Evolutionary Programming (EP)

Evolutionary Programming (EP) was proposed by Lawrence Fogel in 60's.

In the classical example of EP, predictors were evolved in the form of finite state machines. A finite state machine (FSM) is a transducer that can be stimulated by a finite alphabet of input symbols and can respond in a finite alphabet of output symbols. It consists of a number of states S and a number of state transitions. The state transitions define the working of the FSM: depending on the current state and the current input symbol, they define an output symbol and the next state to go to.

The idea was evolving finite state machines (FSMs). There are five generally usable mutation operators to generate new FSMs:

- Changing an output symbol
- Changing a state transition
- Adding a state
- Deleting a state
- Changing the initial state

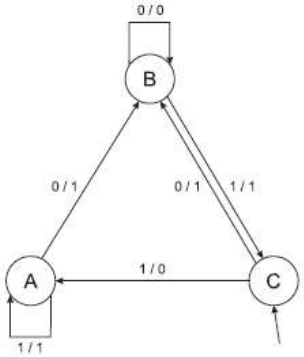

Figura 2 An example of finite state machine consisting of three states: A, B, and C. The input alphabet is I={0,1}, and the output alphabet is O={a,b,c}.

Since 90's, some variants of Evolutionary Programming were proposed, they used real vectors for solving continuous multidimensional optimization problems. In fact, they are the new standard for Evolutionary Programming.

In Evolutionary Programming, **the solutions represent species** instead of individuals.

Representation

The representation is the same of Evolution Strategies using uncorrelated mutation with d step sizes. The solutions are represented as vectors whose inputs are the values of the variables, the i-th solution is represented as the vector $\overrightarrow{x_i} \in \mathbb{R}^d$, where d represents the number of features. In addition, the solution contains the mutation parameters $\overrightarrow{\sigma_i} \in \mathbb{R}^d$.

$$\underbrace{< x_1, x_2, \dots, x_d,}_{\vec{x}} \quad \underbrace{\sigma_1, \sigma_2, \dots, \sigma_d >}_{\overrightarrow{\sigma_i}}$$

Mutation

The species can be seen as points in a $d-$multidimensional space, where the mutation's goal is to move those points so that the position of the mutated individual is close to the position of the individual before mutation. The specie's position $\overrightarrow{x_i}$ is modified adding an random number to each component following the next equations:

$$\sigma_i' = \sigma_i * (1 + N(0, \alpha))$$
$$x_i' = x_i + N(0, \sigma_i')$$

where $\alpha \approx 0.2$. It is recommended avoiding values of σ_i close to 0. If $\sigma' < \epsilon \Rightarrow \sigma' = \epsilon$.

Crossover

It does not apply. It is because the solutions are seen as species.

Parent selection

It does not apply. Each specie (or solution) generates a new one using the mutation operator.

Survivor selection

The survivor selection used in Evolution Programming is known as $(\mu + \mu)$. Each specie generates a new one by mutation. Then, from the set of all the species (the original and the new ones), the μ best are deterministically selected as be part of the new generation.

To sum up, the book Introduction to Evolutionary Algorithms (Eiben) presents the following summary of EP:

- Representation: real-valued vectors
- Parent selection: Deterministic (each parent creates one offspring via mutation)
- Recombination: None
- Mutation: Gaussian perturbation
- Survivor selection: $(\mu + \mu)$.
- Specialty: self-adaptation of mutation step sizes

Evolutionary Programming Algorithm

```
Parameters:
    μ, population size
    G, Maximum number of generations

Return: the elite individual

Begin
  Create the initial population
  Calculate the population fitness
  Get the elite
  While the number of generations is less than G or we haven't found a good solution
      Mutation of all the species
      Calculate the population fitness
      Survivor selection
      Get the elite or include the elite in the population
  End while
End
```