Laboratory of Evolutionary Computing

Evolution Strategies

Norbert Henzel, Ph.D.

Robert Czabanski, D.Sc.

Gliwice, 2019

1   Introduction

Evolutionary algorithms have started to obtain a growing attention during the last years. They have been widely applied in science and engineering to solve many difficult problems. This instruction presents the general structure and working principles of recent evolutionary algorithms. Although different approaches are briefly presented, the main attention is given to Evolution Strategy method.

A very good starting point for further studies is the book of Michalewicz [4].

2   Evolutionary Algorithms

Evolutionary Algorithms (EA) known also frequently under the name Evolutionary Computation (EC) are search techniques based on a powerful principle of evolution, i.e. "survival of the fittest". Evolutionary computation uses computational models of evolutionary processes as key elements in the design and implementation of computer-based problem solving systems.

Although the origins of EC can be found in the late 1950s, the field was relatively unknown for almost three decades. This was mainly due to the lack of available powerful computer systems.

There are a variety of evolutionary computational model techniques that have been proposed and studied, which can be referred to as evolutionary algorithms. They share a common conceptual base of simulating the evolution of individual structures via processes of selection and reproduction. These processes depend on the perceived performance (fitness) of the individual structures as defined by an environment.

In general, many real world problems can be perceived as an optimization problem. They require finding values of free parameters of the system under consideration, $\mathbf{x} \in D$, such that a certain quality function (called objective function), $f : D \rightarrow \mathbb{R}$ is maximized (or, equivalently, minimized)

$$f\left(\mathbf{x}\right) \rightarrow \max. \tag{1}$$

The solution to the global optimization problem requires finding such a vector $\mathbf{x}^*$ that fulfills the following criterion:

$$\underset{\mathbf{x} \in D}{\forall} f\left(\mathbf{x}\right) \leq f\left(\mathbf{x}^*\right) = f^*. \tag{2}$$

Different characteristics of the investigated problem, such as multimodality (existence of several local maxima), constraints (set of feasible solutions $F \subseteq D$ is only a subset of the considered domain), large dimensionality, strong nonlinearities, nondifferentiability, noisy and time-varying objective functions, frequently lead to difficult, if not unsolvable, optimization task. Also, the identification of the improvement of the best known solutions so far is of great importance. In many cases EA provide an efficient method to achieve this goal.

The most significant advantages of using evolutionary search consist in the gain of flexibility and adaptability to the investigated problem, robust performance and global search capability.

3 The Structure of an Evolutionary Algorithm

Evolutionary algorithms are based on the process of natural evolution, the driving process for the adaptation of organic structures to their environment. Evolution can be understood as a result of interaction between the creation of new genetic information and its evaluation and selection. A single individual in a population is affected by others individuals (e.g., predators, food competition, mating, etc.) as well as by the environment (e.g., climate and food supply). The better performs an individual under these conditions, the greater are the chances for the individual to live longer and produce offspring. This leads to constant reinsertion of the genetic information of the above-average individuals into the population. On the other hand, the nondeterministic nature of reproduction favorises the creation of novel genetic information and creation of different, possibly better adapted, offspring.

Evolutionary algorithms maintain a population of structures that evolve according to rules of selection and other operators, such as recombination and mutation. Each individual in the population receives a measure of its fitness to the environment. Selection focuses attention on high fitted individuals, thus exploiting the available fitness information. Recombination and mutation perturb those individuals, providing general heuristics for exploration. Although simplified from a biologist viewpoint, these algorithms are sufficiently complex to provide robust and powerful adaptive search mechanisms.

Figure 1 outlines a typical evolutionary algorithm (EA). A population of individual structures is initialized and then evolved from generation to generation by repeated applications of evaluation, selection, recombination, and mutation. The population size μ is generally constant in the algorithm, although there is no a priori reason (other than convenience) to make this assumption.

```
procedure EA{
    t = 0
    initialize population P(t)
    evaluate P(t)
    while (not termination condition) do{
        t = t + 1
        select P(t) from P(t-1)
        alter P(t)
        evaluate P(t)
    }
}
```

Figure 1: A typical evolutionary algorithm.

An evolutionary algorithm operates on a population of individuals $P(t) = \left\{ x_1^{(t)}, x_2^{(t)}, \cdots, x_\mu^{(t)} \right\}$, where μ is the size of the population and t is the iteration number. In this population each individual represents a possible solution to the investigated

problem. The individuals are implemented as some data structure, particular for the problem. An evolutionary algorithm typically initializes the population randomly, although domain specific knowledge can also be used to improve the search. Evaluation step measures the fitness of each individual $x_i^{(t)}$ according to its worth in an environment. Evaluation may be as simple as computing a fitness function or as complex as running an elaborate simulation. Then, a new population (in the iteration $t+1$) is formed by selecting a set of individuals that will participate in further steps. The selection operation is based on the fitness values of the individuals. The most popular selection schema are: proportional selection, rank-based selection and tournament selection:

- the **proportional selection** is the most popular selection approach, where the selection is implemented as a probabilistic operator. The relative fitness is used to determine the selection probability

$$p\left(x_i^{(t)}\right) = \frac{f\left(x_i^{(t)}\right)}{\sum\limits_{i=1}^{\mu} f\left(x_i^{(t)}\right)}, \tag{3}$$

  of an individual,

- the **rank-based selection** utilizes the indexes of individuals ordered according to fitness values to determine the corresponding selection probabilities,

- the **tournament selection** method performs by taking a random uniform sample from the population of the size q ($q > 1$). Next, from these q individuals the best c ($1 \le c \le q$) are selected for the next population. This selection process is repeated until a new population of size μ is formed.

Selection serves to focus search into areas of high fitness. Of course, if selection was the only genetic operator, the population would never have any individuals other than those introduced in the initial population. Other genetic operators perturb (alter) these individuals, providing exploration in nearby areas. Although a number of operators are possible, we will concentrate on the two predominant operators, namely, mutation and recombination.

The importance of mutation in EAs varies widely. Some approaches do not use mutation at all, some use mutation as a simple background operator to ensure that a particular bit value is not lost forever. In other EC techniques mutation is the most important genetic operator. In this case, instead of a global mutation rate, mutation probability distributions are usually used for every variable of every individual. Thus, each variable can be mutated according to a different probability distribution. More importantly, ESs and EPs encode the probability distributions as an extra information within each individual, and allow this information to evolve as well.

In its simplest form, mutation is implemented as a random change (or multiple changes) of the parent. If the parent can be represented in the following form:

$$x_i = \langle x_{i1}, x_{i2}, \cdots, x_{ik}, \cdots, x_{in} \rangle, 1 \le k \le n, \tag{4}$$

then every element x_{ik} has the same probability to be modified. In this case, the effect of mutation is represented as follows:

$$x_i^* = \langle x_{i1}, x_{i2}, \cdots, x_{ik}^*, \cdots, x_{in} \rangle, 1 \leq k \leq n, \qquad (5)$$

where x_{ik}^* is random value chosen from the domain of variable x_{ik}.

For example, the mutation operator applied to a sequence of binary values may be represented by

$$x_i = \langle 011\mathbf{0}100010010001 \rangle, \Longrightarrow x_i^* = \langle 011\mathbf{1}100010010001 \rangle. \qquad (6)$$

Recombination is the other predominant genetic operator. Recombination merges variables from two parents to produce offsprings that have characteristics from both parents. Like mutation, the relative importance and types of recombination in various EAs varies widely. We will present only the most popular one-point and multi-point crossover recombination.

One-point recombination inserts a cut-point within the two parents. Then the information before the cut-point is swapped between the two parents. For two parents represented by

$$x_i = \langle x_{i1}, \cdots, x_{ik}, x_{ik+1} \cdots, x_{in} \rangle, x_j = \langle x_{j1}, \cdots, x_{jk}, x_{jk+1} \cdots, x_{jn} \rangle, \qquad (7)$$

we have

$$x_i^* = \langle x_{j1}, \cdots, x_{jk}, x_{ik+1} \cdots, x_{in} \rangle, x_j^* = \langle x_{i1}, \cdots, x_{ik}, x_{jk+1} \cdots, x_{jn} \rangle. \qquad (8)$$

Multi-point recombination is a generalization of this idea, by introducing a higher number of cut-points. Information is then swapped between pairs of cut-points.

Of course, any genetic operator such as mutation and recombination must be defined with a particular individual representation in mind. The most popular representations used are bit strings, vectors of real values, ordered lists, neural networks, finite state automata and Lisp expressions. For each of these representations, special mutation and recombination operators are introduced.

4 Varieties of Evolutionary Algorithms

There exist a great variety of evolutionary computation techniques. Although similar at the main level, each of these techniques implements an evolutionary algorithm in a different way. The differences touch upon almost all aspects of evolutionary algorithms, but the most important ones are the choices of representation for the individual structures, types of selection mechanism used, forms of genetic operators, and measures of performance. In the following section we will present the most important techniques represented by the current family of evolutionary algorithms with particular emphasis on to the Evolution Strategy method.

4.1   Genetic Algorithms

Genetic algorithms (GAs), developed by Holland [2], perform a multi-dimensional search by maintaining a population of potential solutions. The GAs have traditionally used a domain independent representation, namely, bit-strings. However, many applications of GAs have focused on other representations, such as graphs (neural networks), ordered lists, and real-valued vectors.

After initialization parents are selected according to a probabilistic function based on relative fitness. In other words, those individuals with higher relative fitness are more likely to be selected as parents. N offspring are created via recombination from the N parents. The N descendant are mutated and survive, replacing the N parents in the population. In a GA mutation flips bits with some small probability, and is often considered to be a back-ground operator. Recombination, on the other hand, is emphasized as the primary search operator. GAs are often used as optimizers, although some researchers emphasize their general adaptive capabilities.

4.2   Evolutionary Programming

The main objective of Evolutionary programming (EP) [1] is to develop an artificial intelligence capability to predict changes in an environment. The environment is described as a sequence of symbols selected from a finite alphabet and the prediction ability takes a form of a new symbol produced on the output of the evolved structure.

As a chromosomal representation finite state machines (FSM), real-valued vectors or ordered lists are usually used.

After initialization, all N individuals are selected to be parents, and then are mutated, producing N offspring. These offspring are evaluated and N survivors are chosen from the $2N$ individuals, using a probabilistic function based on the fitness. In other words, individuals with a greater fitness have a higher chance of survival. The form of mutation is based on the representation used, and is often adaptive.

4.3   Genetic Programming

One of the most fascinating recent developments is the use of EAs to evolve complex structures such as computer programs. This approach, named Genetic Programming (GP) was developed recently by Koza [3]. The main idea of this approach is, instead of developing a computer program that solves the investigated problem, to search a space of all possible computer programs for the best one (which solves our problem in a most adequate manner). The structure which is changed during the evolution process is a hierarchically structured computer program. The searched space is a hyperspace of all valid computer programs. The computer programs are viewed as a space of rooted trees composed of elements appropriate for the particular problem.

The initial population is composed of randomly generated trees. The evaluation function measures the fitness (performance) of the programs in the population. Each tree has a selection probability proportional to its fitness. The primary genetic operator, crossover, creates a new offspring by exchanging subtrees between the two parents.

Mutation replaces a subtree by a new generated one.

4.4   Evolution Strategies

Evolution strategies, introduced by Rechenberg [5] and Schwefel [6] emphasize the behavioral link between parents and offsprings. The original evolution strategies were based on a population containing one individual only. This single parent produces a single offspring by means of a mutation operator. Evolution strategies (ES) were developed as a method to solve parameter optimization problems. The structure of chromosome representing an individual is a pair of float vectors $\mathbf{v} = (\mathbf{x}, \sigma)$. The first vector, $\mathbf{x} \in \mathbb{R}^n$, represents here a point in the n-dimensional search space and the second vector $\sigma \in \mathbb{R}^n_+$ is the vector of standard deviations. The main idea behind ES was to introduce a mechanism for a self-adaptation of the control parameters. This is achieved by applying the genetic operators to the object variable $\mathbf{x}$, and to the strategy parameter σ. The mutation operator works by adding a normally distributed random vector $\mathbf{r} \in \mathbb{R}^n$ with a mean of zero and covariance matrix $\sum = \mathrm{diag}\left(\sigma_1^2, \sigma_2^2, \cdots, \sigma_n^2\right) = \mathrm{diag}\left(\sigma^{\mathbf{2}}\right)$

$$\mathbf{r} = \mathbf{N}\left(0, \sigma\right), \tag{9}$$

to the object variable

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \mathbf{r}, \tag{10}$$

and the strategy parameter is updated as follows

$$\sigma^{(t+1)} = \sigma^{(t)} \cdot \exp\left(r_{\sigma 1}\right) \cdot \exp\left(\mathbf{r}_{\sigma 2}\right), \tag{11}$$

where superscript (t) denotes iteration t, $r_{\sigma 1}$ is a single realization of zero-mean gaussian variable with variance τ_1^2

$$r_{\sigma 1} = \mathrm{N}\left(0, \tau_1\right) = \tau_1 \cdot \mathrm{N}\left(0, 1\right), \tag{12}$$

and $\mathbf{r}_{\sigma 2}$ s a single realization of zero-mean gaussian variable with covariance matrix $\sum_\sigma = \tau_2^2 \mathcal{I}$

$$\mathbf{r}_{\sigma 2} = \mathbf{N}\left(\mathbf{0}, \tau_2^2 \mathcal{I}\right) = \tau_2 \cdot \mathbf{N}\left(\mathbf{0}, \mathbf{1}\right), \tag{13}$$

where $\mathcal{I}$ denotes identity matrix. Parameters τ_1 and τ_2 usually are chosen as follows:

$$\tau_1 = \frac{1}{\sqrt{2n}}, \tag{14}$$

and

$$\tau_2 = \frac{1}{\sqrt{2\sqrt{n}}}. \tag{15}$$

The offspring $\mathbf{x}^{(t+1)}$ is accepted as a new member of the population (it replaces the parent) if and only if (iff) it has better fitness with respect to the old parent $\mathbf{x}^{(t)}$. If $f(\mathbf{x})$ denotes a cost (fitness) function, then $\mathbf{x}^{(t+1)}$ replaces $\mathbf{x}^{(t)}$ iff $f\left(\mathbf{x}^{(t+1)}\right) < f\left(\mathbf{x}^{(t)}\right)$ (minimization problem).

The evolution strategies evolved into two mature approaches denoted as $(\mu + \lambda)$-ESs and (μ, λ)-ESs. The $(\mu + \lambda)$ strategy selects the μ best solutions from the union of parents and offsprings. In contrast, in the (μ, λ) strategy the best μ offsprings are selected from λ $(\lambda > \mu)$ descendants to replace the parents (the life of each individual is limited to only one generation).

Sometimes also a recombination process (crossover operation) is included in evolution strategies. Selecting two individuals $\mathbf{v}_i$ and $\mathbf{v}_j$ form the population

$$
\begin{aligned}
\mathbf{v}_i &= (\mathbf{x}_i, \sigma_i) = (x_{i1}, \cdots, x_{in}, \sigma_{i1}, \cdots, \sigma_{in}), \\
\mathbf{v}_j &= (\mathbf{x}_j, \sigma_j) = (x_{j1}, \cdots, x_{jn}, \sigma_{j1}, \cdots, \sigma_{jn}),
\end{aligned}
\tag{16}
$$

there are two types of the crossover:

- discrete crossover, where the new offspring is

$$
\mathbf{v} = (\mathbf{x}, \sigma) = (x_{q_1 1}, \cdots, x_{q_n n}, \sigma_{q_1 1}, \cdots, \sigma_{q_n n}),
\tag{17}
$$

  where $q_k = i$ or $q_k = j$, $k = 1, 2, \cdots, n$. In this case each component comes from the first or second previously selected parent,

- intermediate crossover, where the created offspring is given as

$$
\mathbf{v} = (\mathbf{x}, \sigma) = \left(\frac{x_{i1} + x_{j1}}{2}, \cdots, \frac{x_{in} + x_{jn}}{2}, \frac{\sigma_{i1} + \sigma_{j1}}{2}, \cdots, \frac{\sigma_{in} + \sigma_{jn}}{2} \right).
\tag{18}
$$

Denoting the population in iteration t as $P_\mu^{(t)} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_\mu\}$ the evolution strategies method can be summarized in the following steps:

1. Set iteration index, $t = 0$. Initialize $P_\mu^{(0)}$.

2. Evaluate $P_\mu^{(0)}$.

3. Perform reproduction of $P_\mu^{(t)}$ to $R_\lambda^{(t)}$.

4. Perform mutation of $R_\lambda^{(t)}$ to $O_\lambda^{(t)}$.

5. Evaluate $O_\lambda^{(t)}$.

6. Select $\left(O_\lambda^{(t)} \cup Q \right)$ to $P_\mu^{(t+1)}$.

7. Update iteration index, $t = t + 1$.

8. If termination condition is not satisfied go to the Step 3.

In the (μ, λ) strategy $Q = \emptyset$ and in the $(\mu + \lambda)$ strategy $Q = P_\mu^{(t)}$. In the third step, the individuals from μ parents are reproduced to λ individuals in the set $R_\lambda^{(t)}$ by independently random drawn individuals from $P_\mu^{(t)}$. As the termination condition usually pre-set number of iteration or achieving of pre-set value of the cost function is used.

5 Tasks

1. Write a computer program to solve the optimization problem provided by a tutor using the Evolution Strategy. Implement both, (μ, λ) and $(\mu + \lambda)$ approaches. As the population varying operator only mutation should be used.

2. Implement ES crossover operators: discrete and intermediate crossover. Evaluate the influence of the ES parameters (number of individuals, number of offspring, selection and crossover method, ...) on the performance and the time of computations.

References

[1] L.J. Fogel, A.J Owens, M.J. Walsh, *Artificial Intelligence Through Simulated Evolution*, Wiley Publishing, New York, 1966.

[2] J.H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, Michigan, 1975.

[3] J.R. Koza, *Genetic Programming*, MIT Press, Cambridge, MA, 1992.

[4] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs*, Springer-Verlag, Berlin, 1996.

[5] I. Rechenberg, *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*, Frommann-Holzboog, Stuttgart, 1973.

[6] H.-P. Schwefel, *Evolutionsstrategie and numerische Optimierung Disertation*, Technische Universität Berlin, 1975.