Programación evolutiva

María del Rocío Ochoa Montiel

Contacto: ma.rocio.ochoa@gmail.com

An Overview of Evolutionary Computation

by
William M. Spears
Kenneth A. De Jong
Thomas Baeck
David B. Fogel
Hugo de Garis

Evolutionary computation uses computational models of evolutionary processes as key elements in the design and implementation of computer-based problem solving systems. There are a variety of evolutionary computational models that have been proposed and studied which we will refer 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.

More precisely, 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 in the environment. Selection focuses attention on high fitness individuals, thus exploiting the available fitness information. Recombination and mutation perturb those individuals, providing general heuristics for exploration. Although simplistic from a biologist's viewpoint, these algorithms are sufficiently complex to provide robust and powerful adaptive search mechanisms.

Evolutionary programming (EP), developed by Fogel et al. (1966) traditionally has used representations that are tailored to the problem domain. For example, in real-valued optimization problems, the individuals within the population are real-valued vectors. Similarly, ordered lists are used for traveling salesman problems, and graphs for applications with finite state machines. EP is often used as an optimizer, although it arose from the desire to generate machine intelligence.

```
procedure EA; {
t = 0;
initialize population P(t);
evaluate P(t);
until (done) {
        t = t + 1;
        parent_selection P(t);
        recombine P(t);
        mutate P(t);
        evaluate P(t);
        survive P(t);
}       }
```

A typical evolutionary algorithm

```
procedure EP; {
t = 0;
initialize population P(t);

 evaluate P(t);
 until (done) {
        t = t + 1;
        parent_selection P(t);
        mutate P(t);
        evaluate P(t);
        survive P(t);
 }      }
```

The evolutionary programming algorithm

An efficient constraint handling method for genetic algorithms

Kalyanmoy Deb

Kanpur Genetic Algorithms Laboratory (KanGAL), Department of Mechanical Engineering, Indian Institute of Technology Kanpur, Kanpur 208 016, India

Many search and optimization problems in science and engineering involve a number of constraints which the optimal solution must satisfy. A constrained optimization problem is usually written as a nonlinear programming (NLP) problem of the following type:

$$\begin{aligned} \text{Minimize} \quad & f(\vec{x}) \\ \text{Subject to} \quad & g_j(\vec{x}) \geqslant 0, \quad j = 1, \ldots, J, \\ & h_k(\vec{x}) = 0, \quad k = 1, \ldots, K, \\ & x_i^l \leqslant x_i \leqslant x_i^u, \quad i = 1, \ldots, n. \end{aligned} \tag{1}$$

In the above NLP problem, there are n variables (that is, $\vec{x}$ is a vector of size n), J greater-than-equal-to type inequality constraints, and K equality constraints. The function $f(\vec{x})$ is the objective function, $g_j(\vec{x})$ is the jth inequality constraints, and $h_k(\vec{x})$ is the kth equality constraints. The ith variable varies in the range $[x_i^l, x_i^u]$.

2. Constraint handling in GAs

In most applications of GAs to constrained optimization problems, the penalty function method has been used. In the penalty function method for handling inequality constraints in minimization problems, the fitness function $F(\vec{x})$ is defined as the sum of the objective function $f(\vec{x})$ and a penalty term which depends on the constraint violation $\langle g_j(\vec{x})\rangle$:

$$F(\vec{x}) = f(\vec{x}) + \sum_{j=1}^{J} R_j \langle g_j(\vec{x})\rangle^2, \tag{2}$$

where $\langle\ \rangle$ denotes the absolute value of the operand, if the operand is negative and returns a value zero, otherwise. The parameter R_j is the penalty parameter of the jth inequality constraint. The purpose of a penalty parameter R_j is to make the constraint violation $g_j(\vec{x})$ of the same order of magnitude as the objective function value $f(\vec{x})$. Equality constraints are usually handled by converting them into inequality constraints as follows: [1]

$$g_{k+J}(\vec{x}) \equiv \delta - |h_k(\vec{x})| \geqslant 0,$$

where δ is a small positive value. This increases the total number of inequality constraints to $m = J + K$ and the term J in Eq. (2) can then be replaced by m to include all inequality and equality constraints. Thus, there are total of m penalty parameters R_j which must be set right in a penalty function approach.

Excercises

$$
\begin{aligned}
\text{Minimize} \quad & f_1(\vec{x}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \\
\text{Subject to} \quad & g_1(\vec{x}) \equiv 4.84 - (x_1 - 0.05)^2 - (x_2 - 2.5)^2 \geqslant 0, \\
& g_2(\vec{x}) \equiv x_1^2 + (x_2 - 2.5)^2 - 4.84 \geqslant 0, \\
& 0 \leqslant x_1 \leqslant 6, \ 0 \leqslant x_2 \leqslant 6.
\end{aligned} \tag{7}
$$

The unconstrained objective function $f_1(x_1, x_2)$ has a minimum solution at (3,2) with a function value equal to zero. However, due to the presence of constraints, this solution is no more feasible and the constrained optimum solution is $x^* = (2.246826, 2.381865)$ with a function value equal to $f_1^* = 13.59085$.

$$
\begin{aligned}
\text{Minimize} \quad & f_4(\vec{x}) = x_1 + x_2 + x_3 \\
\text{Subject to} \quad & g_1(\vec{x}) \equiv 1 - 0.0025(x_4 + x_6) \geqslant 0, \\
& g_2(\vec{x}) \equiv 1 - 0.0025(x_5 + x_7 - x_4) \geqslant 0, \\
& g_3(\vec{x}) \equiv 1 - 0.01(x_8 - x_5) \geqslant 0, \\
& g_4(\vec{x}) \equiv x_1 x_6 - 833.33252 x_4 - 100 x_1 + 83333.333 \geqslant 0, \\
& g_5(\vec{x}) \equiv x_2 x_7 - 1250 x_5 - x_2 x_4 + 1250 x_4 \geqslant 0, \\
& g_6(\vec{x}) \equiv x_3 x_8 - x_3 x_5 + 2500 x_5 - 1\,250\,000 \geqslant 0, \\
& 100 \leqslant x_1 \leqslant 10\,000, \\
& 1000 \leqslant (x_2, x_3) \leqslant 10\,000, \\
& 10 \leqslant x_i \leqslant 1000, \quad i = 4, \ldots, 8.
\end{aligned}
$$

The optimum solution is

$$\vec{x}^* = (579.3167,\ 1359.943,\ 5110.071,\ 182.0174,\ 295.5985,\ 217.9799, 286.4162,\ 395.5979),$$

$$f_4^* = 7049.330923.$$

All six constraints are active at this solution.

Referencias

1. Coello Coello C.A. (2020, May). *Introducción a la Computación Evolutiva (Notas de Curso)*. CINVESTAV-IPN Departamento de Computación. México, D.F.
2. Deb K. (2001). *Multiobjective Optimization using Evolutionary Algorithms* (1st. ed.) John Wiley & Sons, Ltd. U.K.
3. Haupt L.Randy and Haupt Sue Ellen. (2003). *Practical Genetic Algorithms*. (2nd. Ed.) John Wiley & Sons, Inc. USA.
4. Fogel (1998); Evolutionary Computation: The Fossil Record; IEEE Press.

1. Abarca-García, et.al. (2015). Complejidad y sistemas complejos: Un acercamiento multidimensional. CopIt-arXives.
2. Abraham Ajith, Jain Lakhmi and Goldberg Robert. (2005). *Evolutionary Multiobjective Optimization Theoretical Advances and Applications*. Springer-Verlag London.
3. Storn, R.M., Price, K. V. (1997, 01) Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*. 11, 341-359. doi: 10.1023/A:1008202821328.
4. Coello Coello Carlos A., Lamont Gary A. and Van Veldhuizen A. Van. (2007). *Evolutionary algorithms for solving multi-objective problems*. (2ª. ed). Springer US.