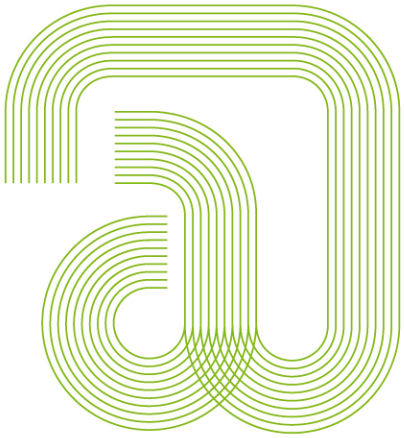
Universidade Vigo

Evolutionary Computation – Lab-Session 2



Escola Superior de Enxeñaría Informática
Edificio Politécnico
Campus universitario
32004 Ourense

http://esei.uvigo.es
mailto:formella@uvigo.es

Referencia: 1.0
Documento: labs-ec
Fecha: February 7, 2024
Páginas: 3

In the lecture we briefly touched mutation operations for genomes coded with genes of real numbers. Let's review.

To simplify notation and reduce the number of indices, let us name the two parent genomes with f (father) and m (mother) and the two child genomes with s (son) and d (daughter), or c (child) whenever we just generate one off-spring.

Hence, for n genes within the genome, we have $f = (f_1, f_2, \ldots, f_n)$ and $m = (m_1, m_2, \ldots, m_n)$, both elements of $\mathbb{R}^n$ as parents. The same for the off-springs: $s, d, c \in \mathbb{R}^n$.

We can compute the lower and upper bounds of the parents, i.e., $\mathfrak{lo}_i = \min(f_i, m_i)$ and $\mathfrak{up}_i = \max(f_i, m_i)$ and the current spread (or box dimensions) of the parents $\Delta_i = \mathfrak{up}_i - \mathfrak{lo}_i$.

Now we can write the different mutation operations abusing a little bit the vector notation (as used in `python` and `numpy`):

arithmetic crossover: The children are just a convex combination of the parents given a random number α_i in $(0,1)$ for each gene

$$s = \alpha \cdot f + (1-\alpha) \cdot m \tag{1}$$
$$d = (1-\alpha) \cdot f + \alpha \cdot m \tag{2}$$

Observe that a child is always contained in the box spanned by the parents. We have excluded the limits for α in order not to reproduce the parents, but what-ever-you-like.

blended crossover: First we select a fixed parameter $\beta \in [-0.5, \infty]$. Then we choose a child uniform-randomly in an interval:

$$c = U[\mathfrak{lo} - \beta\Delta, \mathfrak{up} + \beta\Delta] \tag{3}$$
$$= \mathfrak{lo} - (\beta(1-\alpha) - \alpha \cdot (1+\beta)) \cdot \Delta \tag{4}$$

with $\beta = 0$ we have simply

$$c = \mathfrak{lo} + \alpha \cdot \Delta \tag{5}$$

which is just the same as the arithmetic crossover generating one of the siblings.

Observe, that a $\beta < 0$ shrinks the spread of the off-springs, i.e., the child is always contained within the box of the parents, whereas for a $\beta > 0$ a child might escape the box of the parents.

simulated binary crossover: This operation (called SBC or SBX as well) generates two children according to a random value taken from a specific distribution (which has an additional tuning parameter $\eta \in [1, \infty)$)

$$s = 0.5 \cdot ((1-\beta) \cdot f + (1+\beta) \cdot m) \tag{6}$$
$$d = 0.5 \cdot ((1+\beta) \cdot f + (1-\beta) \cdot m) \tag{7}$$

$$\beta = \begin{cases} \sqrt[\eta]{2\alpha} & \text{if } \alpha \leq 0.5 \\ \sqrt[\eta]{1/(2-2\alpha)} & \text{otherwise} \end{cases} \tag{8}$$

Observe that the sum of the parents is always equal to the sum of the children, i.e., $f + m = s + d$, the same as it happens in the arithmetic crossover above.

The tuning parameter η models the shape of the distribution, a small η generates children close to the parents, a large η generates children away from the parents.

1. Second Week

Objectives: Understand the genome codifications available in the Guofei-package. Understand the mutation and crossover operations implemented in the Guofei-package. Generate convergence plots.

1. Run the jupyter notebook on sorting to repeat and to settle what has been presented in the lecture.
2. Run the examples of the Guofei-packages that minimize the Schaffer-function (with both encodings).
3. Run the examples with different parameter settings and generate the corresponding convergence plots.
4. Which types of mutation operations does the package implement? Give a brief description.
5. Which types of crossover operations does the package implement? Give a brief description.
6. Add the eta-parameter to the corresponding crossover operation. Do you experiment differences in the convergence rate when changing eta for the Schaffer function (maintaining the rest of the parameters)?