Evolution Strategies

Ricardo Navares

Abstract

This is a step by step guide to Evolution Strategies applied to function optimization. Several evolution schemes and strategies were implemented over a set of classic functions to check the performance of the method.

Introduction

The objective of the implementation of the evolution strategy is to optimize (maximize / minimize) a set of proposed functions. Notice that a minimization problem is equivalent to a maximization problem as it applies: $f_{max}(x) = -f_{min}(-x)$. Evolution strategies start from a set of initial solutions which evolve generation by generation until a predifined maximum number of generations N is reached:

1. Initialize the population: $\Theta_{\mu}^{t=0}(x_i, \sigma_i, f(x_i))$
2. Recombine the parent population μ to generate the population λ: $\Theta_{\lambda}^{t}(\Theta_{\mu}^{t})$
3. Mutate population λ: Θ_{λ}^{tm}
4. Select μ individuals $\Theta_{\mu}^{t+1}(\Theta_{\mu}^{t}, \Theta_{\lambda}^{tm})$
5. Increment t and repeat 2-5 for N generations

Where $\Theta_{\mu}^{t}(x_i, \sigma_i, f(x_i))$ is the parent population at t (or generation t). This population consists on μ solutions x_i, each of which has a σ_i associated. This σ_i can be interpret as the strength of mutation. Finally, the fitness function is represented by $f(x_i)$. We will explain each member in detail in the following sections.

Population initialization was implemented from a uniform random distribution within the limits predefined by the user, which also define the search range. Thus, for each of the μ individuals in the population, a random number is generated per number of idependent variables in the objective function. Sigma initialization is provided by the user and its value is defined according to the initial problem.

October 11, 2017

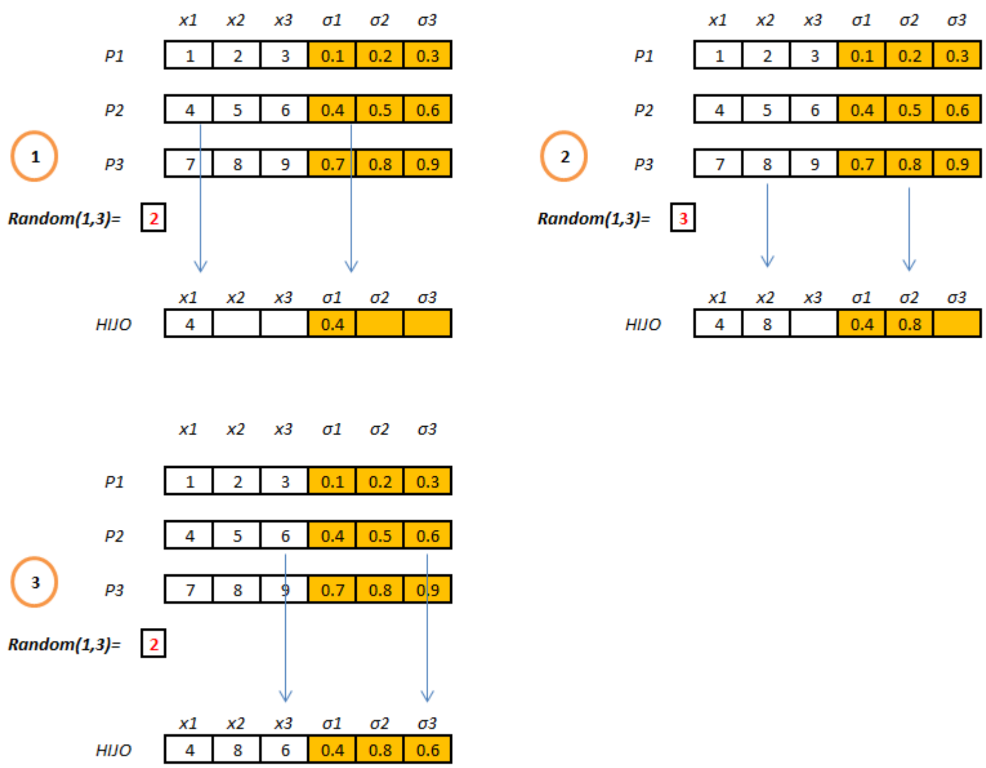


Figure 1: Discrete combination example. Legend: HIJO = child

Recombination

Two types of recombination were implemented: discrete and itermediate enabling the user to define the number of parents (ρ) involved in the generation of each individual. Parent selection is randomly done among the μ candidates.

Discrete recombination. Among the candidates, a tuple $< x_i, \sigma_i >$ is randomly selected and directly transferred to the child. Consider a $\rho = 3$, Figure 1 shows how each tuple is randomly selected among the ρ parents. Notice that it could happen the situation of a parent not being randomly selected to generate any tuple of the child.

Intermediate recombination. Given the ρ parents from the parent population of μ individuals, this time what it is transfered to the child is the mean of the values of the parents. Thus, the value x_i of the child is defined by:

$$H(x_i) = \frac{1}{\rho} \sum_{j=1..\rho} P_j(x_i) \tag{1}$$

being $P(x_i)$ the value of x_i from parent j. For example, in Figure 1 the value of x_1 in the first step is:

$$H(x_1) = \frac{1}{\rho} \sum (P_1(x_1) + P_2(x_2) + P_3(x_3)) = \frac{1}{3}(1 + 4 + 7) \tag{2}$$

Notice that σ represents standard deviations so we cannot directly apply the mean in order to transfer it. However, the variance needs to be used:

$$H(\sigma_i) = \sqrt{\frac{1}{\rho} \sum_{j=1..\rho} P_j(\sigma_i^2)} \tag{3}$$

which, continuing with the example:

$$H(\sigma_i) = \sqrt{\frac{1}{3}(0.1^2 + 0.4^2 + 0.7^2)} \tag{4}$$

Mutation

This is the most important step of this kind of strategies as it encapsulates population diversity. As opposed to genetic algorithms, individuals do not mutate with certain probability but they always do so. Two schemes were implemented:

Uncorrelated with 1 step. This schema modifies each sigma of an individual by scaling them by the same factor τ. Given a child λ it applies $\sigma_{\lambda 1} = \sigma_{\lambda 2} = ... = \sigma_{\lambda n}$, being n the number of variables of the problem. The steps are as follows:

1. Generate the disturbance: $r = e^{\tau \cdot N(0,1)}$
2. For each sigma in the individual: ${\sigma_i}' = \sigma_i \cdot r$
3. For each variable: ${x_i}' = x_i + {\sigma_i}' \cdot N(0, 1)$

Wherre $N(0, 1)$ is a random number generated from a normal distribution. The parameter τ takes as a default a value of $\tau \sim \frac{1}{\sqrt{n}}$ where n is the number of variables of the problem.

Uncorrelated with N steps. As opposed to the previous schema, this one modifies each sigma by a unique factor on each generation:

(a) (μ, λ) (b) (μ, λ)

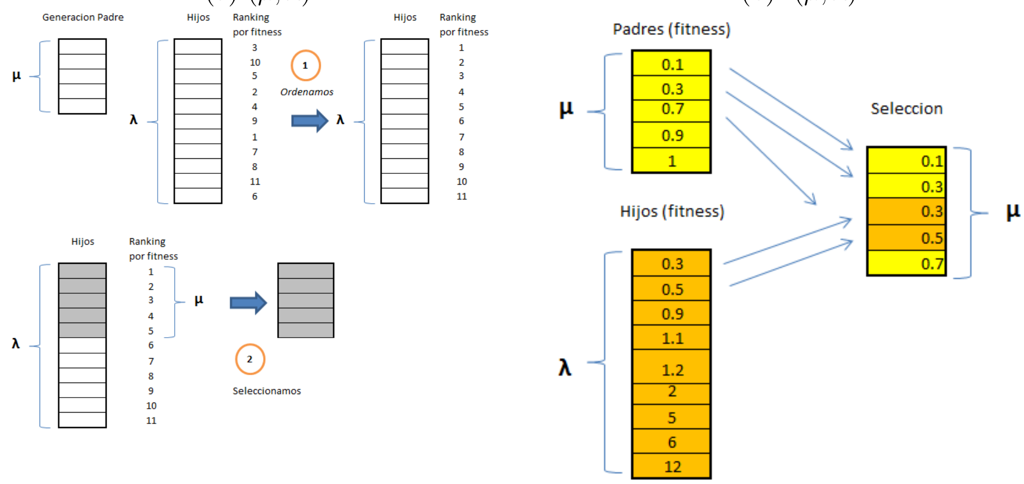


Figure 2: Selection schemes

1. Generate one disturbance per sigma: $r_i = e^{\tau' \cdot N(0,1) + \tau \cdot N(0,1)}$
2. Mutate the sigmas: $\sigma_i{}' = \sigma_i \cdot r_i$
3. Mutate each variable: $x_i{}' = x_i + \sigma_i{}' \cdot N(0,1)$

Having the new parameter τ' a default value of $\tau' \sim \frac{1}{\sqrt{2}}\sqrt{\tau}$.

In both schemas, there is an extra paramater ϵ defined by the user which controls the sufficiency of the mutation by capping the value of sigma: if $\sigma_i{}' < \epsilon \rightarrow \sigma_i{}' = \epsilon$

Selection

Selection (μ, λ). After previous recombination and mutation steps, the algorithm has a parent population of size μ and a generated child population of size λ. This selection scheme directly gets the best μ childs from the λ population (Figure 2 (a). Notice that this requires $\mu \leq \lambda$.

Selection $(\mu + \lambda)$. This time the selection is done from the μ best candidates among both (parent and child) populations (Figure 2 (b)).

Test Functions

Three classic test functions were selected in order to test the evolution strategy. Per function, 30 simulations were performed with the input parameters detailed below.

(a) Sphere Function

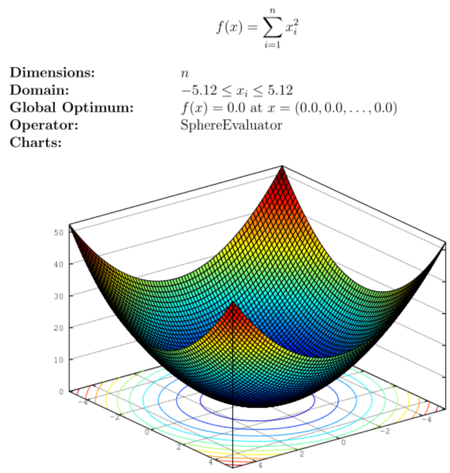


Figure 10: Sphere function [-5.12, 5.12].

(b) Simulations convergence

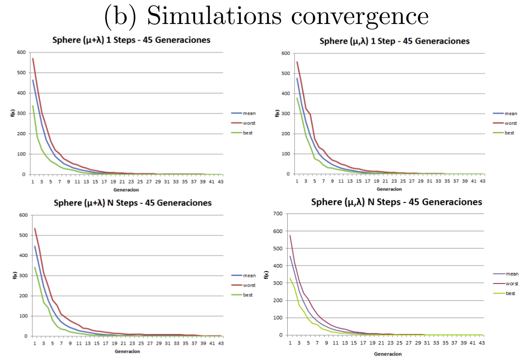


Figure 3: Sphere Function

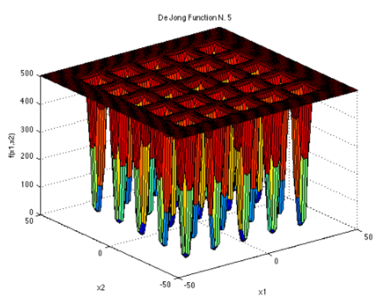
Sphere Function. It is a monomodal functions (with only one minimum which is reached at $f(x_i = 0) = 0$ for each value of i (Figure 3 (a)). The parameters tested are shown in Table 1. Figure 3 (b) shows the best, worst and the average of the simulations. It can be clearly seen that the strategy converges after 30 generations being the (μ, λ) selection scheme the best performer due to the evolution of the sigmas.

DeJong nr. 5 Function. De Jong test function is a multimodal function where the minimum is reached approximately at 1 (Figure 4 (a)). We tested the parameters defined in Table 2. Again, in Figure 4 (b) we show the convergence of the strategy which is quite fast being the $(\mu + \lambda)$ the best selection scheme.

Table 1: Parameters tested for the sphere function.

Parameter	Value
Generations	1200
Num. Variables	25
μ	30
λ	200
Selection	$(\mu+\lambda)$ and (μ,λ)
Mutation	Uncorrelated N steps
Recombination	Discrete
σ	0.2
τ	$\frac{1}{\sqrt{n_{var}}}$
τ'	$\frac{1}{\sqrt{2}} \cdot \sqrt{\tau}$
ϵ	0.001
ρ	2

(a) DeJong 5 Function



$$f(\mathbf{x}) = \left(0.002 + \sum_{i=1}^{25} \frac{1}{i + (x_1 - a_{1i})^6 + (x_2 - a_{2i})^6}\right)^{-1}, \text{ where}$$

$$\mathbf{a} = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \dots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \dots & 32 & 32 & 32 \end{pmatrix}$$

(b) Simulations convergence

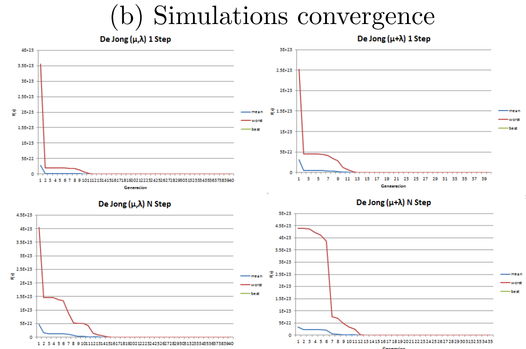


Figure 4: DeJong 5 Function

Table 2: Parameters tested for DeJong function.

Parameter	Value
Generations	1200
Num. Variables	2
μ	30
λ	200
Selection	$(\mu + \lambda)$ and (μ, λ)
Mutation	Uncorrelated N steps
Recombination	Discrete
σ	0.2
τ	$\frac{1}{\sqrt{n_{var}}}$
τ'	$\frac{1}{\sqrt{2}} \cdot \sqrt{\tau}$
ϵ	0.001
ρ	2

Table 3: Parameters tested for Schwefel function.

Parameter	Value
Generations	1200
Num. Variables	20
μ	30
λ	200
Selection	$(\mu + \lambda)$ and (μ, λ)
Mutation	Uncorrelated N steps
Recombination	Discrete
σ	2
τ	$\frac{1}{\sqrt{n_{var}}}$
τ'	$\frac{1}{\sqrt{2}} \cdot \sqrt{\tau}$
ϵ	1
ρ	2

Schwefel Function. Figure 5 (a) shows Schwefel test function which has the minimum at 0. We can see in Figure 5 (b) that the most optimal selection scheme is $(\mu + \lambda)$ with parameters setup defined in Table 3.

(a) Schwefel Function

$$f(x) = 418.982887272433n - \sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|})$$

Dimensions:	n
Domain:	$-500.0 \leq x_i \leq 500.0$
Global Optimum:	$f(x) \approx 0.0$ at $x = (420.9687, 420.9687, \ldots, 420.9687)$
Operator:	SchwefelEvaluator
Charts:	

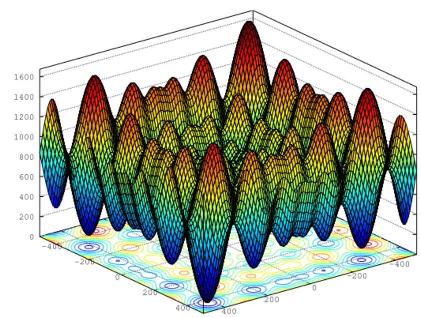


Figure 9: Schwefel function [-500.0, 500.0].

(b) Simulations convergence

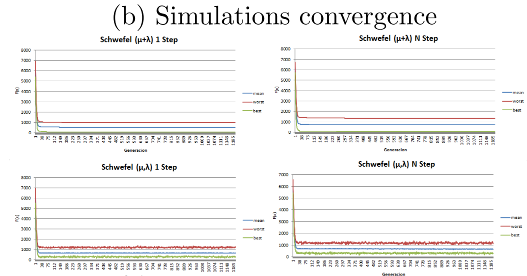


Figure 5: Schwefel Function

References

[1] A.E. Eiben, J.E. Smith. Introduction to Evolutionary Computing.ISBN 978-3-662-05094-1.