Otimizations Based on Evolutionary Methods

Carlos Matheus Barros da Silva, *Bachelor Student of ITA*
Prof. Marcos Ricardo Omena de Albuquerque Máximo

***Abstract*—In this paper is the evaluation of an Evolutionary Optimization Method denominated Simple Evolution Strategy (SES) and its performance is compared with the Covariance Matrix Adaptation Evolution Strategy (CMA-ES).**

The results shows that since the SES approach not so sofisticated it needs a larger population in order to achieve as good resuls as CMA-ES gets with a smaller population.

Although in some cases they performed similary, and with a small increase in population in SES makes it perform as good as CMA-ES, and in some cases SES with a larger population performed better then CMA-ES. With this, therefore, SES is a simple and effective optimization method.

***Index Terms*—Simple Evolution Strategy, SES, Covariance Matrix Adaptation Evolution Strategy, CMA-ES, optimization**

I. Introduction

In computer science, an evolution strategy (ES) is an optimization technique based on ideas of evolution. It belongs to the general class of evolutionary computation or artificial evolution methodologies.

Usualy Evolution strategies use natural problem-dependent representations, and primarily mutation and selection, as search operators. In common with evolutionary algorithms, the operators are applied in a loop. An iteration of the loop is called a generation. The sequence of generations is continued until a termination criterion is met.

A. Simple Evolution Strategy

In this paper the Simple Evolution Strategy (SES) approach relies on a covariance matrix and a mean. For each itetation the algorithm makes a uniform sampling with the mean and covariance matrix and then gets the μ best positions and makes the mean of its position be the next mean and uptate the covariance matrix. This SES behavior can be described with the Equation 1 and Equation 2.

$$m^{g+1} = \frac{1}{\mu}\sum_{i=1}^{\mu} S_{i:\lambda}^{g+1} \tag{1}$$

$$C^{g+1} = \frac{1}{\mu}\sum_{i=1}^{\mu}(S_{i:\lambda}^{g+1} - m^g)(S_{i:\lambda}^{g+1} - m^g)^T \tag{2}$$

B. Covariance Matrix Adaptation Evolution Strategy

CMA-ES stands for covariance matrix adaptation evolution strategy. Evolution strategies (ES) are stochastic, derivative-free methods for numerical optimization of non-linear or non-convex continuous optimization problems. They belong to the class of evolutionary algorithms and evolutionary computation. An evolutionary algorithm is broadly based on the principle of biological evolution, namely the repeated interplay of variation (via recombination and mutation) and selection: in each generation (iteration) new individuals (candidate solutions, denoted as x) are generated by variation, usually in a stochastic way, of the current parental individuals. Then, some individuals are selected to become the parents in the next generation based on their fitness or objective function value $f(x)$. Like this, over the generation sequence, individuals with better and better f-values are generated.

In an evolution strategy, new candidate solutions are sampled according to a multivariate normal distribution in R^n. Recombination amounts to selecting a new mean value for the distribution. Mutation amounts to adding a random vector, a perturbation with zero mean. Pairwise dependencies between the variables in the distribution are represented by a covariance matrix. The covariance matrix adaptation (CMA) is a method to update the covariance matrix of this distribution. This is particularly useful if the function f is ill-conditioned.

Two main principles for the adaptation of parameters of the search distribution are exploited in the CMA-ES algorithm.

First, a maximum-likelihood principle, based on the idea to increase the probability of successful candidate solutions and search steps. The mean of the distribution is updated such that the likelihood of previously successful candidate solutions is maximized. The covariance matrix of the distribution is updated (incrementally) such that the likelihood of previously successful search steps is increased. Both updates can be interpreted as a natural gradient descent. Also, in consequence, the CMA conducts an iterated principal components analysis of successful search steps while retaining all principal axes. Estimation of distribution algorithms and the Cross-Entropy Method are based on very similar ideas, but estimate (non-incrementally) the covariance matrix by maximizing the likelihood of successful solution points instead of successful search steps.

Second, two paths of the time evolution of the distribution mean of the strategy are recorded, called search or evolution paths. These paths contain significant information about the correlation between consecutive steps. Specifically, if consecutive steps are taken in a similar direction, the evolution paths become long. The evolution paths are exploited in two ways. One path is used for the covariance matrix adaptation procedure in place of single successful search steps and facilitates a possibly much faster variance increase of favorable directions. The other path is used to conduct an additional step-size control. This step-size control aims to make consecutive movements of the distribution mean orthogonal in expectation. The step-size control effectively prevents premature convergence yet allowing fast convergence to an optimum.

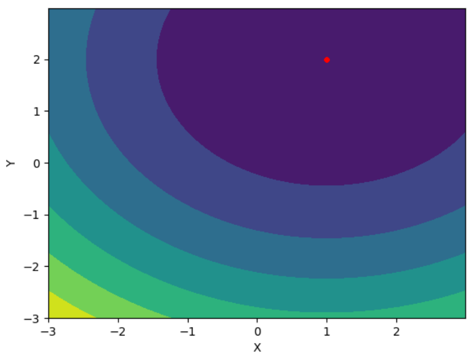

Fig. 1. Convergence of SES on Translated Sphere function

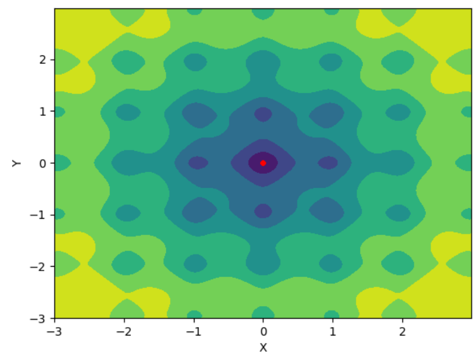

Fig. 2. Convergence of SES on Translated Sphere function

II. Simple Evolution Strategy (SES) Implementation

The implementation was based on the file *simple evolution strategy*. The essence of the implementation is on *SimpleEvolutionStrategy* Class and on *tell* method.

The *tell* method updates the mean and the coraviance matrix of SES and its implementation can be seen on Code 1.

```
def tell(self, fitnesses):
    """
    Tells the algorithm the evaluated fitnesses.
    The order of the fitnesses in this array
    must respect the order of the samples.
    :param fitnesses: array containing the value
    of fitness of each sample.
    :type fitnesses: numpy array of floats.
    """
    indices = np.argsort(fitnesses)
    best_samples = self.samples[indices[0:self.mu], :]
    array = self.C * 0
    for idx in range(self.mu):
        mt = best_samples[idx] - self.m
        mt = np.matrix(mt)
        transpose = mt.transpose()
        array += (transpose*mt)
    self.C = array / self.mu
    self.m = sum(best_samples)/self.mu
    self.samples = np.random.multivariate_normal(self.m
    , self.C, self.population_size)
```

Code 1. Code of *tell* method

III. SES performance analysis

In this analysis was used four functions: Translated Sphere, Ackley, Schaffer function, Rastrigin (2D).

SES performed very well in most case finding the minimum global. In Translated Sphere and Ackley using the parameters defined on the file test evolution strategy the result of SES for these two test functions was similar to the result of CMA-ES.

On the images Image 1 and Image 2 is shown the convergence of SES on these two test functions. And for these two tests the codes from Code 2 to Code 5 shown its fitness and sample progresses.

```
001: 3.4163140057961123
002: 0.4856090879492764
003: 0.4736399744586906
004: 0.13243457399923464
005: 0.007752869225068693
006: 0.014563762862005922
007: 0.014096386253612033
008: 0.014603284480635267
009: 0.009003339595536274
010: 0.009283704315168865
...
191: 0.0
192: 0.0
193: 0.0
194: 0.0
195: 0.0
196: 0.0
197: 0.0
198: 0.0
199: 0.0
200: 0.0
```

Code 2. First 10 and last 10 fitness results of SES on Translated Sphere function

```
001: [2.31756678 0.04204495]
002: [ 1.81706685 -1.25139784]
003: [1.66124008 0.17654011]
004: [ 2.49384662 -1.35622335]
005: [0.99374436 0.78897232]
006: [0.27929801 2.12855099]
007: [0.88621959 1.95929002]
008: [0.67694663 1.88000947]
009: [0.83539439 2.06674017]
010: [0.85522325 1.94730609]
...
191: [1. 2.]
192: [1. 2.]
193: [1. 2.]
194: [1. 2.]
195: [1. 2.]
196: [1. 2.]
197: [1. 2.]
198: [1. 2.]
199: [1. 2.]
200: [1. 2.]
```

Code 3. First 10 and last 10 samples of SES on Translated Sphere function

```
001: 7.99575793183242
002: 6.187956577458923
003: 4.153919874475818
004: 3.180270116478958
005: 0.14533429920713203
006: 3.882400415732498
007: 1.9665710380619075
008: 2.617890144090879
009: 0.6152172492689196
010: 1.9856709214708275
...
191: 0.0
192: 0.0
193: 0.0
194: 0.0
195: 0.0
196: 0.0
197: 0.0
198: 0.0
199: 0.0
```

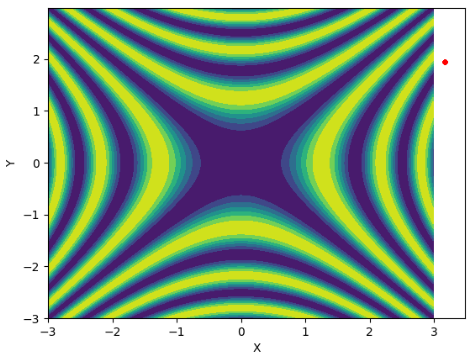

Fig. 3. Convergence of SES on Schaffer function

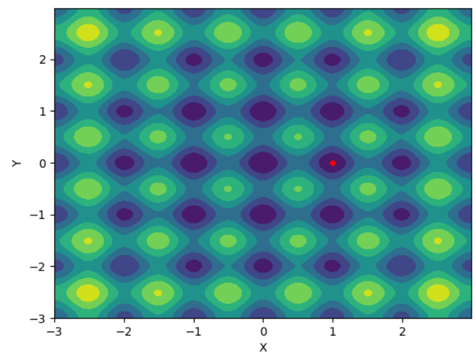

Fig. 4. Convergence of SES on Rastrigin (2D) function

```
200: 0.0
```

Code 4. First 10 and last 10 fitness results of SES on Ackley function

```
001: [ 2.42977745 -0.83179142]
002: [ 1.42824907 -1.07409605]
003: [ 0.96170699 -0.77810187]
004: [0.53792831 0.03003805]
005: [-0.03724346 0.00738052]
006: [-1.0386611 0.87975928]
007: [-0.21195263 0.1568565 ]
008: [-0.0427703 0.37445813]
009: [0.10947725 0.02082328]
010: [-0.0639214 0.26807859]
...
191: [ 5.87363261e-17 -5.88660696e-17]
192: [ 6.27725547e-17 -4.92242539e-17]
193: [ 6.32111451e-17 -5.40652829e-17]
194: [ 5.99962031e-17 -3.77014351e-17]
195: [ 6.14066606e-17 -3.91811845e-17]
196: [ 6.25888185e-17 -4.83790439e-17]
197: [ 6.61969119e-17 -5.21704645e-17]
198: [ 6.78664170e-17 -6.14891389e-17]
199: [ 7.42745732e-17 -4.83778917e-17]
200: [ 6.50809040e-17 -4.04812879e-17]
```

Code 5. First 10 and last 10 samples of SES on Ackley function

But it was verified as well that for more dificult functions like Schaffer function and Rastrigin (2D) the SES can easily be stocked on a local minimum. The images Image 3 and Image 4 shows Schaffer and Rastrigin respectively. And for these two tests the codes from Code 6 to Code 9 shown its fitness and sample progresses.

```
001: 0.018113530433598368
002: 0.006906375715666879
003: 0.05378057289220978
004: 0.010535249980094918
005: 0.04467804051169594
006: 0.01075981718129393
007: 0.017325061706229383
008: 0.008707117460351743
009: 0.03125471014254144
010: 0.026476375760913773
...
191: 0.013591027644826004
192: 0.013591027212420725
193: 0.013591024258682971
194: 0.013591021561958294
195: 0.013591021186900476
196: 0.013591021827416283
197: 0.013591019618472311
198: 0.013591018186265125
199: 0.013591017082358325
200: 0.013591017827301433
```

Code 6. First 10 and last 10 fitness results of SES on Schaffer function

```
001: [2.75977226 2.11754802]
002: [4.30968882 0.8023816 ]
003: [4.9188067 1.76377647]
004: [2.87923209 2.5876754 ]
005: [-1.69160523 0.42984834]
006: [4.97782558 3.78282123]
007: [5.75074779 3.2687313 ]
008: [4.16761473 4.07309245]
009: [4.4486064 2.79289268]
010: [4.39789713 2.53148571]
...
191: [3.17472796 1.94802696]
192: [3.17472775 1.94802663]
193: [3.17472749 1.94802623]
194: [3.17472752 1.94802628]
195: [3.17472751 1.94802626]
196: [3.17472736 1.94802603]
197: [3.17472722 1.9480258 ]
198: [3.17472728 1.94802589]
199: [3.1747274 1.94802609]
200: [3.17472701 1.94802548]
```

Code 7. First 10 and last 10 samples of SES on Schaffer function

```
001: 6.718339753353399
002: 12.622005755494676
003: 11.9512056250556
004: 4.18502622565544
005: 18.534497695451
006: 23.837760285033053
007: 18.746050927963537
008: 16.43934246899969
009: 14.19279655566823
010: 10.247492723558947
...
191: 7.959662381108175
192: 7.959662381108174
193: 7.959662381108174
194: 7.959662381108174
195: 7.959662381108174
196: 7.959662381108174
197: 7.959662381108174
198: 7.959662381108174
199: 7.959662381108174
200: 7.959662381108174
```

Code 8. First 10 and last 10 fitness results of SES on Rastrigin (2D) function

```
001: [1.02400164 0.83596626]
002: [0.98004935 1.25031446]
003: [0.26345237 1.00961533]
004: [2.02155056 0.00586581]
005: [1.73738226 0.85246949]
006: [1.18122247 0.64322307]
007: [1.76289741 1.87492214]
008: [1.76017384 1.9906414 ]
009: [0.97391809 3.13302776]
010: [1.00521085 2.94717065]
...
191: [1.98991223 1.98991223]
192: [1.98991223 1.98991223]
193: [1.98991223 1.98991223]
```

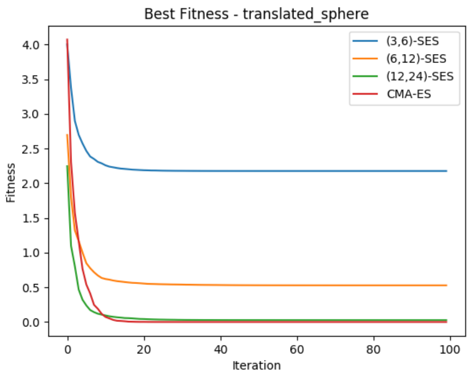

Fig. 5. SES and CMA-ES best fitness comparison on Translated Sphere function

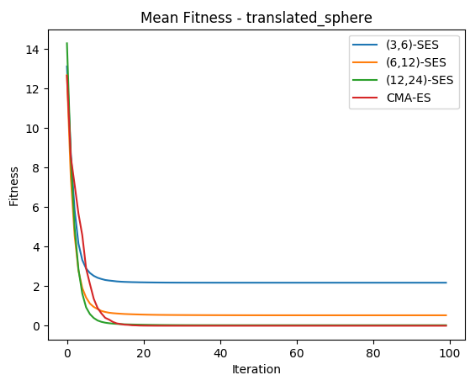

Fig. 6. SES and CMA-ES mean fitness comparison on Translated Sphere function

```
194: [1.98991223 1.98991223]
195: [1.98991223 1.98991223]
196: [1.98991223 1.98991223]
197: [1.98991223 1.98991223]
198: [1.98991223 1.98991223]
199: [1.98991223 1.98991223]
200: [1.98991223 1.98991223]
```

Code 9. First 10 and last 10 samples of SES on Rastrigin (2D) function

A. SES and CMA-ES comparison

For test functions like Translated Sphere and Ackley it is evident that CMA-ES performs much better than SES even with a lower population.

Tests using a population of size 6 for CMA-ES and 3 diferent populations can be seen in the images Image 5 to Image 8 for these 2 test functions.

In these cases it is clear how SES needs a population four times CMA-ES in order to get similar performance.

Although when the comparison is made using more dificult test function like Schaffer function and Rastrigin (2D) the diference gap between SES and CMA-ES decrease and in some cases CMA-ES performs worse than SES with a slight bigger population.

This shows that for dificult functions it may be important to increase the CMA-ES population in order to get a good performance. And shows as well that SES can have a compareble performance to CMA-ES in some cases.

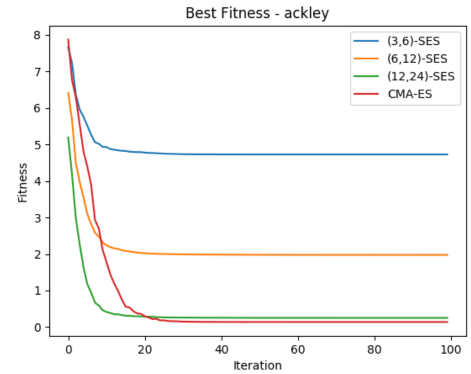

Fig. 7. SES and CMA-ES best fitness comparison on Ackley function

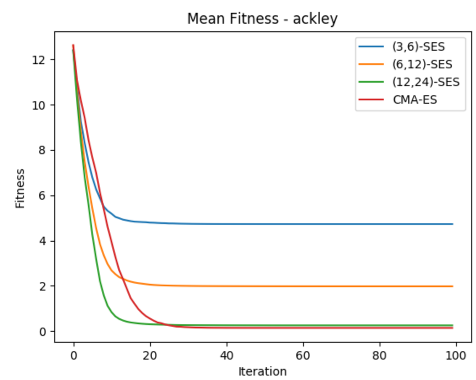

Fig. 8. SES and CMA-ES mean fitness comparison on Ackley function

Tests using a population of size 6 for CMA-ES and 3 diferent populations can be seen in the images Image 9 to Image 12 for these 2 test functions.

IV. Conclusion

It was clear, therefore, that since the SES approach not so sofisticated it needs a larger population in order to achieve as

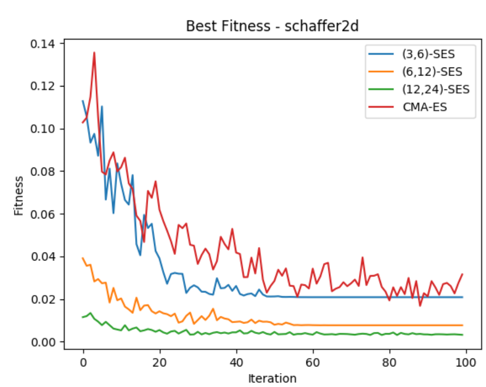

Fig. 9. SES and CMA-ES best fitness comparison on Schaffer function

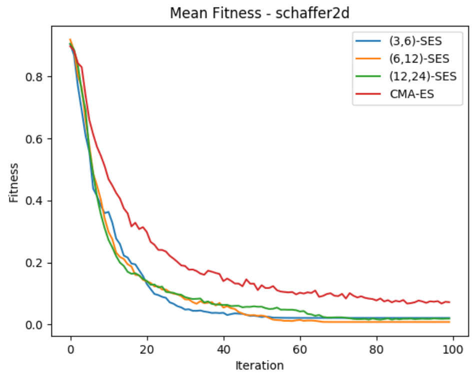

Fig. 10. SES and CMA-ES mean fitness comparison on Schaffer function

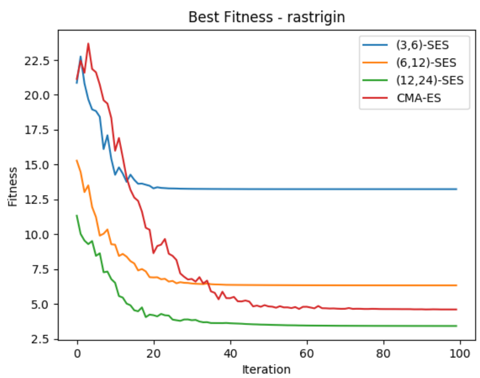

Fig. 11. SES and CMA-ES best fitness comparison on Rastrigin function

good resuls as CMA-ES gets with a smaller population.

Although in some cases they performed similary, and with a small increase in population in SES makes it perform as good as CMA-ES, and in some cases SES with a larger population performed better then CMA-ES. With this, therefore, SES is a simple and effective optimization method.

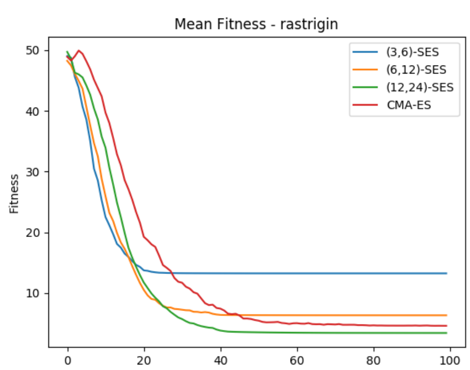

Fig. 12. SES and CMA-ES mean fitness comparison on Rastrigin function