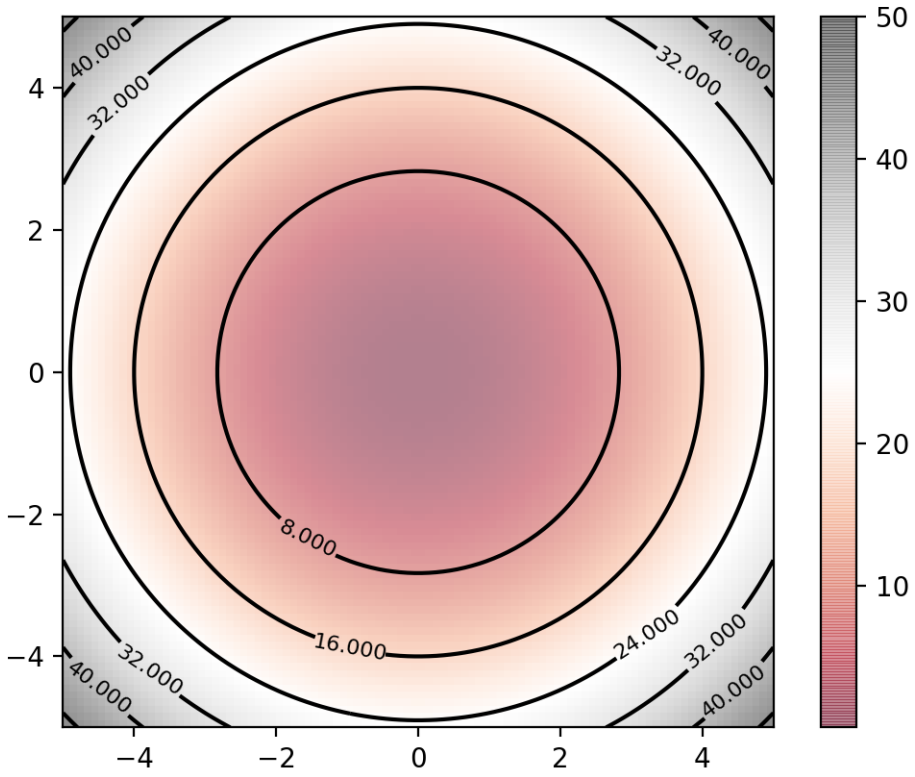
Alex Caselli Assignment 5

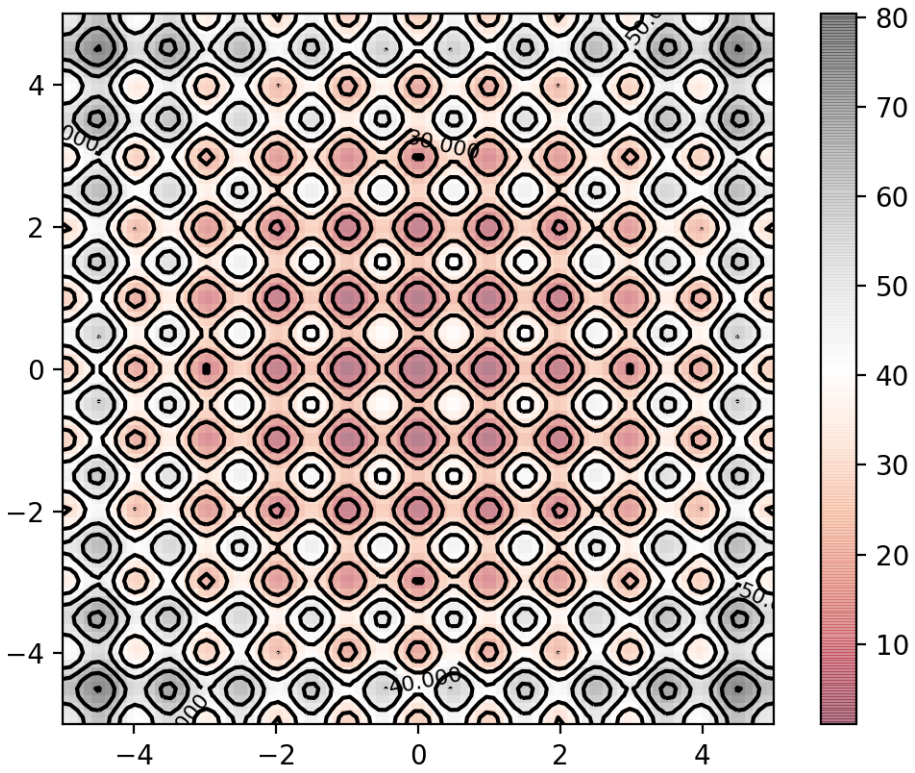
1 Exercise 1: Visualize test functions

Supposing a constraint in the search domain as $x_i \in [-5, 5]$

(a) Generate an 2D contour plot of 2-dimensional Sphere function (i.e. n = 2)



(b) Generate an 2D contour plot of 2-dimensional Rastrigin function (i.e. n = 2)



(c) For each test function, uniformly sample 100 points in the domain, evaluate them with the test function and guess what might be the region of the global optimum.

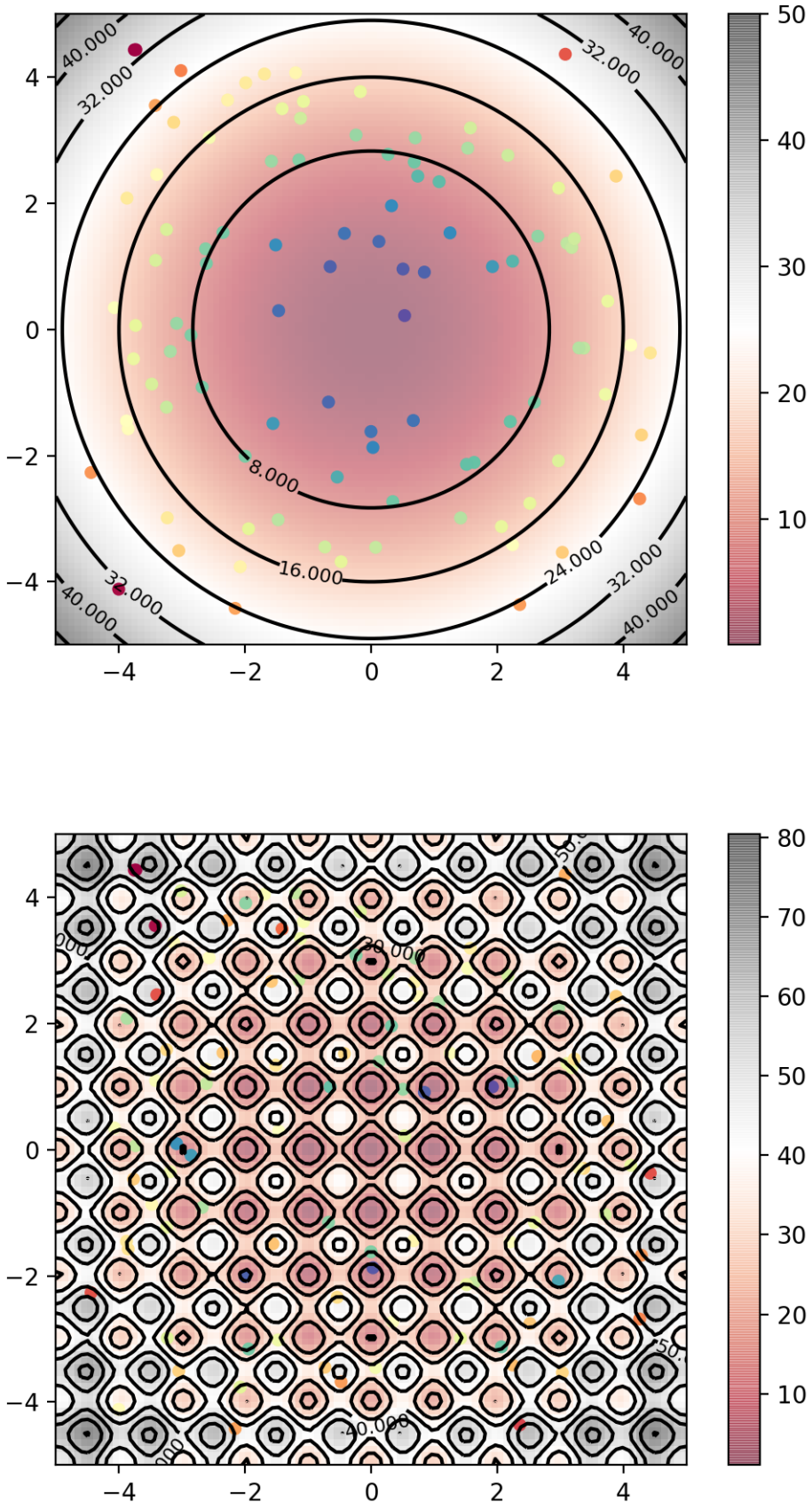


Figura 1: 100 points sampled on the Sphere function and on the Rastrigin function

By plotting the 100 samples on the contour plots we can easily see where the global optimum could be:

- In the Sphere function the global optimum is where both the components of the points are 0 and the value of the function in that point is 0.0.
- While in the Rastrigin we have multiple optima mainly in the range [-2, 2] for both the components. we can see these points in dark red very easily on the contour plot.

2 Exercise 2: Cross-Entropy Method (CEM)

Considering a 100-dimensional domain (n = 100) let's now try to find the best configuration in terms of population size and elite set ratio for the CEM.

1. **Sphere Function**

100 generations.		
Population	**Elite ratio**	**Mean fitness**
10	20	823.25
100	**20**	**0.0237**
100	10	2.19
1000	20	0.0203

From this test we can see that the combination of parameters which allow this algorithm to achieve the best score is a population size of 100 samples and elite set ratio of 20% even if the Mean fitness score is a little worst than what we got with a population of 1000. This different is to irrelevant considering that to achieve a little better fitness it require 10 times the samples.

2. **Rastrigin Function**

100 generations.		
Population	**Elite ratio**	**Mean fitness**
10	20	1389.3
100	**10**	**144.05**
100	20	165.51
1000	20	490.85
1000	10	197.66

With this function the algorithm seems to require different parameters to get closer to the global optimum, in particular with 100 iterations the best configuration seems to be with a population size of 100 samples and elite set ratio of 10%

Considering the best parameters founded, let's now try to find the best configuration in terms of number of generations for the CEM.

1. **Sphere Function**

100 samples, 20% Elite ratio.		
Generations	**Worst fitness**	**Best fitness**
100	0.0206	0.0091
200	0.0162	0.0078
500	**0.0150**	**0.0076**
1500	0.0195	0.0084

From these test we can see that the best set of parameter seems to be Population size of 100, 20% elite set ratio and a number of generations equal to 500 for the sphere function, in this way we can go quite close to the global optimum.

2. **Rastrigin Function**

100 samples, 20% Elite ratio.		
Generations	**Worst fitness**	**Best fitness**
100	**146.49**	**144.05**
1000	147.87	146.56
1000 samples, 20% Elite ratio.		
100	871.9	490.85
250	4.06	2.20
500	**3.20**	**1.25**
1500	3.70	1.86

Despite the initial test results on the best size of the population,from these test we can see that the best set of parameter is **500 generations of 1000 samples with an Elite ratio of 20%** for this function, we can go close to the global optimum but we are not able to get close enough, probably because there are many local optima also around our target optimum.

Let's plot now the Best and the worst fitness for both the functions:

1. **Sphere Function**

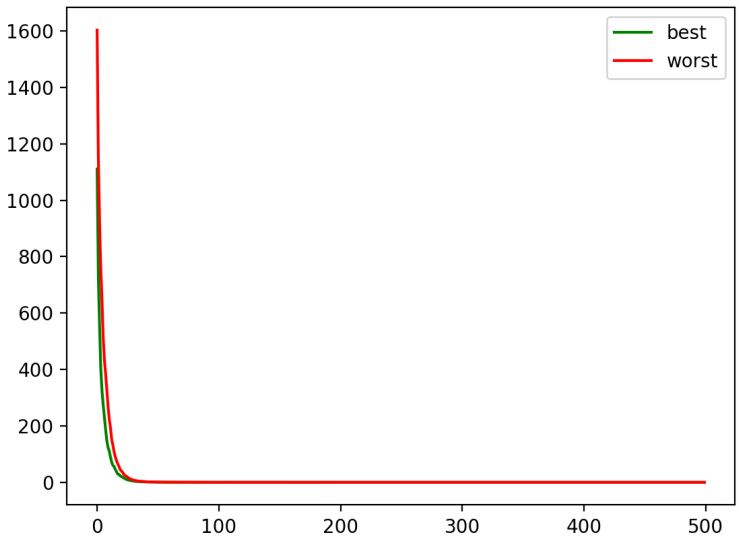


Figura 2: Best and Worst fitness on Sphere function obtained with the best parameters founded

2. **Rastrigin Function**

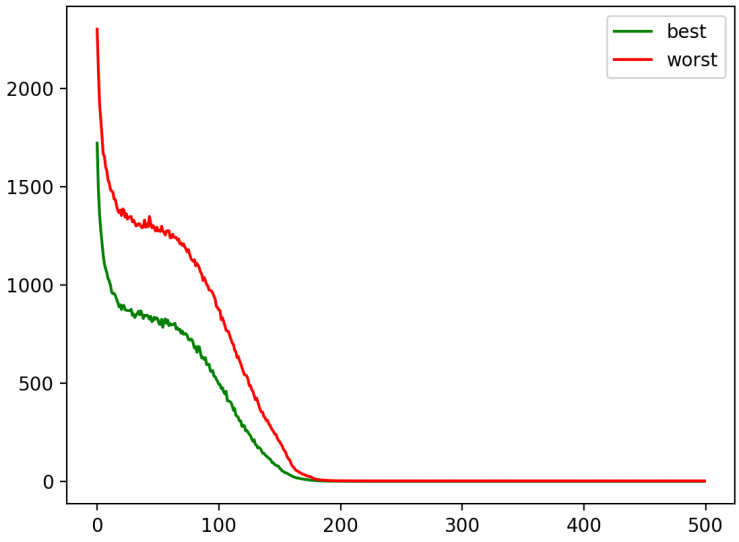


Figura 3: Best and Worst fitness on Rastrigin function obtained with the best parameters founded

3 Exercise 3: Natural Evolution Strategy (NES)

Considering a 100-dimensional domain (n = 100) let's now try to find the best configuration in terms of population size and learning rate for NES.

This Method seems to require a larger number of samples to work and a small learning rate, in fact NES produces errors with large learning rates and small population size, not all the possible combinations of values worked as we can see from the table below.

Also constrain the samples values in the range [-5, 5] makes the algorithm not working correctly in my tests. We can see this weird behaviour in the figure below.

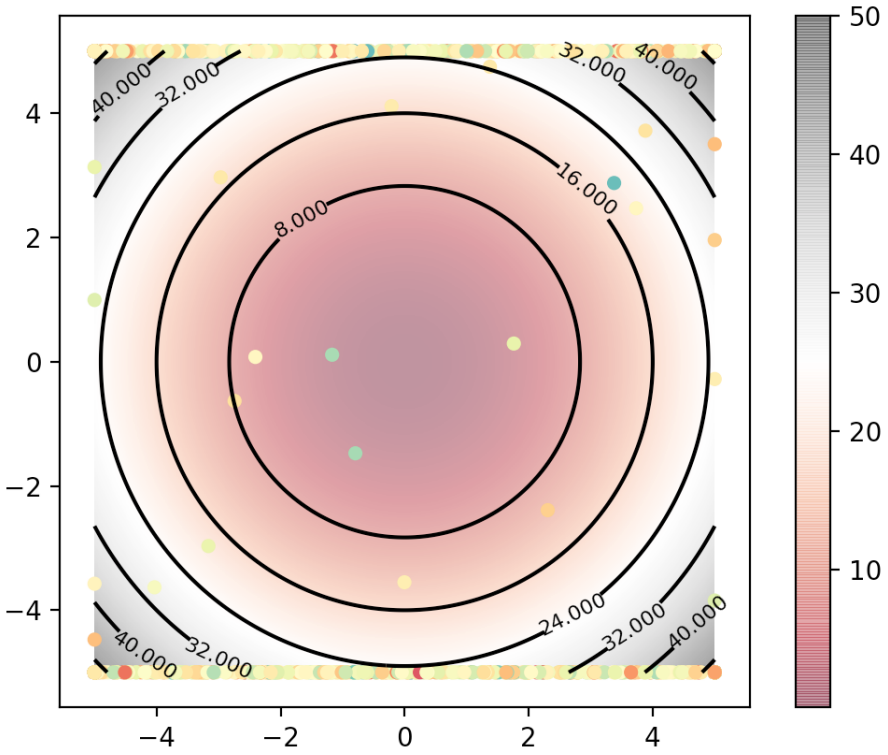


Figura 4: The points tend to go away from the global optimum and stop on the borders of the domain space.

1. **Sphere Function**

1000 generations.		
Population	**Learning rate**	**Mean fitness**
100	1e-3	Err
500	1e-3	Err
1000	1e-3	1354.3
1000	1e-2	Err
2000	1e-2	Err
2000	1e-3	1228.5
2000	1e-4	1593.2
3000	1e-3	1082.1
5000	**1e-3**	**727.5**
10000	1e-3	703.7

From this test we can see that with this algorithm, initializing Θ in the same way we did for CEM (mean between -5 and 5, std between 4 and 5), we are not able to get close to the optimum, that said the best combination of parameters i found takes a lot of time.

2. **Rastrigin Function**

1000 generations.		
Population	**Learning rate**	**Mean fitness**
100	1e-3	Err
500	1e-3	Err
1000	1e-3	Err
1000	1e-2	Err
2000	1e-2	Err
2000	1e-3	1894.0
2000	1e-4	2022.7
3000	1e-3	1838.3
5000	**1e-3**	**1788.1**

With this function the algorithm seems to work even worse, we achieve very bad results although the high computational time.

Considering the best parameters founded, let's now try to find the best configuration in terms of number of generations for the NES.

1. **Sphere Function**

5000 samples, 1e-3 Learning rate.		
Generations	**Worst fitness**	**Best fitness**
2000	1298.6	580.4

As we can see the performance are quite bad, go over 2000 iterations with this number of samples does not make any sense because it would take a lot of time without any nice improvement.

Instead we can try to initialize the mean at 0, and we can see that in this case the algorithm is able to get quite close to the global optimum, this suggest that NES performance are highly correlated with the initialization of Θ

1000 samples, 1e-3 Learning rate.		
Generations	**Worst fitness**	**Best fitness**
3000	75.7	25.6

We can achieve this results also initializing the mean with 5.0, the problem seems appear if we initialize the mean sampling from an uniform distribution between the range -5 and 5.

2. **Rastrigin Function**

5000 samples, 1e-3 Learning rate.		
Generations	**Worst fitness**	**Best fitness**
3000	2507.1	1509.7

Also with this function we are not able to get nice results, let's try again initializing the mean with 0.

3000 generations, 1e-3 Learning rate.		
Population	**Worst fitness**	**Best fitness**
2000	1543.5	949.3

In this case we are not able to get close to the global optimum, probably because of the complexity of the function.

Let's plot now the best and the worst fitness for both the functions with the best parameters we have found:

1. **Sphere Function**

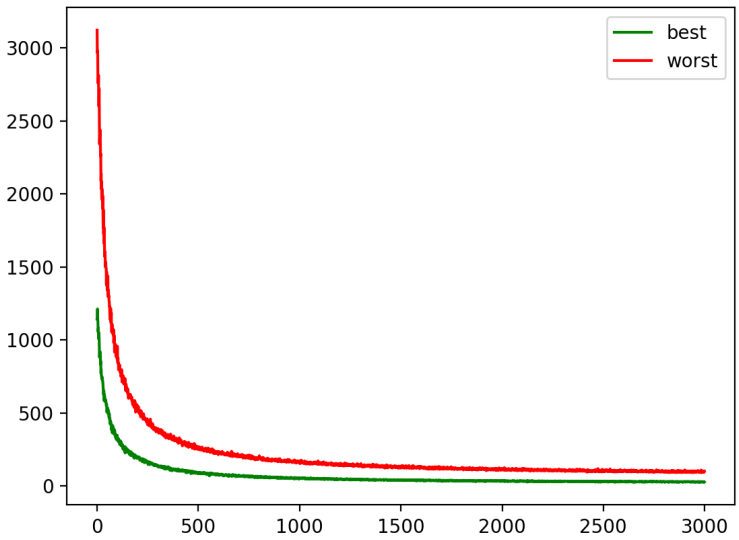


Figura 5: Best and Worst fitness on the Sphere function obtained with the best parameters founded

2. **Rastrigin Function**

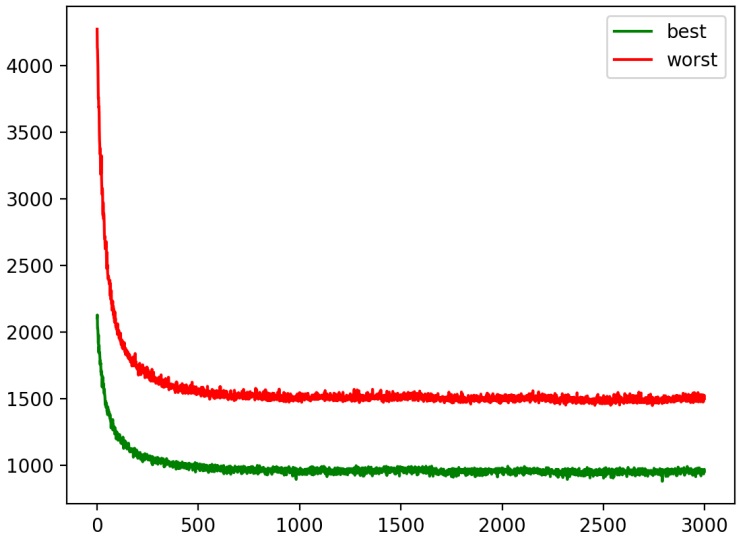


Figura 6: Best and Worst fitness on Rastrigin function obtained with the best parameters founded

4 Exercise 4: Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

Considering a 100-dimensional domain (n = 100) let's now try to find the best configuration in terms of population size and elite set ratio for the CMA-ES.

1. **Sphere Function**

100 generations.		
Population	**Elite ratio**	**Mean fitness**
10	20	349.8
100	10	42.14
100	20	48.36
1000	**10**	**0.187**
1000	20	0.253

From this test we can see that the combination of parameters which allow this algorithm to achieve the best score is a population size of 1000 samples and elite set ratio of 10%, CMA-ES seems to perform quite good getting close to the global optimum.

2. **Rastrigin Function**

100 generations.		
Population	**Elite ratio**	**Mean fitness**
10	20	996.4
100	10	483.3
100	20	422.3
1000	10	131.8
1000	20	101.3
2000	20	78.3
3000	**20**	**75.2**

With this function CMA-ES require a bigger population size, probably because it needs to escape from the many local optima of this function. The combination of parameters which allow this algorithm to achieve the best score is a population size of 3000 samples and elite set ratio of 20%, always considering a number of iterations fix to 100.

Considering the best parameters founded, let's now try to find the best configuration in terms of number of generations for the CMA-ES.

1. **Sphere Function**

1000 samples, 10% Elite ratio.		
Generations	**Worst fitness**	**Best fitness**
100	**0.566**	**0.187**
200	0.615	0.198
500	0.639	0.196

From these test we can see that we can't get closer than we get with 100 generations increasing the number of generations.

2. **Rastrigin Function**

3000 samples, 20% Elite ratio.		
Generations	**Worst fitness**	**Best fitness**
100	189.2	75.2
200	**167.9**	**69.87**
300	163.8	70.1

From these test we can see that the best set of parameter is **200 generations of 3000 samples with an Elite ratio of 20%** for this function, we can go close to the global optimum but we are not able to get close enough, probably because of the many local optima.

Let's plot now the Best and the worst fitness for both the functions:

1. **Sphere Function**

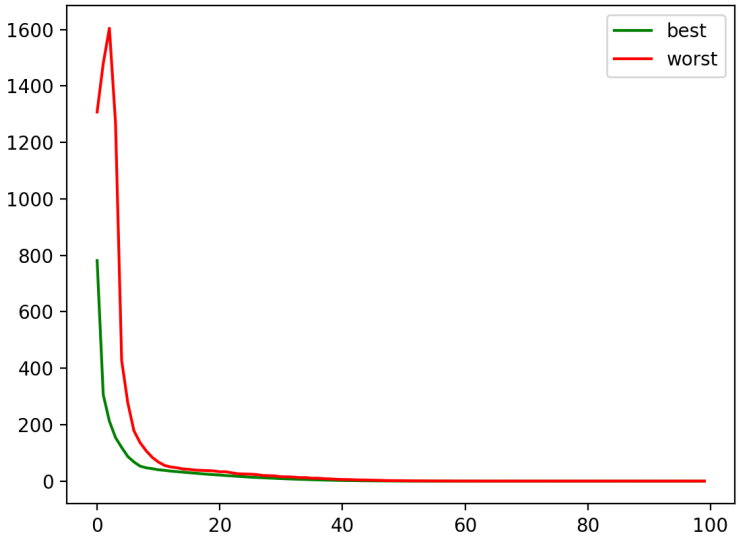


Figura 7: Best and Worst fitness on the Sphere function obtained with the best parameters founded

2. **Rastrigin Function**

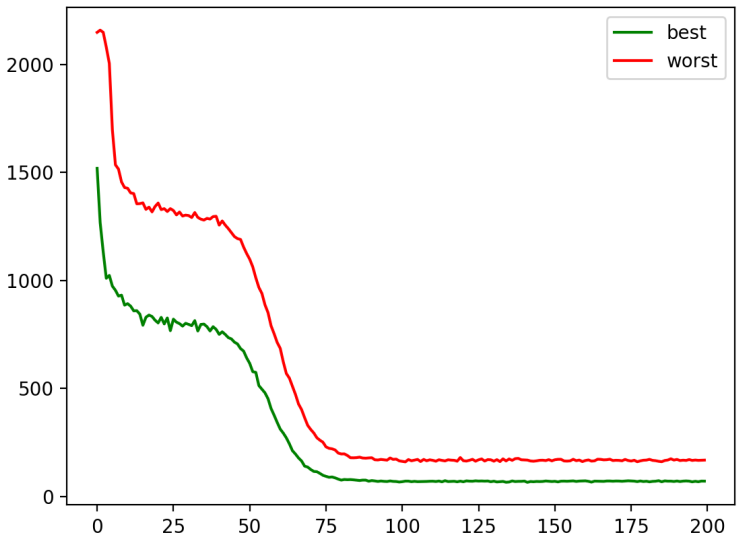


Figura 8: Best and Worst fitness on the Rastrigin function obtained with the best parameters founded

5 Benchmarking

(a) **Plot the comparison of CEM, NES and CMA-ES for the best fitness in each generation. Note that this should be one single figure with three curves, each for one algorithm.**

(a) **Sphere Function**

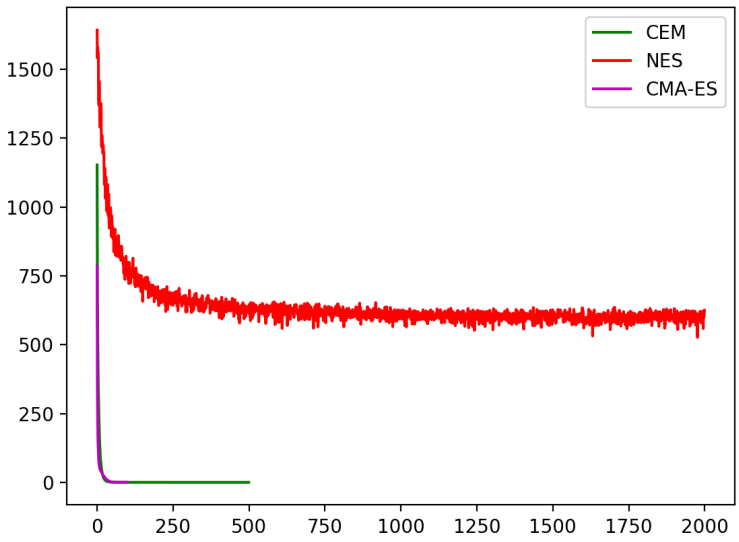


Figura 9: Best fitness obtained with the best parameter founded for each function

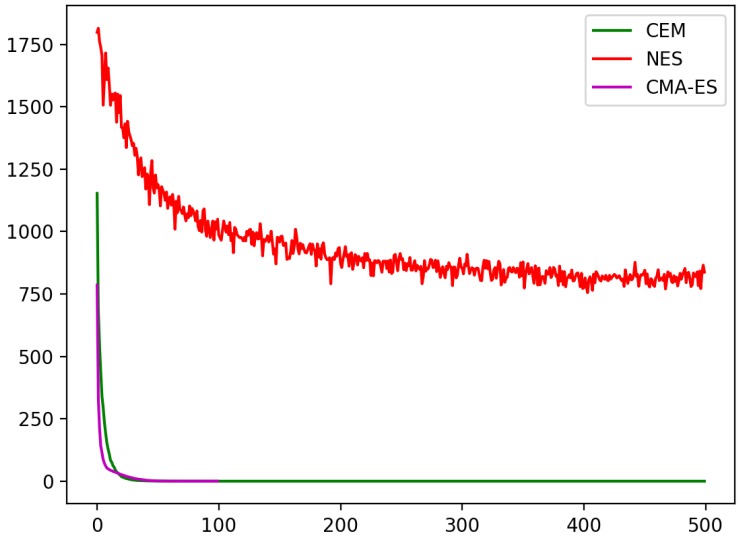


Figura 10: Best fitness obtained using very similar parameter for each function, to make differences more visible

(b) **Rastrigin Function**

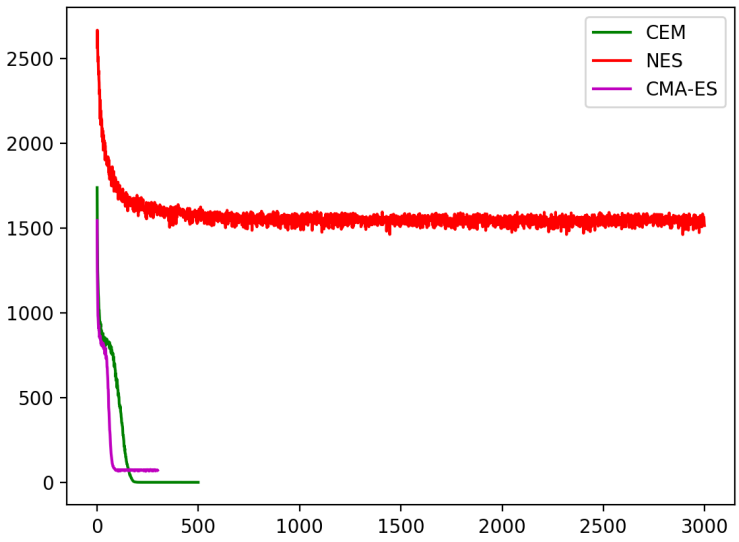


Figura 11: Best fitness obtained with the best parameter founded for each function

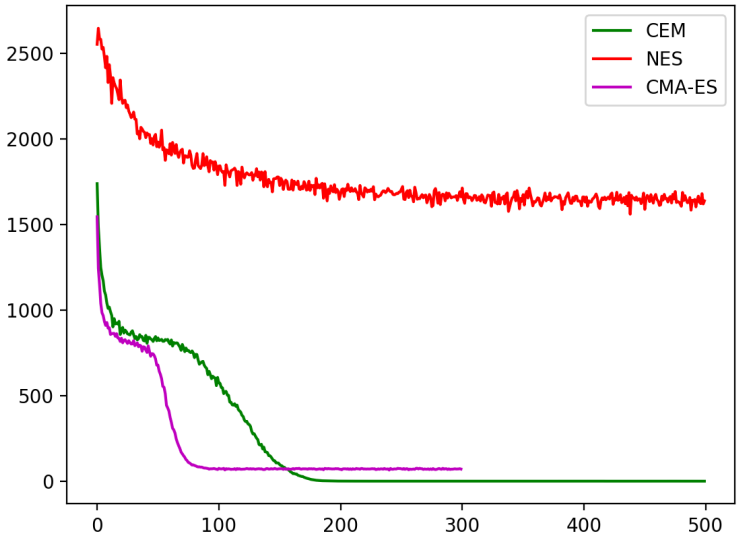


Figura 12: Best fitness obtained using very similar parameter for each function, to make differences more visible

(b) **Plot the comparison of CEM, NES and CMA-ES for the worst fitness in each generation. Note that this should be one single figure with three curves, each for one algorithm.**

(a) **Sphere Function**

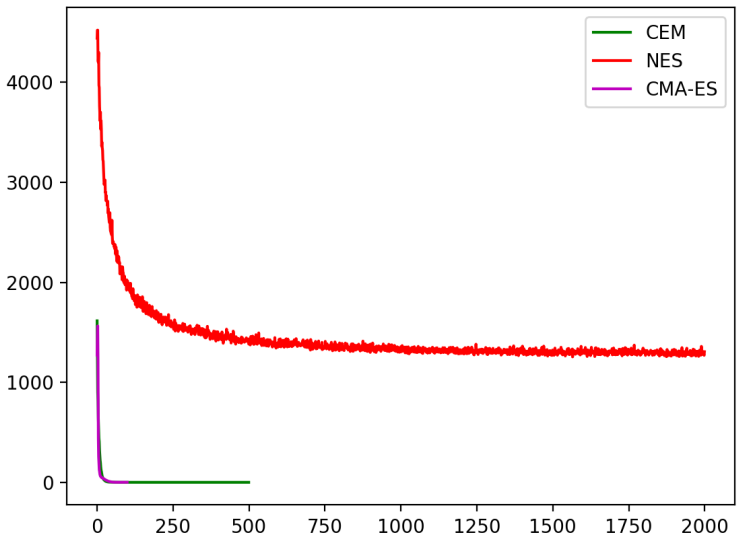


Figura 13: Worst fitness obtained with the best parameter founded for each function

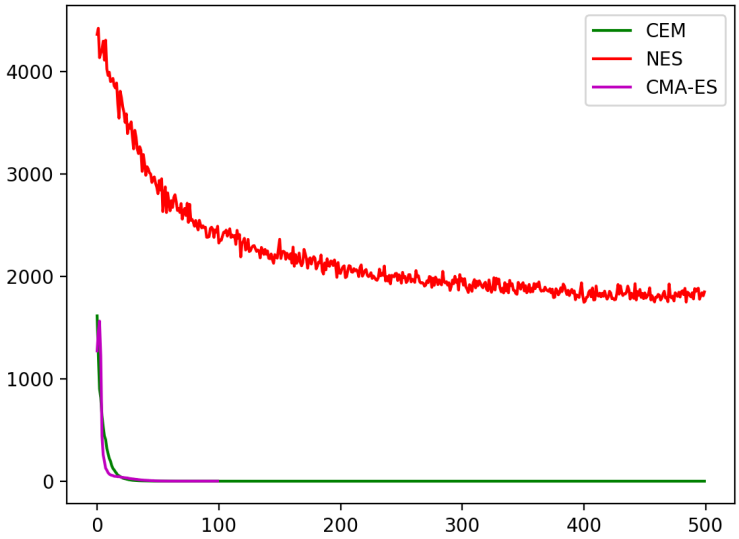


Figura 14: Worst fitness obtained using very similar parameter for each function, to make differences more visible

(b) Rastrigin Function

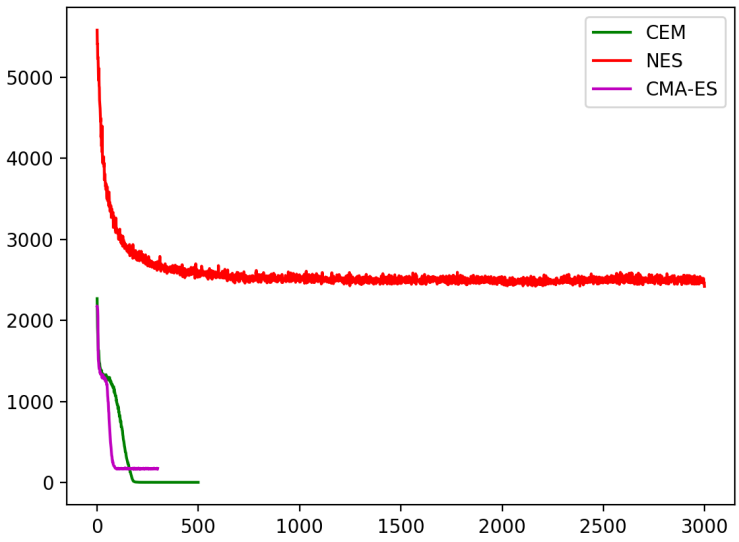


Figura 15: Worst fitness obtained with the best parameter founded for each function

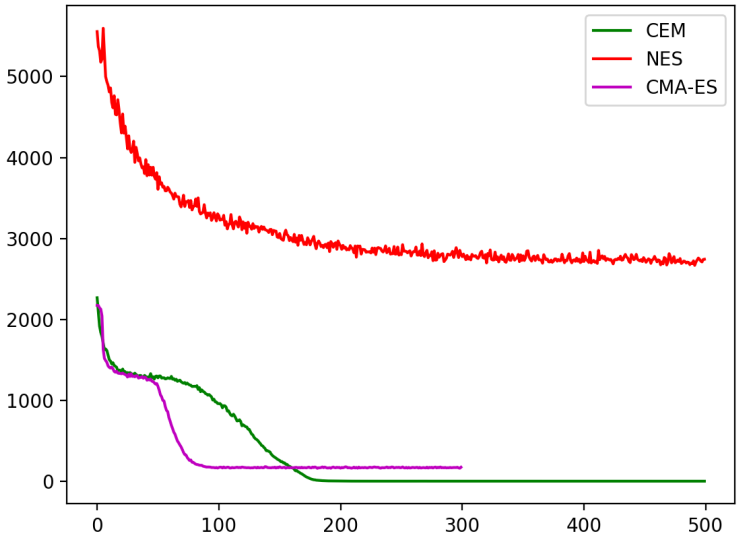


Figura 16: Worst fitness obtained using very similar parameter for each function, to make differences more visible

(c) **For each test function, which algorithm is best? which is the worst?** Based on the results obtained from the different tests the best algorithm seems to be the Cross Entropy Method (CEM). This algorithm was able to get really close to the global optimum in both the functions with a very little computing effort, in fact the execution time required for the CEM with the best parameters I found is around

60 seconds, quite similar to the time required by CMA-ES but many times less than what is required by NES. CEM is also very easily implementable and based on a very simple idea.

5.1 Extra

In the zip is possible to see some quick demo examples of the 3 algorithms running on the Sphere function.