0.0	0.0	0.95 sec
`pop_size-1000+elite-30`	0.0	0.0	0.98 sec
`pop_size-1000+elite-10`	0.0	0.0	0.95 sec
`pop_size-3000+elite-40`	0.0	0.0	2.17 sec
`pop_size-3000+elite-30`	0.0	0.0	2.27 sec
`pop_size-3000+elite-10`	0.0	0.0	2.24 sec
`iter-200`	8.57	8.57	0.62 sec
`iter-50`	12.09	12.9	0.43 sec
`iter-30`	27.18	35.71	0.36 sec
`iter-200+elite-30`	0.18	0.18	0.5 sec
`iter-200+pop_size-3000+elite-30`	0.0	0.0	3.63 sec

Table 2: Sphere CEM performance

In Figure 5 is plotted the best and the worse fitness for each generation (averaged over 3 runs) for the baseline model. Below, in Figures 6 and 7, are plotted the fitness of the two models that perform better with the sphere function.

For the sphere function, it is possible to see that simply using a large enough population size the algorithm converges. Moreover, increasing the elite set ratio the algorithm converges quickly.

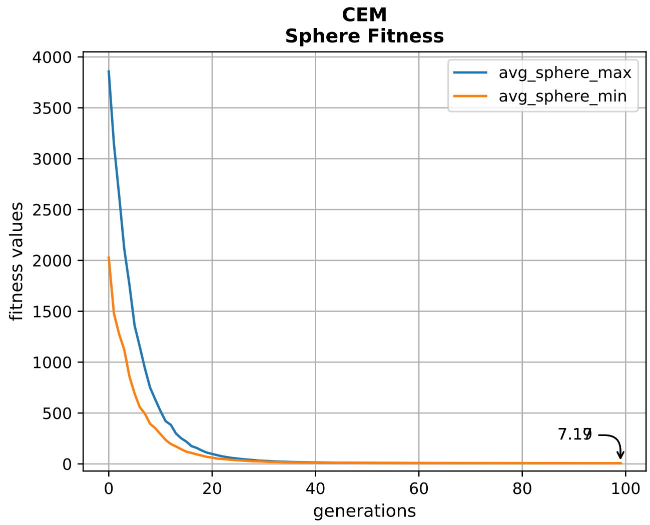


Figure 5: Sphere fitness `baseline`

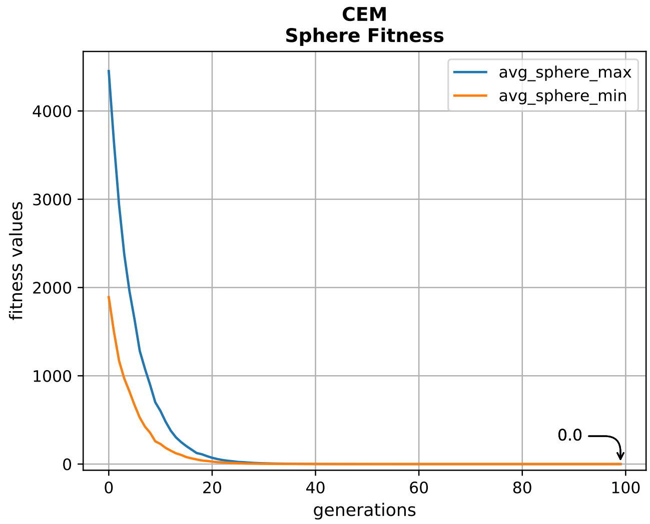


Figure 6: Sphere fitness `pop_size-1000`

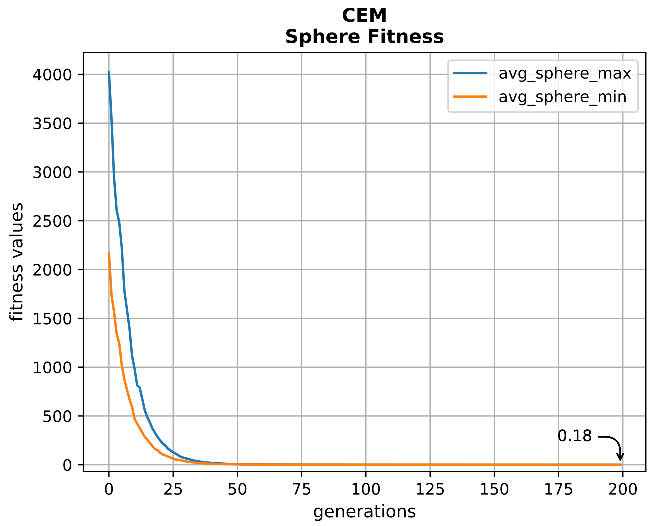
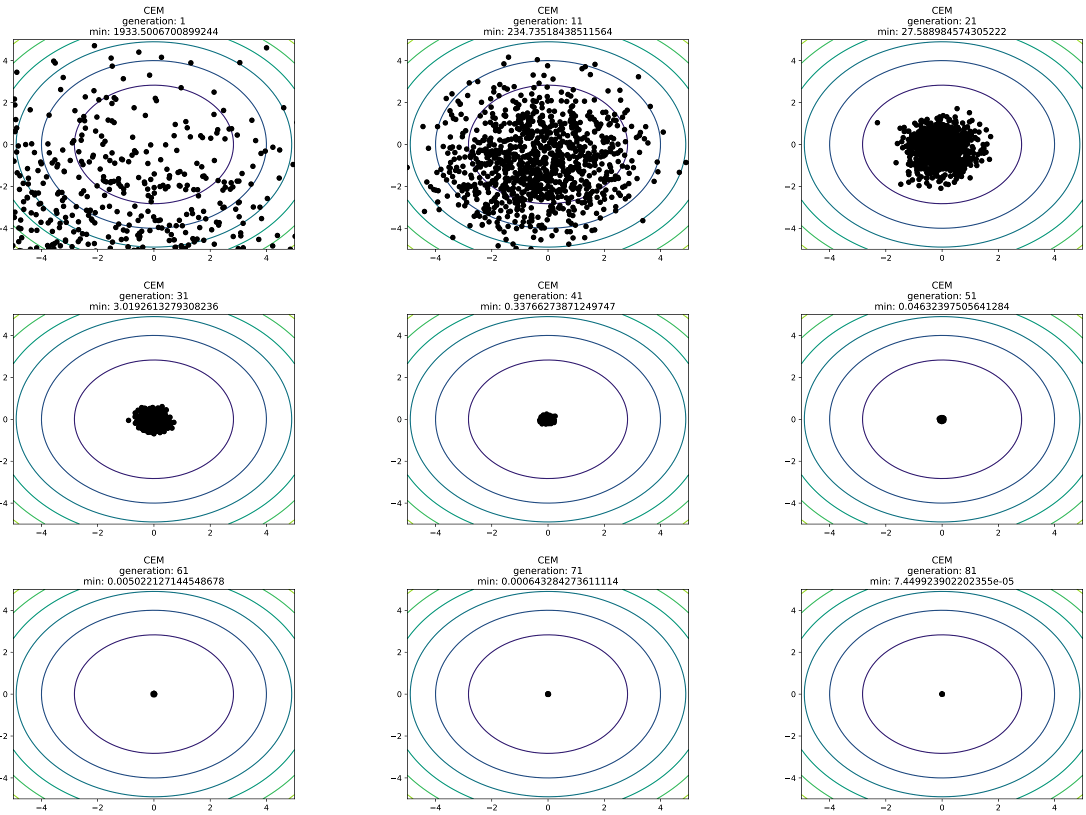


Figure 7: Sphere fitness `iter-200+elite-30`

In the following Figures is shown the effect of executing the algorithm on the experiment `pop_size-1000`.



For the rastrigin function, it is necessary not only to increase the size of the population but also the number of generations to allow the algorithm to converge. Moreover, in this case, by decreasing the elite set ratio the algorithm performs better.

In Figure 8 is plotted the best and the worse fitness for each generation (averaged over 3 runs) for the baseline model. Below, in Figures 9 and 10, are plotted the fitness of the two models that perform better with the rastrigin function.

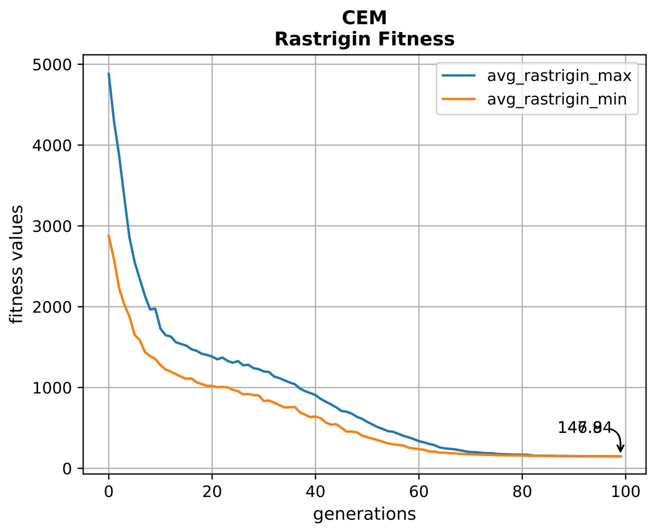


Figure 8: Sphere fitness `baseline`

experiment	best fitness	worse fitness	avg run time
`baseline`	146.9	147.84	0.74 sec
`pop_size-1000`	408.33	726.5	1.39 sec
`pop_size-3000`	561.48	998.0	3.0 sec
`elite-40`	183.21	308.04	0.64 sec
`elite-30`	143.56	173.52	0.68 sec
`elite-10`	342.08	342.11	0.66 sec
`pop_size-1000+elite-40`	754.4	1225.99	1.37 sec
`pop_size-1000+elite-30`	647.67	1084.44	1.37 sec
`pop_size-1000+elite-10`	70.65	194.74	1.4 sec
`pop_size-3000+elite-40`	774.94	1290.65	3.08 sec
`pop_size-3000+elite-30`	740.86	1245.91	3.37 sec
`pop_size-3000+elite-10`	117.32	317.21	2.9 sec
`iter-200`	134.35	134.36	0.88 sec
`iter-50`	449.25	641.98	0.8 sec
`iter-30`	831.28	1178.28	0.64 sec
`iter-200+elite-30`	83.65	83.76	0.74 sec
`iter-200+pop_size-3000+elite-30`	15.85	45.2	4.96 sec

Table 3: Rastrigin CEM performance

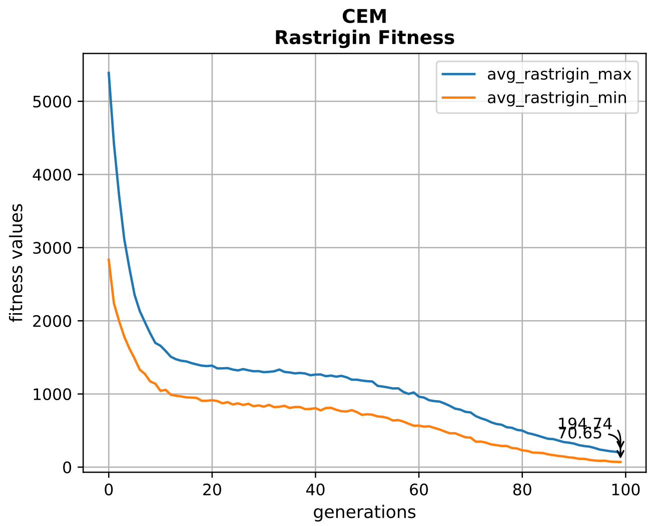


Figure 9: Sphere fitness `pop_size-1000`

For the sphere function, 40 generations are enough to obtain a solution close enough to the global optimum, as shown for the experiment `pop_size-1000`.

For the rastrigin function and the parameter tested, at least 200 generations are necessary to obtain a solution close enough to the global optimum, as shown for the experiment `iter-200+elite-30`.

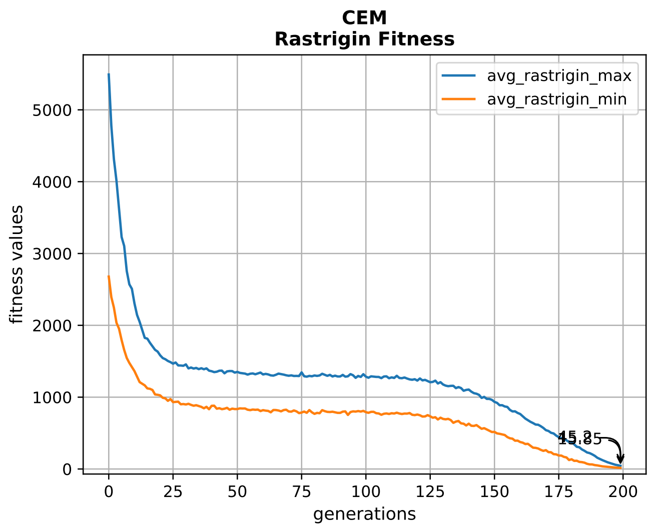


Figure 10: Sphere fitness `iter-200+elite-30`

4 Natural Evolution Strategy (NES)

The second algorithm implemented is the **Natural Evolution Strategy (NES)**.

As for the previous method, different execution of the algorithm has been performed.

experiment	population	learning rate	generations
`baseline`	100	1e-2	100
`pop_size-1000`	1000	1e-2	100
`pop_size-3000`	3000	1e-2	100
`pop_size-5000`	5000	1e-2	100
`lr-001`	100	1e-3	100
`lr-0001`	100	1e-4	100
`lr-00001`	100	1e-5	100
`pop_size-1000+lr-001`	1000	1e-3	100
`pop_size-1000+lr-0001`	1000	1e-4	100
`pop_size-5000+lr-001`	5000	1e-3	100
`pop_size-5000+lr-0001`	5000	1e-4	100
`iter-2000`	100	1e-2	2000
`iter-5000`	100	1e-2	5000
`iter-2000+pop-5000`	5000	1e-2	2000
`iter-5000+pop-5000`	5000	1e-2	5000
`iter-2000+pop-5000+lr-001`	5000	1e-3	5000
`iter-2000+pop-5000+lr-0001`	5000	1e-4	5000

Table 4: NES parameters

The first experiment is called baseline, and it uses the same parameters as for the following algorithms. After that, were made some attempts of improving the performances trying different population size and elite set ratio, and also with different number of generations. In all the experiments the domain size is fixed to 100 dimensions.

For all the experiments, the initial population parameters are initialised reasonably far from the global optimum, in particular, the mean is uniformly sampled in the range [-5, 5] and the covariance matrix is initialised as a diagonal matrix with points uniformly sampled in the range [4, 5].

In Table 4 are shown the different combinations of parameters used for this algorithm.

The algorithm has been run 3 times for both test functions. In order to evaluate the performance, for each pair of experiment and test function, the best and the worse fitness for each generation (averaged over 3 runs) has been plotted.

In Tables 2 and 3 are summarised the results.

experiment	**best fitness**	**worse fitness**	**avg run time**
`baseline`	Err	Err	Err
`pop_size-1000`	Err	Err	Err
`pop_size-3000`	31.77	55.11	1.37 sec
`pop_size-5000`	37.28	103.57	4.46 sec
`lr-001`	Err	Err	Err
`lr-0001`	Err	Err	Err
`lr-00001`	Err	Err	Err
`pop_size-1000+lr-001`	326.19	831.2	1.28 sec
`pop_size-1000+lr-0001`	1154.99	3024.52	1.33 sec
`pop_size-5000+lr-001`	247.58	702.4	4.9 sec
`pop_size-5000+lr-0001`	1090.84	2837.48	4.67 sec
`iter-2000`	Err	Err	Err
`iter-5000`	Err	Err	Err
`iter-2000+pop-5000`	0.5	1.44	26.06 sec
`iter-5000+pop-5000`	0.21	0.55	60.59 sec
`iter-2000+pop-5000+lr-001`	14.39	41.89	81.64 sec
`iter-2000+pop-5000+lr-0001`	132.99	383.86	84.32 sec

Table 5: Sphere NES performance

For the sphere function, it is possible to see that for many combinations of parameters it is not possible to execute the algorithm. This is because a `ValueError` is often returned due to the transformation of the covariance matrix into a non-positive one. Therefore, by increasing the population size enough and the number of generations, the algorithm is able to converge. With the presented parameters, using too low a learning rate reduces the algorithm's convergence speed.

In Figures 11, 12 and 13 are plotted the best and the worse fitness for each generation (averaged over 3 runs) for the models that perform better with the sphere function.

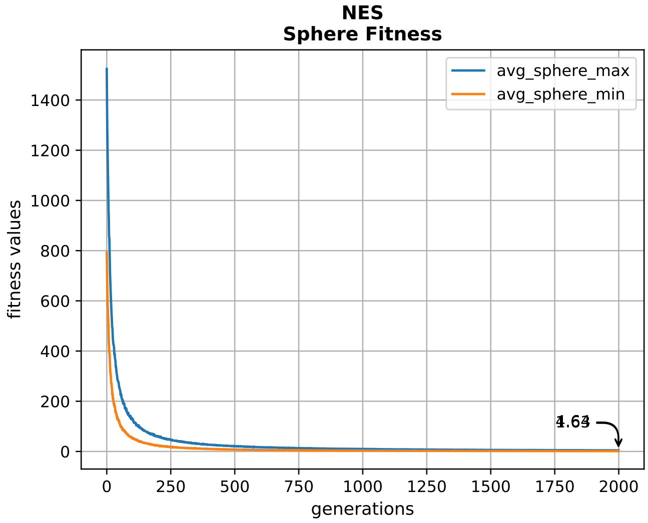


Figure 11: Sphere fitness `iteration-2000+pop_size-5000`

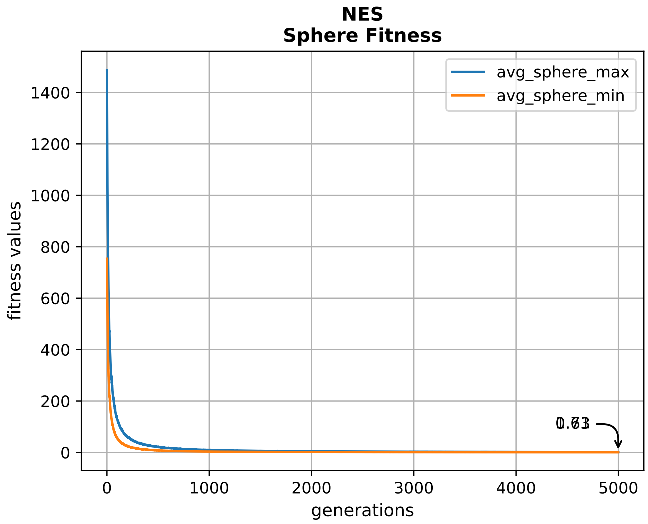


Figure 12: Sphere fitness `iteration-5000+pop_size-5000`

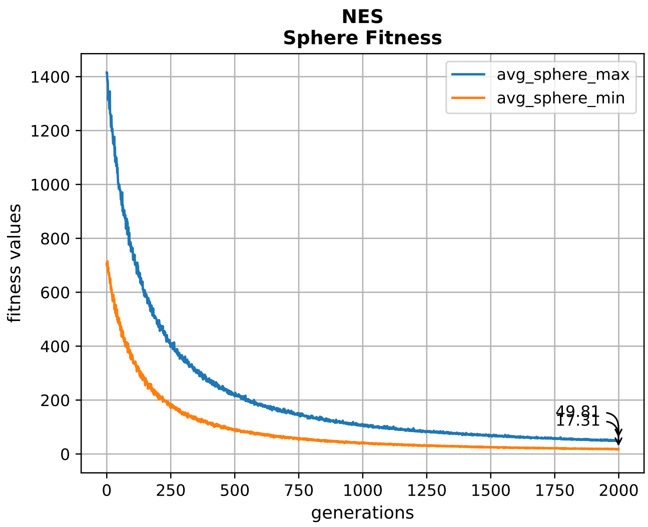


Figure 13: Sphere fitness `iteration-2000+pop_size-5000+lr-001`

For the rastrigin function, it is fundamental to increase the number of generations to allow the algorithm to converge. With the tested parameters, it was not possible to converge the algorithm to a global optimum.

experiment	**best fitness**	**worse fitness**	**avg run time**
`baseline`	Err	Err	Err
`pop_size-1000`	Err	Err	Err
`pop_size-3000`	Err	Err	Err
`pop_size-5000`	Err	Err	Err
`lr-001`	Err	Err	Err
`lr-0001`	Err	Err	Err
`lr-00001`	Err	Err	Err
`pop_size-1000+lr-001`	1358.28	2154.99	1.74 sec
`pop_size-1000+lr-0001`	2169.83	4029.91	1.94 sec
`pop_size-5000+lr-001`	1142.55	1834.66	6.05 sec
`pop_size-5000+lr-0001`	2000.94	4007.66	5.74 sec
`iter-2000`	Err	Err	Err
`iter-5000`	Err	Err	Err
`iter-2000+pop-5000`	Err	Err	Err
`iter-5000+pop-5000`	Err	Err	Err
`iter-2000+pop-5000+lr-001`	764.96	1279.41	103.81 sec
`iter-2000+pop-5000+lr-0001`	955.97	1558.37	106.25 sec

Table 6: Rastrigin NES performance

In Figures 14 and 15 are plotted the best and the worse fitness for each generation (averaged over 3 runs) of two models performed.

For the sphere function, 2000 generations are enough to obtain a solution close enough to the global optimum, as shown for the experiment `iteration-2000+pop_size-5000`.

For the rastrigin function are probably needed more than 5000 generations to obtain a solution close enough to the global optimum.

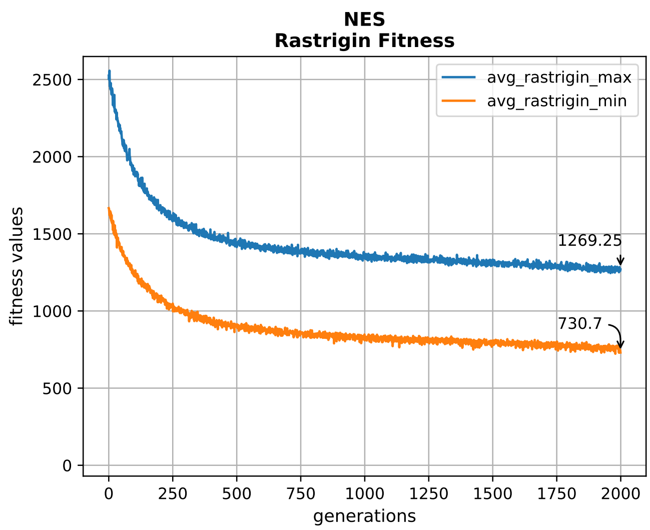


Figure 14: Rastrigin fitness `iteration-2000+pop_size-5000+lr-001000`

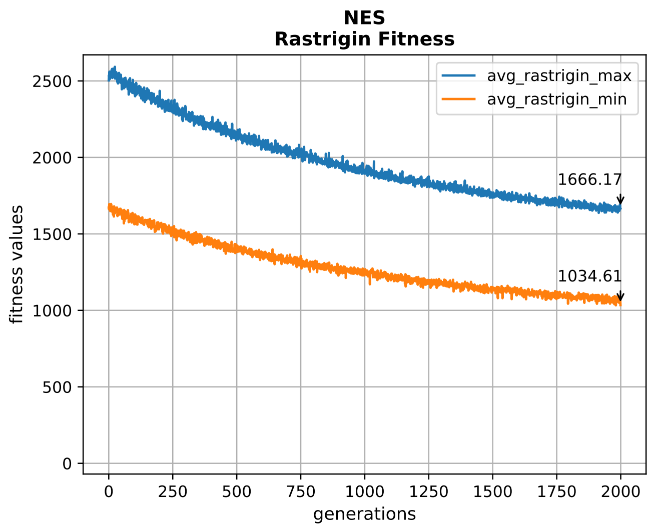


Figure 15: Rastrigin fitness `iteration-2000+pop_size-5000+lr-0001`

5 Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

The last algorithm implemented is the **Covariance Matrix Adaptation Evolution Strategy (CMA-ES)**.

Different execution of the algorithm has been performed using the same parameters employed with the first method. In Table 7 are shown the different combinations of parameters used.

experiment	**population**	**elite**	**generations**
`baseline`	100	0.20	100
`pop_size-1000`	1000	0.20	100
`pop_size-3000`	3000	0.20	100
`elite-40`	100	0.40	100
`elite-30`	100	0.30	100
`elite-10`	100	0.10	100
`pop_size-1000+elite-40`	1000	0.40	100
`pop_size-1000+elite-30`	1000	0.30	100
`pop_size-1000+elite-10`	1000	0.10	100
`pop_size-3000+elite-40`	3000	0.40	100
`pop_size-3000+elite-30`	3000	0.30	100
`pop_size-3000+elite-10`	3000	0.10	100
`iter-200`	100	0.20	200
`iter-50`	100	0.20	50
`iter-30`	100	0.20	30
`iter-200+elite-30`	100	0.30	200
`iter-200+pop_size-3000+elite-30`	3000	0.30	200

Table 7: CMA-ES parameters

The algorithm has been run 3 times for both test functions. In order to evaluate the performance, for each pair of experiment and test function, the best and the worse fitness for each generation (averaged over 3 runs) has been plotted.

In Tables 8 and 9 are summarised the results.

experiment	**best fitness**	**worse fitness**	**avg run time**
`baseline`	567.81	567.82	0.63 sec
`pop_size-1000`	92.09	92.11	1.14 sec
`pop_size-3000`	0.01	0.01	2.56 sec
`elite-40`	586.48	586.48	0.75 sec
`elite-30`	570.19	570.21	0.86 sec
`elite-10`	657.87	657.87	0.6 sec
`pop_size-1000+elite-40`	56.86	57.09	1.2 sec
`pop_size-1000+elite-30`	40.8	40.84	1.2 sec
`pop_size-1000+elite-10`	132.82	132.83	1.09 sec
`pop_size-3000+elite-40`	0.0	0.01	2.58 sec
`pop_size-3000+elite-30`	0.0	0.0	2.43 sec
`pop_size-3000+elite-10`	6.4	6.41	2.3 sec
`iter-200`	586.27	586.27	0.83 sec
`iter-50`	609.34	609.35	0.47 sec
`iter-30`	602.65	602.69	0.47 sec
`iter-200+elite-30`	554.4	554.4	0.84 sec
`iter-200+pop_size-3000+elite-30`	0.0	0.0	4.87 sec

Table 8: Sphere CMA-ES performance

For the sphere function, it is possible to see that simply using a large enough population size the algorithm converges.

In Figure 16 is plotted the best and the worse fitness for each generation (averaged over 3 runs) for the baseline model. Below, in Figure 17 is plotted the fitness of the model that performs better with the sphere function.

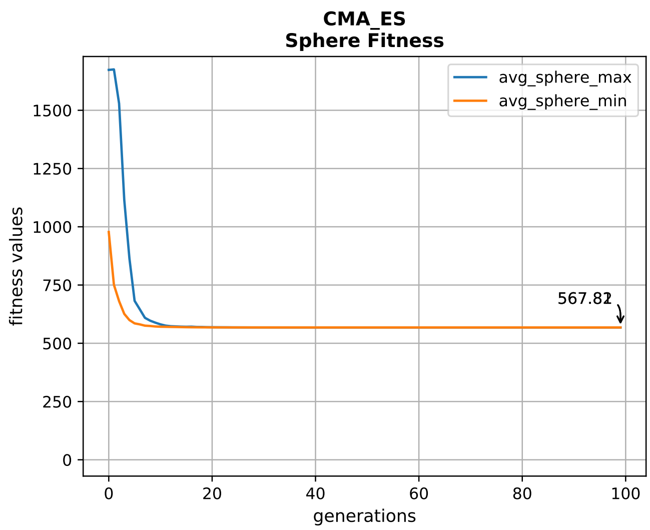


Figure 16: Sphere fitness `baseline`

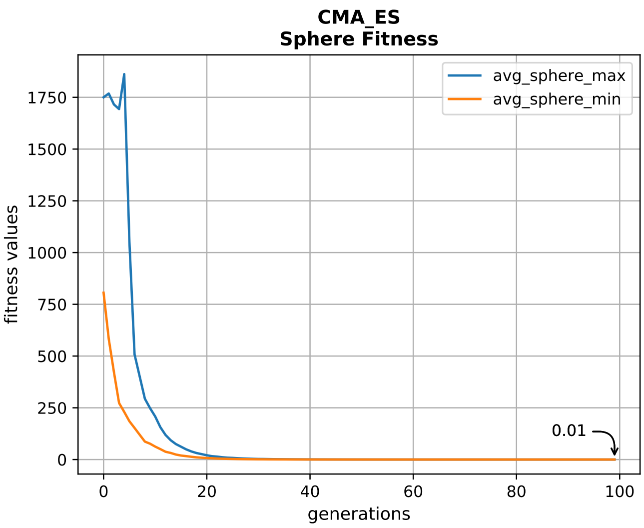


Figure 17: Sphere fitness `pop_size-3000`

For the rastrigin function, it is necessary not only to increase the size of the population but also the number of generations to allow the algorithm to converge.

experiment	best fitness	worse fitness	avg run time
`baseline`	1394.23	1394.23	0.95 sec
`pop_size-1000`	446.51	447.11	1.53 sec
`pop_size-3000`	120.04	126.67	3.51 sec
`elite-40`	1298.12	1298.27	0.86 sec
`elite-30`	1433.35	1433.38	0.87 sec
`elite-10`	1385.37	1385.37	0.82 sec
`pop_size-1000+elite-40`	443.78	453.89	1.6 sec
`pop_size-1000+elite-30`	390.73	393.75	1.58 sec
`pop_size-1000+elite-10`	550.56	550.63	1.51 sec
`pop_size-3000+elite-40`	400.15	751.57	3.42 sec
`pop_size-3000+elite-30`	161.26	222.73	3.17 sec
`pop_size-3000+elite-10`	228.41	229.67	3.04 sec
`iter-200`	1356.23	1356.23	1.09 sec
`iter-50`	1290.16	1290.59	0.69 sec
`iter-30`	1393.38	1394.41	0.66 sec
`iter-200+elite-30`	1229.94	1230.02	1.11 sec
`iter-200+pop_size-3000+elite-30`	101.37	103.35	6.42 sec

Table 9: Rastrigin CMA-ES performance

In Figure 18 is plotted the best and the worse fitness for each generation (averaged over 3 runs) for the baseline model. Below, in Figures 19 and 20 are plotted the fitness of the models that perform better with the rastrigin function.

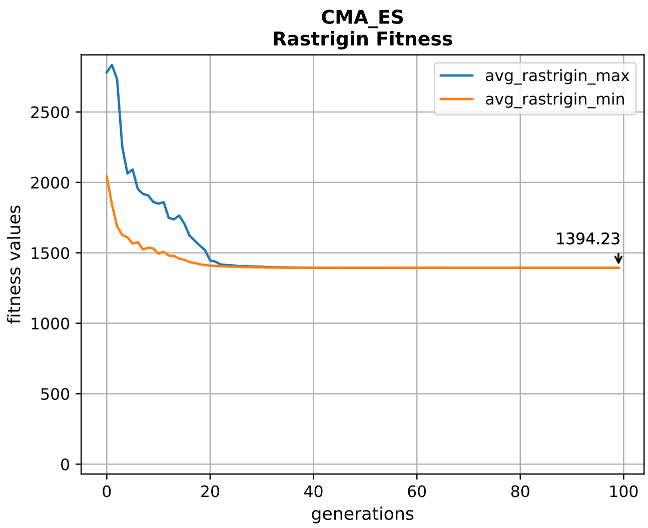


Figure 18: Rastrigin fitness `baseline`

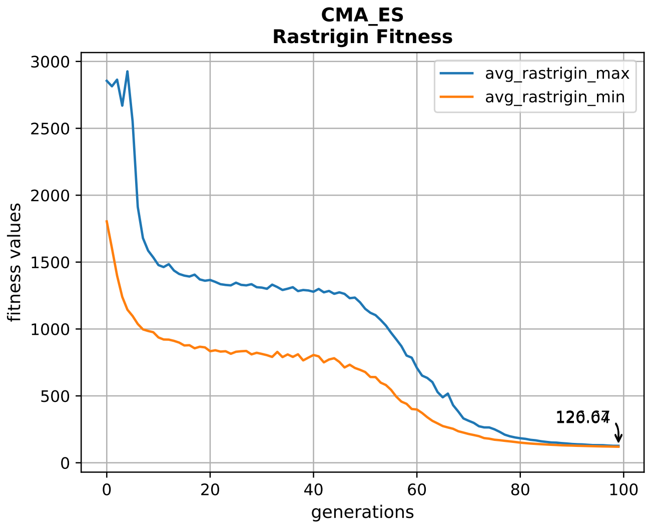


Figure 19: Rastrigin fitness `pop_size-3000`

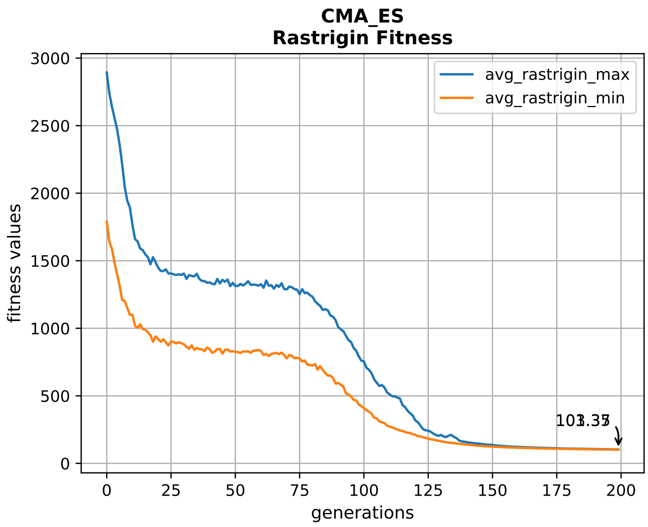
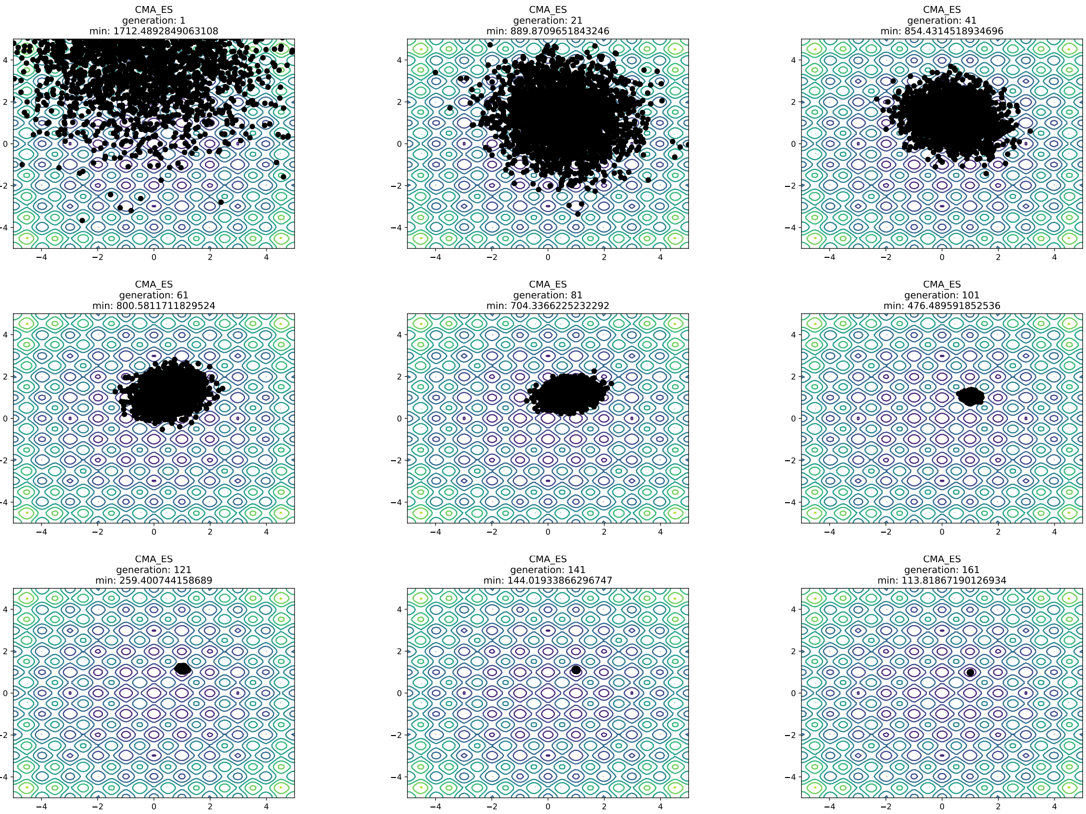


Figure 20: Rastrigin fitness `pop_size-3000+iter-200+elite-30`

In the following Figures is shown the effect of executing the algorithm on the experiment `pop_size-3000+iter-200+elite-30`.



For the sphere function, 40 generations are enough to obtain a solution close enough to the global optimum, as shown for the experiment `pop_size-3000`.

For the rastrigin function and the parameter tested, it was not possible to converge to a global optimum.

6 Benchmarking

The comparison has been carried out using the following parameters for all algorithms: domain size dimension of 100, 5000 population size, 2000 number of generations, elite set ratio of 0.20 and learning rate of 1e-3.

In Figures 21 and 22 are plotted the comparisons of CEM, NES and CMA-ES for the best fitness.

Regarding the sphere function, both CEM and CMA-ES perform well, while NES is much slower to converge. Ultimately, CMA-ES is better than CEM for the sphere function with these parameters.

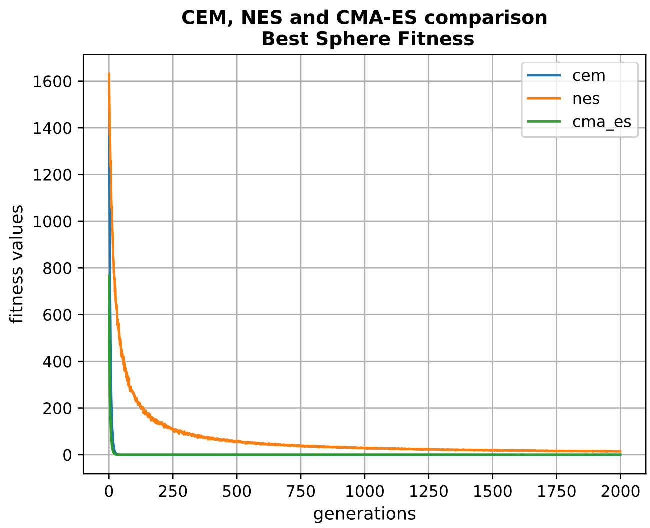


Figure 21: Comparison of sphere function with algorithm CEM, NES and CMA-ES for the best fitness

Regarding the rastrigin function, NES is definitely the worst algorithm. It is not able to converge to the global optimum. Both CEM and CMA-ES perform well. Initially, CEM goes down faster to the minimum, although it is not able to converge to 0 with these parameters, remaining only very close to the optimum. With about 200 iterations, CMA-ES converges to the global optimum, so it is certainly the best algorithm for this test function.

In Figures 21 and 22 are plotted the comparisons of CEM, NES and CMA-ES for the worst fitness.

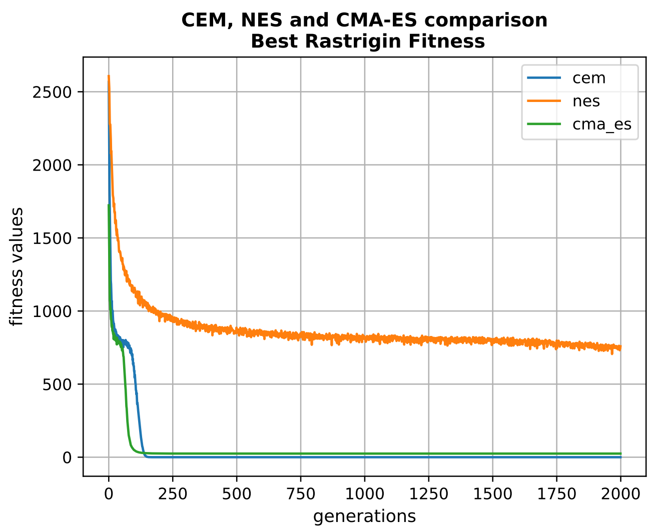


Figure 22: Comparison of CEM, NES and CMA-ES for the best rastrigin fitness

Regarding the sphere function, the algorithms behave as in the case of best fitness. Both CEM and CMA-ES perform well, while NES is much slower to converge.

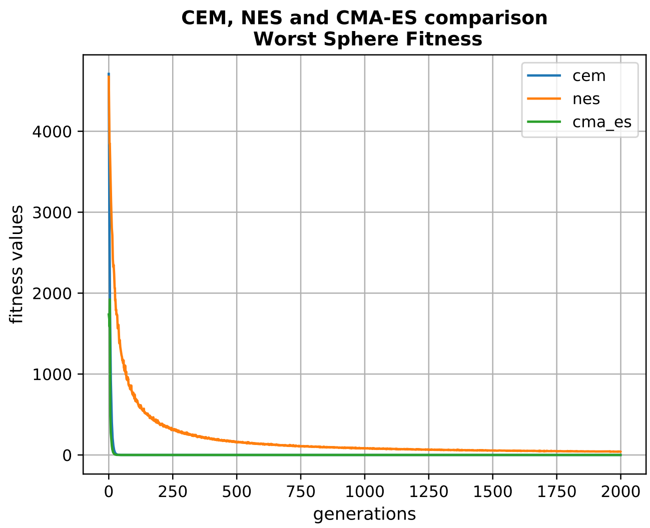


Figure 23: Comparison of sphere function with algorithm CEM, NES and CMA-ES for the worse fitness

Also for the rastrigin function, the algorithms behave as in the case of best fitness. NES is definitely the worst algorithm. It is not able to converge to the global optimum. Both CEM and CMA-ES perform well, although CMA-ES converges to the global optimum in about 150 generations.

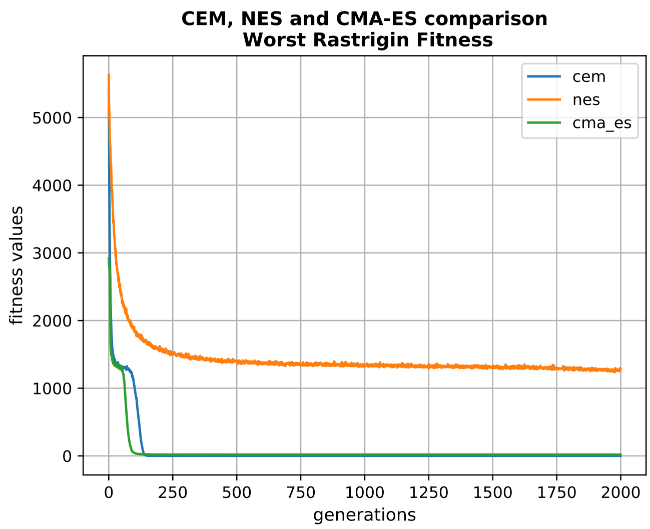


Figure 24: Comparison of CEM, NES and CMA-ES for the worse rastrigin fitness