Mutation

Operator

Real

Gevol.evolution.evoperator.real

Description

Mutation based on the sigma, alpha values and rotation matrix.

The individual has three chromosomes: values, alpha and sigma. Length of sigma is the same as the values' chromosome. Alpha has length according to the formula:

$$\alpha_d = \frac{v_d * (v_d - 1)}{2}$$

Where d is a length of the values' chromosome.

First step is to calculate sigma and alpha values.

$$\alpha_i = \alpha_i + N(0, \beta^2)$$

$$\sigma_i = \sigma_i * e^{\varepsilon_0 + N(0,\tau^2)}$$

Where

- β is a parameter, it determines how strong the alpha chromosome will be modified, $N(0,\beta^2)$ is a random number by Normal Distribution with mean 0 and standard deviation β.
- ε_0 is calculated for the whole sigma chromosome:
$$\varepsilon_0 = N(0, {\tau_0}^2)$$
- τ_0 (tau_prim) is a parameter, it determines how strong ε_0 will be modified
- τ is a parameter, together with determine how strong sigma values will be modified.

Next step is to generate matrix z.

$$z = \begin{matrix} N(0,{\sigma_1}^2) \\ N(0,{\sigma_2}^2) \\ \vdots \\ N(0,{\sigma_d}^2) \end{matrix}$$

Where

- d is the length of the sigma and values chromosome

Then, rotation matrix is created T. Example for the matrx 5x5, for position (p,q) = (2,4)

$$T(p=2, q=4) = \begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \cos(\alpha_j) & 0 & -\sin(\alpha_j) & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & \sin(\alpha_j) & 0 & \cos(\alpha_j) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{matrix}$$

Calculation for each rotation is according to the formula:

$$T = \prod_{p=1}^{d} \prod_{q=p+1}^{d} T_{pq}(\alpha_j)$$

Where

- j is an index of alpha chromosome, it starts from the first element in the list and it is increased by 1 for every iteration
- function T create the matrix from the example above according to the formula:

$$T_{temp} = \begin{Bmatrix} T[p,p] = \cos(\alpha_j) \\ T[p,q] = -sin(\alpha_j) \\ T[q,p] = \sin(\alpha_j) \\ T[q,q] = \cos(\alpha_j) \end{Bmatrix}$$

Last step is to rotate the matrix:

$$T = T * T_{temp}$$

When the matrix is ready, the mutation parameters are calculated by multiplication of matrix T and z.

$$\varepsilon = T * z$$

In the result, ε has some values:

$$\varepsilon = \begin{matrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_d \end{matrix}$$

Then each value is modified by the corresponding ε.

$$v_i = v_i + \varepsilon_i$$

Parameters

1. Beta β - determines how strong alpha chromosome will be modified, usually it is below 0.1
2. Tau prim τ_0 - determines how strong sigma chromosome will be modified, it has stronger influence than tau τ. Usually it should be below 0.2.
3. Tau τ - determines how strong sigma chromosome will be modified. Usually it should be below 0.2.

Pseudocode

```
updateAlpha chromosome
updateSigma chromosome
d = length of value chromosome
z = generateZMatrix(sigma chromosome)
r = calculateRotationMatrix(alpha chromosome, d)
epsilon = z * r
for each g gene with value from value chromosome
      g[i] = g[i] + epsilon[i]

updateAlpha:
for each a alpha value from alpha chromosome
      a[i] = a[i] + N(0,beta$^2$)
```

```
updateSigma:
epsilon0 = N(0,tau_prim²)
for each s sigma value in sigma chromosome
      s[i] = s[i] * exp(epsilon0 + N(0,tau²)

generateZMatrix(sigma):
for each s sigma value from sigma chromosome
      z[i,0] = N(0,sigma[i]²)

calculateRotationMatrix(alpha, d):
matrix = identity matrix
j = -1
for p = 0 .. d - 1
      for q = p + 1 .. d
            j++
            matrix_temp = identity matrix
            matrix_temp[p,p] = cos(alpha[j])
            matrix_temp[p,q] = -sin(alpha[j])
            matrix_temp[q,p] = sin(alpha[j])
            matrix_temp[q,q] = cos(alpha[j])
            matrix = matrix * matrix_temp
```

Implementation details

In the input population should be only one individual for mutation. For matrix operation is used Math.NET Numerics library. The method is implemented step by step according to the description.

```
RealIndividual newIndividual = new
RealIndividual((RealIndividualChromosome)population[0].Chromosome);

updateAlpha(((RealIndividualChromosome)newIndividual.Chromosome).Alpha);
updateSigma(((RealIndividualChromosome)newIndividual.Chromosome).Sigma);

Matrix<double> z =
generateZMatrix(((RealIndividualChromosome)newIndividual.Chromosome).Sigma);
Matrix<double> r =
calculateRotationMatrix(((RealIndividualChromosome)newIndividual.Chromosome).Alpha,
z.RowCount);
Matrix<double> m_epsilon = r.Multiply(z);

//set new values
for(int i = 0; i < m_epsilon.RowCount; i++)
{
        ((RealIndividualChromosome)newIndividual.Chromosome).Values[i] += m_epsilon[i,
0];
}
RealIndividualChromosome newChromosome =
(RealIndividualChromosome)newIndividual.Chromosome;
newChromosome.Age = 0;
```

References

1. Lectures of PhD Piotr Lipiński “Evolutionary strategies” on University of Wrocław
2. Contemporary Evolution Strategies, Bäck, Th.; Foussette, C.; Krause, P., 2013, ISBN: 978-3-642-40136-7