The Continuous Genetic Algorithm

Universidad de los Andes-CODENSA

1. Components of a Continuous Genetic Algorithm

The flowchart in figure1 provides a big picture overview of a continuous GA..

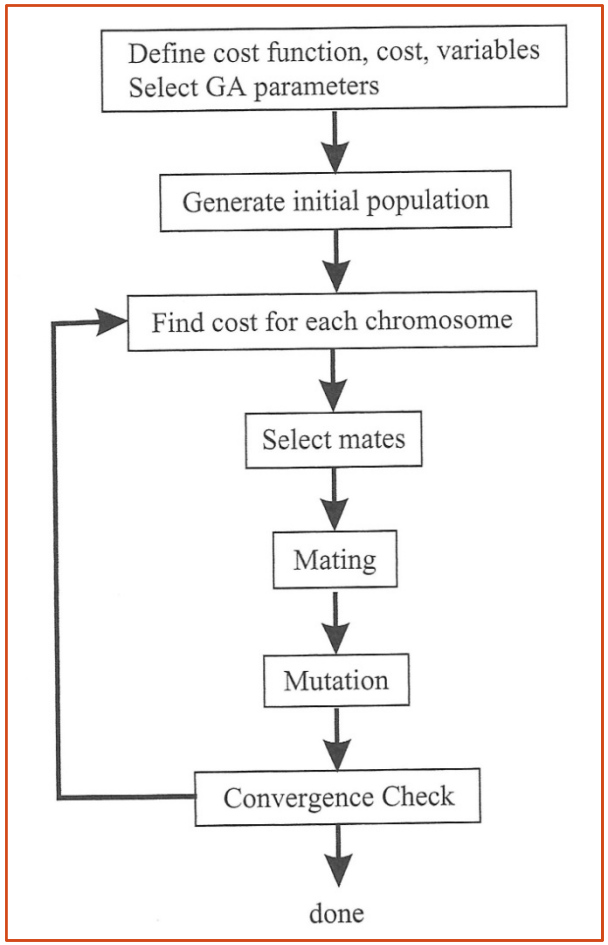

Figure 1. Flowchart of a continuous GA.

1.1. The Example Variables and Cost Function

The goal is to solve some optimization problem where we search for an optimal (minimum) solution in terms of the variables of the problem. Therefore we begin the process of fitting it to a GA by defining a chromosome as an array of variable values to be optimized. If the chromosome as N_{var} variables given by p_1, p_2, …,p_{Nvar} then the chromosome is written as an array with $1 \times N_{var}$ elements so that:

$$\text{chromosome} = \left[p_1, p_2, p_3, \ldots p_{N\,\text{var}} \right]$$

In this case, the variable values are represented as floating point numbers. Each chromosome has a cost found by evaluating the cost function f at the variables p_1, p_2,…,p_{Nvar}.

$$\text{cost} = f(chromosome) = f(p_1, p_2, \ldots, P_{N\,\text{var}})$$

Last two equations along with applicable constraints constitute the problem to be solved.

Example

Consider the cost function

$$\boxed{\text{cost} = f(x, y) = x\sin(4x) + 1.1y\sin(2y)}$$

Subject to

$$\boxed{\begin{aligned} 0 \le x \le 10 \\ 0 \le y \le 10 \end{aligned}}$$

Since f is a function of x and y only, the clear choice for the variable is:

$$\boxed{\text{chromosome} = [x, y]}$$

with $N_{var}=2$.

1.2. Variable Encoding, Precision, and Bounds

We no longer need to consider how many bits are necessary to accurately represent a value. Instead, x and y have continuous values that fall between the bounds.

When we refer to the continuous GA, we mean the computer uses its internal precision and round off to define the precision of the value. Now the algorithm is limited in precision to the round off error of the computer.

Since the GA is a search technique, it must be limited to exploring a reasonable region of variable space. Sometimes this is done by imposing a constraint on the problem. If one does not know the initial search region, there must be enough diversity in the initial population to explore a reasonably sized variable space before focusing on the most promising regions.

1.3. Initial Population

To begin the GA, we define an initial population of N_{pop} chromosomes. A matrix represents the population with each row in the matrix being a $1 \times N_{var}$ array (chromosome) of continuous values. Given an initial population of N_{pop} chromosomes, the full matrix of $N_{pop} \times N_{var}$ random values is generated by:

$$\text{pop} = \text{rand}(N_{pop}, N_{\text{var}})$$

All values are normalized to have values between 0 and 1, the range of a uniform random number generator. The values of a variable are "unnormalized" in the cost function. If the range of values is between p_{lo} and p_{hi}, then the unnormalized values are given by:

$$p = (p_{hi} - p_{lo})\, p_{norm} + p_{lo}$$

Where

p_{lo}: lowest number in the variable range

p_{hi}: highest number in the variable range

p_{norm}: normalized value of variable

This society of chromosomes is not a democracy: the individual chromosomes are not all created equal. Each one's worth is assessed by the cost function. So at this point, the chromosomes are passed to the cost function for evaluating.

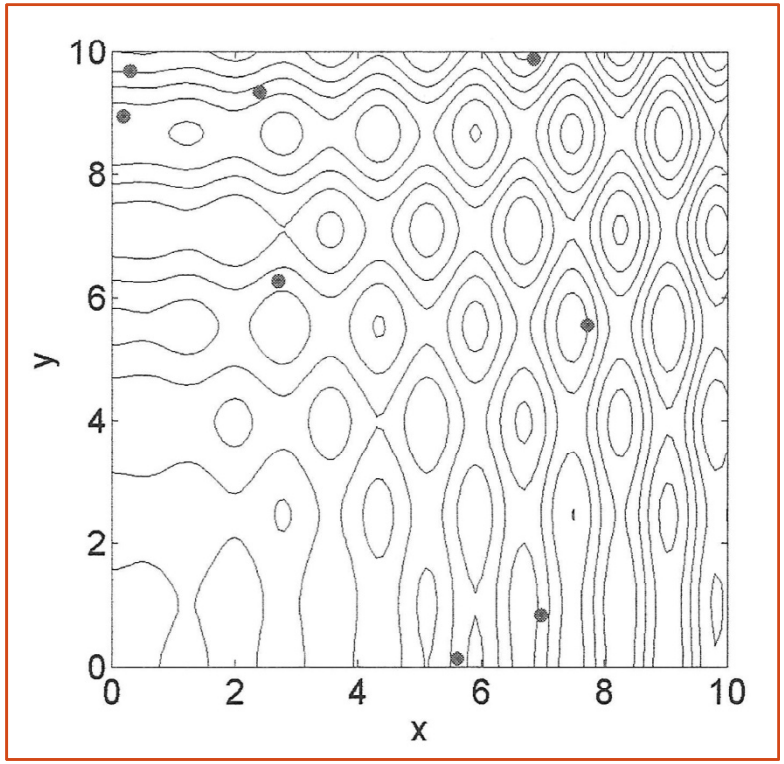

Figure 2. Contour plot of the cost function with the initial population (N_{pop}=*8*) indicated by large dots.

x	y	Cost
6.9745	0.8342	3.4766
0.30359	9.6828	5.5408
2.402	9.3359	–2.2528
0.18758	8.9371	–8.0108
2.6974	6.2647	–2.8957
5.613	0.1289	–2.4601
7.7246	5.5655	–9.8884
6.8537	9.8784	13.752

Table 1. Example Initial population of 8 random chromosomes and their corresponding cost.

1.4. Natural Selection

Now is the time to decide which chromosomes in the initial population are fit enough to survive and possibly reproduce offspring in the next generation. As done for the binary version of the algorithm, the N_{pop} costs and associated chromosomes are ranked from lowest cost to highest cost. The rest die off. This process of natural selection must occur at each iteration of the algorithm to allow the population of chromosomes to evolve over the generations to the most fit members as defined by the cost function. Not all the survivors are deemed fit enough to mate. Of the N_{pop} chromosomes in a given generation, only the top N_{keep} are kept for mating and the rest are discarded to make room for the new offspring.

Number	x	y	Cost
1	7.7246	5.5655	–9.8884
2	0.1876	8.9371	–8.0108
3	2.6974	6.2647	–2.8957
4	5.6130	0.12885	–2.4601

Table 2. Surviving chromosomes after 50% selection rate.

1.5. Pairing

The $N_{keep}=4$ most fit chromosomes form the mating pool. Two mothers and fathers pair in some random fashion. Each pair produces two offspring that contain traits from each parent. In addition the parents survive to be part of the next generation. The more similar the two parents, the more likely are the offspring to carry the traits of the parents.

2	ma(1)	0.18758	8.9371
3	pa(1)	2.6974	6.2647
5	$offspring_1$	0.2558	6.2647
6	$offspring_2$	2.6292	8.9371
3	ma(2)	2.6974	6.2647
1	pa(2)	7.7246	5.5655
7	$offspring_3$	6.6676	5.5655
8	$offspring_4$	3.7544	6.2647

Table 3. Pairing and mating process of single-point crossover chromosome family binary string cost.

1.6. Mating

As for the binary algorithm, two parents are chosen, and the offspring are some combination of these parents.

The simplest methods choose one or more points in the chromosome to mark as the crossover points. Then the variables between these points are merely swapped between the two parents. For example purposes, consider the two parents to be:

$$parent_1 = \left[p_{m1}, p_{m2}, p_{m3}, p_{m4}, p_{m5}, p_{m6}, \ldots, p_{mN\,\mathrm{var}} \right]$$
$$parent_2 = \left[p_{d1}, p_{d2}, p_{d3}, p_{d4}, p_{d5}, p_{d6}, \ldots, p_{dN\,\mathrm{var}} \right]$$

Crossover points are randomly selected, and then the variables in between are exchanged:

$$offspring_1 = \left[p_{m1}, p_{m2}, \uparrow p_{d3}, p_{d4}, \uparrow p_{m5}, p_{m6}, \ldots, p_{mN\,\mathrm{var}} \right]$$
$$offspring_2 = \left[p_{d1}, p_{d2}, \uparrow p_{m3}, p_{m4}, \uparrow p_{d5}, p_{d6}, \ldots, p_{dN\,\mathrm{var}} \right]$$

The extreme case is selecting N_{var} points and randomly choosing which of the two parents will contribute its variable at each position. Thus one goes down the line of the chromosomes and, at each variable, randomly chooses whether or not to swap the information between the two parents.

This method is called uniform crossover:

$$offspring_1 = [p_{m1}, p_{d2}, p_{d3}, p_{d4}, p_{d5}, p_{m6}, \ldots, p_{dN\,\mathrm{var}}]$$
$$offspring_2 = [p_{d1}, p_{m2}, p_{m3}, p_{m4}, p_{m5}, p_{d6}, \ldots, p_{mN\,\mathrm{var}}]$$

The problem with these point crossover methods is that no new information is introduced: Each continuous value that was randomly initiated in the initial population is propagated to the next generation, only in different combinations. Although this strategy work fine for binary representations, there is now a continuum of values, and in this continuum we are merely interchanging two data points. These approaches totally rely on mutation to introduce new genetic material.

The blending methods remedy this problem by finding ways to combine variable values from the two parents into new variable values in the offspring. A single offspring variable value, p_{new}, comes from a combination of the two corresponding offspring variable values:

$$p_{new} = \beta p_{mn} + (1-\beta) p_{dn}$$

Where:

β: random number on the interval [0,1]

p_{mn}: n*th* variable in the mother chromosome

p_{dn}: n*th* variable in the father chromosome

The same variable of the second offspring is merely the complement of the first. If $\beta=1$, the p_{mn} propagates in its entirety and p_{dn} dies. In contrast, if $\beta=0$, then p_{dn} propagates in its entirety and p_{mn} dies. When $\beta=0.5$, the result is an average of the variables of the two parents. Choosing which variables to blend is the next issue. Sometimes, this linear combination process is done for all variables to the right or to the left of some crossover point. Any number of points can be chosen to blend, up to N_{var} values where all variables are linear combination of those of the two parents. The variables can be blended by using the same β for each variable or by choosing different β's for each variable.

However, they do not allow introduction of values beyond the extremes already represented in the population. Top do this requires an extrapolating method. The simplest of these methods is linear crossover.

In this case three offspring are generated from the two parents by:

$$pnew1 = 0.5\,pmn + 0.5\,pdn$$
$$pnew2 = 1.5\,pmn - 0.5\,pdn$$
$$pnew3 = -0.5\,pmn + 1.5\,pdn$$

Any variable outside the bounds is discarded in favor of the other two. Then the best two offspring are chosen to propagate. Of course, the factor 0.5 is not the only one that can be used in such a method. Heuristic crossover is a variation where some random number, β, is chosen on the interval *[0,1]* and the variables of the offspring are defined by:

$$p_{new} = \beta(p_{mn} - p_{dn}) + p_{mn}$$

Variations on this theme include choosing any number of variables to modify and generating different β for each variable. This method also allows generation of offspring outside of the values of the two parent variables. If this happens, the offspring is discarded and the algorithm tries another β. The blend crossover method begins by choosing some parameter α that determines the distance outside the bounds of the two parent variables that the offspring variable may lie. This method allows new values outside of the range of the parents without letting the algorithm stray too far.

The method used for us is a combination of an extrapolation method with a crossover method. We want to find a way to closely mimic the advantages of the binary GA mating scheme. It begins by randomly selecting a variable in the first pair of parents to be the crossover point:

$$\alpha = \text{roundup}\{\text{random} * N_{\text{var}}\}$$

We'll let

$$parent_1 = [p_{m1} p_{m2} \ldots p_{m\alpha} \ldots p_{mN\,\text{var}}]$$
$$parent_2 = [p_{d1} p_{d2} \ldots p_{d\alpha} \ldots p_{dN\,\text{var}}]$$

Where the *m* and *d* subscripts discriminate between the *mom* and the *dad* parent. Then the selected variables are combined to form new variables that will appear in the children:

$$p_{new1} = p_{m\alpha} - \beta[p_{m\alpha} - p_{d\alpha}]$$
$$p_{new2} = p_{d\alpha} + \beta[p_{m\alpha} - p_{d\alpha}]$$

Where β is also a random value between 0 and 1. The final step is to complete the crossover with the rest of the chromosome as before:

$$offspring_1 = [p_{m1} p_{m2} \ldots p_{new1} \ldots p_{dN\,\text{var}}]$$
$$offspring_2 = [p_{d1} p_{d2} \ldots p_{new2} \ldots p_{mN\,\text{var}}]$$

If the first variable of the chromosomes is selected, then only the variables to the right to the selected variable are swapped. If the last variable of the chromosomes is selected, then only the variables to the left of the selected variable are swapped. This method does not allow offspring variables outside the bounds set by the parent unless $\beta > 1$.

For our example problem, the first set of parents are given by

$$chromosome_2 = [0.1876, 8.9371]$$
$$chromosome_3 = [2.6974, 6.2647]$$

A random number generator selects p_1 as the location of the crossover. The random number selected for β is $\beta = 0.0272$. the new offspring are given by

$$offspring_1 = [0.18758 - 0.0272 \times 0.18758 + 0.0272 \times 2.6974, 6.2647]$$
$$= [0.2558, 6.2647]$$

$$offspring_2 = [2.6974 + 0.0272 \times 0.18758 - 0.0272 \times 2.6974, 8.9371]$$
$$= [2.6292, 8.9371]$$

Continuing this process once more with a $\beta=0.7898$. The new offspring are given by

$$\begin{aligned} offspring_3 &= [2.6974 - 0.7898 \times 2.6974 + 0.7898 \times 7.7246, 6.2647] \\ &= [6.6676, 5.5655] \end{aligned}$$

$$\begin{aligned} offspring_4 &= [7.7246 + 0.7898 \times 2.6974 - 0.7898 \times 7.7246, 8.9371] \\ &= [3.7544, 6.2647] \end{aligned}$$

1.7. Mutations

To avoid some problems of overly fast convergence, we force the routine to explore other areas of the cost surface by randomly introducing changes, or mutations, in some of the variables.

As with the binary GA, we chose a mutation rate of 20%. Multiplying the mutation rate by the total number of variables that can be mutated in the population gives $0.20 \times 7 \times 2 \approx 3$ mutations. Next random numbers are chosen to select the row and the columns of the variables to be mutated. A mutated variable is replaced by a new random variable. The following pairs were randomly selected:

$$mrow = [4 \quad 4 \quad 7]$$
$$mcol = [1 \quad 2 \quad 1]$$

The first random pair is (4,1). Thus the value in row 4 and column 1 of the population matrix is replaced with a uniform random number between 1 and 10:

$$5.6130 \Rightarrow 9.8190$$

Mutations occur two more times. The first two columns in table 4 show the population after mating. The next two columns display the population after mutation. Associated costs after the mutations appear in the last column.

Population after Mating		Population after Mutations		
x	y	x	y	cost
7.7246	5.5655	7.7246	5.5655	–9.8884
0.18758	8.9371	0.18758	8.9371	–8.0108
2.6974	6.2647	2.6974	6.2647	–2.8957
5.613	0.12885	*9.819*	*7.1315*	17.601
0.2558	6.2647	0.2558	6.2647	–0.03688
2.6292	8.9371	2.6292	8.9371	–10.472
6.6676	5.5655	*9.1602*	5.5655	–14.05
3.7544	6.2647	3.7544	6.2647	2.1359

Table 4. Mutating the population.

Figure 3 shows the distribution of chromosomes after the first generation.

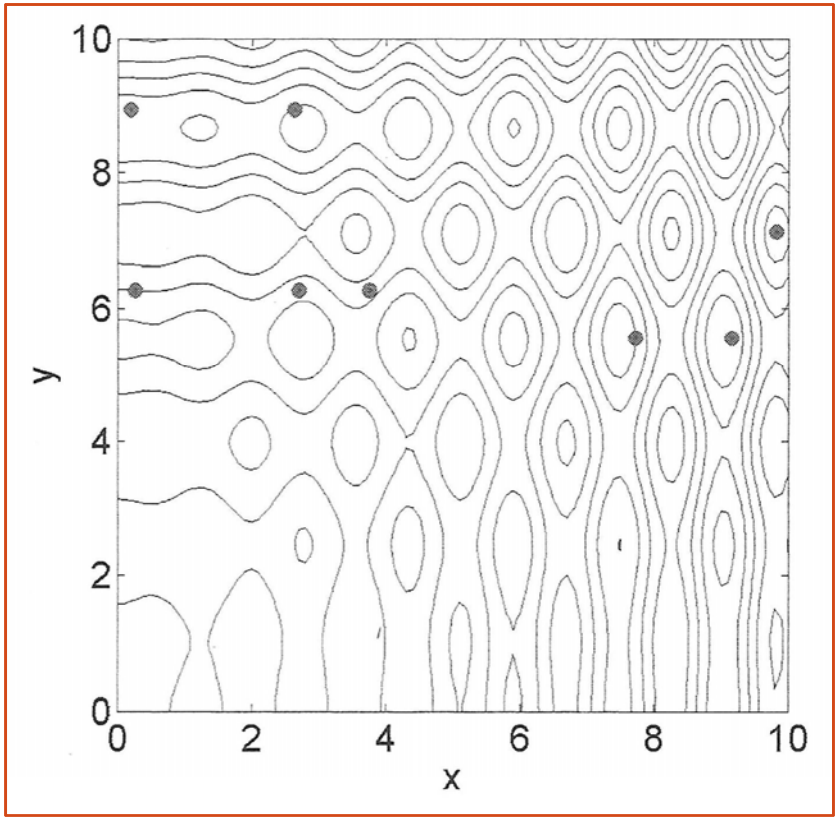

Figure 3. Contour plot of the cost function with the population after the first generation.

Most users of the continues GA add a normally distributed random number to the variable selected for mutation:

$$p_n^{'} = p_n + \sigma N_n(0,1)$$

Where

σ: standard deviation of the normal distribution

$N_n(0,1)$: standard normal distribution (mean=0 and variance=1)

1.8. The Next Generation

The process described is iterated until an acceptable solution is found. For our example, the starting population for the next generation is shown in table 5 after ranking. The population at the end of generation 2 is shown in table 6. Table 7 is the ranked population at the beginning of the generation 3. After mating mutation and ranking, the final population after three generations is shown in table 8 and figure 4.

x	y	Cost
9.1602	5.5655	–14.05
2.6292	8.9371	–10.472
7.7246	5.5655	–9.8884
0.18758	8.9371	–8.0108
2.6974	6.2647	–2.8957
0.2558	6.2647	–0.03688
3.7544	6.2647	2.1359
9.819	7.1315	17.601

Table 5. New ranked population at the start of the second generation.

x	y	Cost
9.1602	5.5655	–14.05
2.6292	8.9371	–10.472
7.7246	6.4764	–1.1376
0.18758	8.9371	–8.0108
2.6292	5.8134	–7.496
9.1602	8.6892	–17.494
7.7246	8.6806	–13.339
4.4042	7.969	–6.1528

Table 6. Population after crossover and mutation in the second generation.

x	y	Cost
9.1602	8.6892	–17.494
9.1602	5.5655	–14.05
7.7246	8.6806	–13.339
2.6292	8.9371	–10.472
0.18758	8.9371	–8.0108
2.6292	5.8134	–7.496
4.4042	7.969	–6.1528
7.7246	6.4764	–1.137

Table 7. New ranked population at the start of the third generation.

x	y	Cost
9.0215	8.6806	–18.53
9.1602	8.6892	–17.494
9.1602	8.323	–15.366
9.1602	5.5655	–14.05
9.1602	8.1917	–13.618
2.6292	8.9371	–10.472
7.7246	1.8372	–4.849
7.8633	3.995	4.6471

Table 8. Ranking of generation 3 from least to most cost.

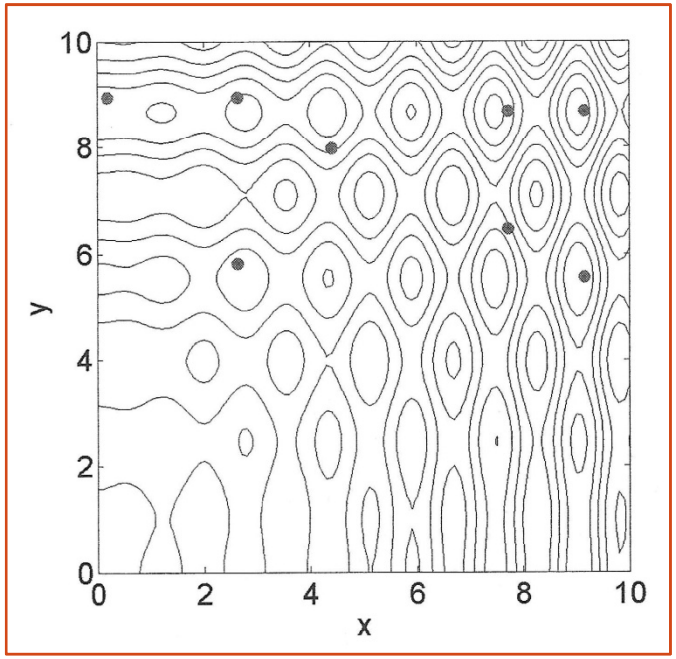

Figure 4. Contour plot of the cost function with the population after the second generation.

1.9. Convergence

This run of the algorithm found the minimum cost (-18.53) in three generations. Members of the population are shown as large dots on the cost surface contour plot in figures 2 and 5. By the end of the second generation, chromosomes are in the basins of the four lowest minima on the cost surface. The global minimum of -18.5 is found in generation 3. All but two of the population members are in the valley of the global minimum in the final generation. Figure 6 is a plot of the mean and minimum cost of each generation.

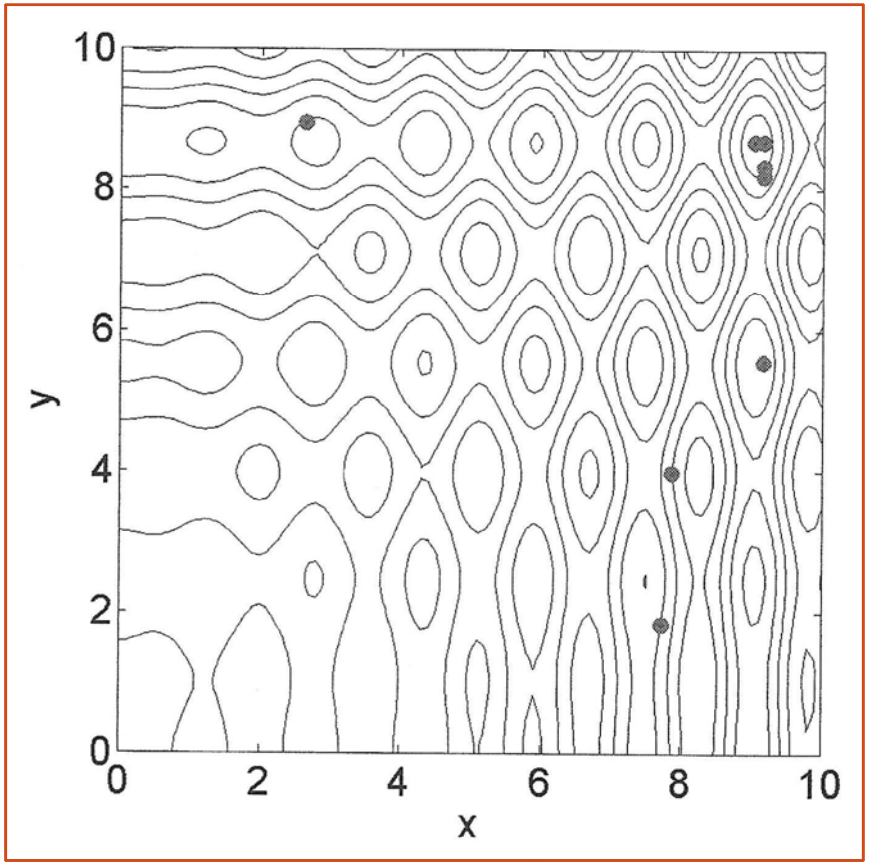

Figure 5. Contour plot of the cost function with the population after the third and final generation.

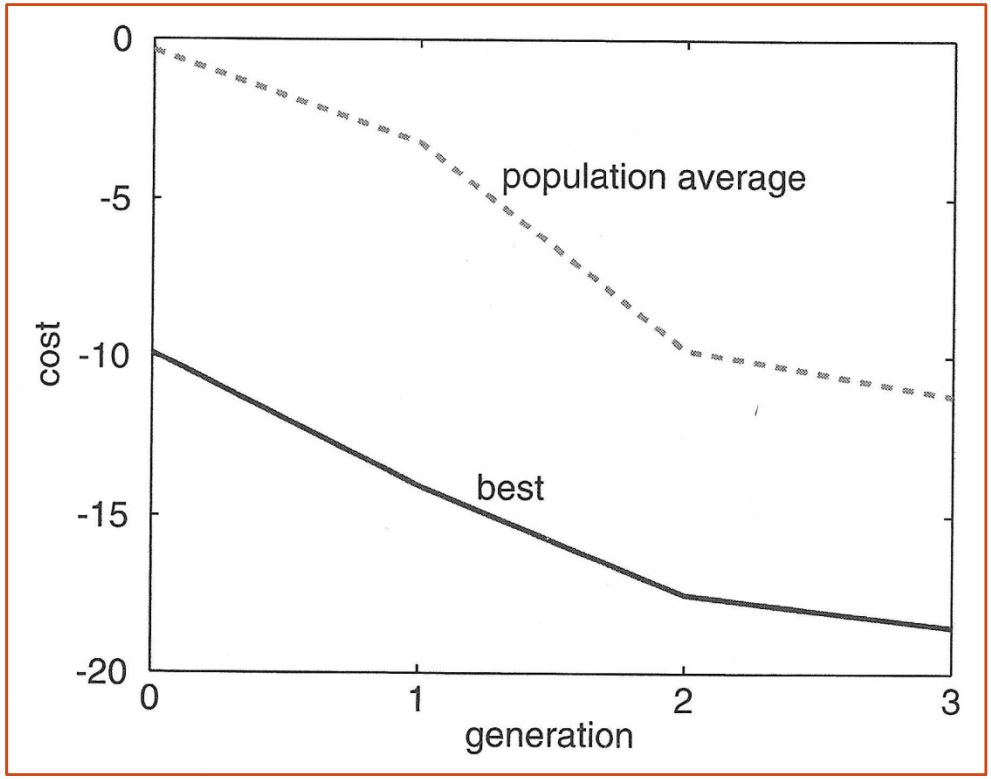

Figure 6. Plot of the minimum and mean costs as a function of generation. The algorithm converged in three generations

2. A Parting Look

The binary GA could have been used in this example as well as a continuous GA. Since the problem used continuous variables, it seemed more natural to use the continuous GA.

3. Bibliography

- Randy L. Haupt and Sue Ellen Haupt. “Practical Genetic Algorithms”. Second edition. 2004.