(Brest State Technical University 2006 Fall Semester: Course Practice)

Contemporary Intelligent Information Technology

Akira Imada
(e-mail akira@bstu.by)

This document is still under construction and was lastly modified on

December 6, 2007

1 The Simplest Test Function — Sphere Model

The first task of this practice is to obtain the minimul value of a multi-dimensional function.

To be more specific, we now assume that we have the following function defined in a 20-dimensional space:

$$y = x_1^2 + x_2^2 + x_3^2 + \cdots + x_{20}^2. \tag{1}$$

Then obtain which point of $(x_1, x_2, x_3, \cdots, x_{20})$ gives a minimum value of y and how much is the value of minimum y.

Try the following algorithm.

Algorithm 1 (The simplest example)

1. *Create 100 chromosomes at random.*

   cdot The number of gene is 20. Thus our chromosomes here have the form $(x_1, x_2, x_3, \cdots, x_{20})$.
   cdot Assume here each of x_i takes the continuous value from -1 to 1, that is $-1 < x_i < 1$.

2. *Calculate fitness value by $y = x_1^2 + x_2^2 + x_3^2 + \cdots + x_{20}^2$. Note that the smalle the better.*
3. *Select 2 chromosomes at random from the better half of the population of 100 chromosomes.*
4. *Create a child chromosome by a crossover.*
5. *Give the child a mutation*
6. *Repeat from 2. to 5. above 100 times and create the next generation.*
7. *Repeat 6. until the fitness value reaches 0.*

The the question is as follows.

Excersize 1 (Obtaining the global minimum)
(1) Plot the average fitness value of all the 100 chromosomes versus generation. (2) Also plot the minimum fitness value of each generation.

2 Neural Networks for XOR

This is probably a simplest examples. We now try the below.

- Output Y of the neuron which receives weighted-sum of the signals X_i from other N neurons is:

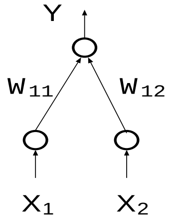
$$Y = \mathrm{sgn}(\sum_{i=1}^{N} w_i X_i - \theta)$$

where $\mathrm{sgn}(x) = 1$ if $x \geq 0$ and 0 otherwise, and w_i and θ are called *weight* and *threshold*, respectively.

- NN to solve AND & OR.

AND

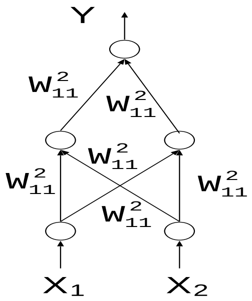
X_1	X_2	Y
0	0	0
0	1	0
1	0	0
1	1	1



· Well, how about NN to solve XOR?

XOR

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0



Exercise 1 *Obtain six weights values so that the above NN function as XOR.*

Exercise 2 *Create Pseudo code for EC to obtain the six weights above.*

3 TSP or Knapsack Problem

4 Multi Modal Genetic Algorithms – What if we have multipul different meaningful solution?

4.1 Target Functions

Assuming our goal is maximization, that is, we want to know when y takes the maximum value and for which x, we try two test functions.

$$y = \sin^6(5\pi x) \tag{2}$$

and

$$y = -2((x-0.2)/0.8)^2 \sin^6(5\pi x) \tag{3}$$

Now take a look what do these two function look like.

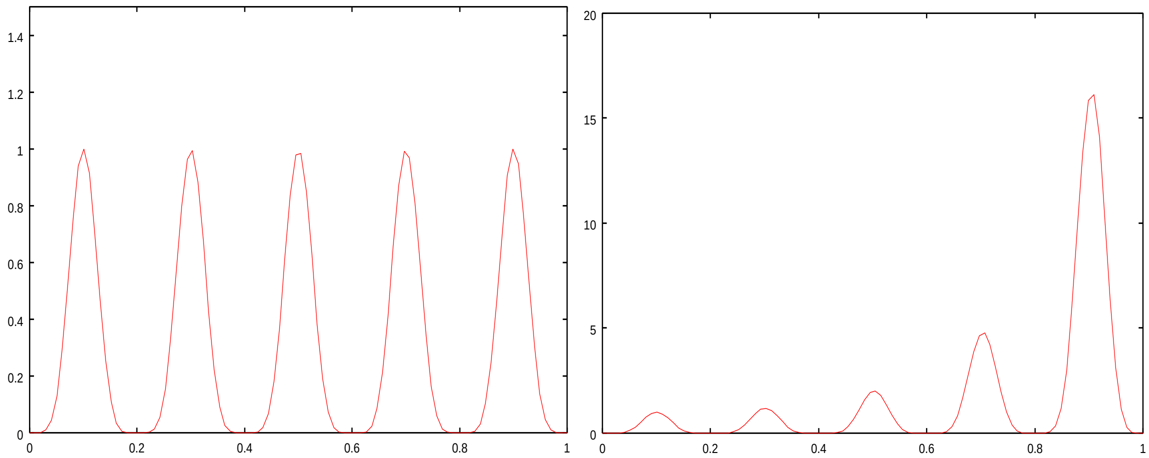


Figure 1: A multi-peak 2-D function and its variation

4.2 Encoding

The question is how we design our chromosomes. In the multi-dimensional function $y = f(x_1, x_2, \cdots, x_n)$ our genes might be continuos value each of which corresponds to the independent variable x_i $(i = 1, 2, \cdots n)$, that is, our chromosomes are made up of n genes. On the other hand how should we design our chromosomes when the number of independent variable is only one? A chromosome with only one gene? Then how we crossover two parents?

O.K. we usually use binary chromosome in this situation. Any (decimal) real-valued variable $x_i \in [a, b]$ could be encoded by n-bit binary strings where a and b is represented by $(00 \cdots 0)$ and $(11 \cdots 1)$, respectivley, and therfore accuracy (or granularity) is $(b-a)/(2^n-1)$. For example, if our concern is the above $x \in [0, 1]$ then 10 bit of binary strings from 0000000000 to 1111111111 are expresses decimals with the precision of 1/1024.

4.3 Two Algorithms

Here, we have two algorithms for the current purpose of finding multiple solutions at a run.

4.3.1 Fitness Sharing

Fitness of each individual is derated by an amount related to the number of similar individuals in the population. That is, shared fitness $F_s(i)$ of the individual i is

$$F_s(i) = \frac{F(i)}{\sum_{j=1}^{\mu} s(d_{ij})}$$

where $F(i)$ is fitness of individual i; d_{ij} is distance between individual i and j; Typically d_{ij} is *Hamming distance* if in *genotypic space Euclidean distance* if in *phenotypic space* and $s(\cdot)$ is called *sharing function* and defined as:

$$s(d_{ij}) = \begin{cases} 1 - (d_{ij}/\sigma_{\text{share}})^{\alpha} & \text{if } d_{ij} < \sigma_{\text{share}} \\ 0 & \text{otherwise} \end{cases}$$

where σ_{share} is interpreted as size of niche, and α determines the shape of the function. The denominator is called *niche count*. You see shape dependency of $s(d_{ij})$ on α in Figure 4.3.1.

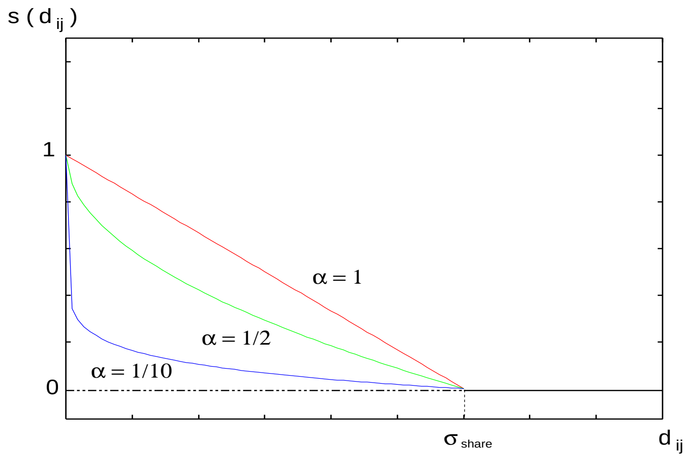


Figure 2: A shape dependency of $s(d_{ij})$ on α.

To be short (not so short though): Similar individual should share fitness. The number of individuals that can stay around any one of peaks (niche) is limited.
The number of individuals stay near any peak will theoretically be proportional to the *hight* of the peak

4.3.2 Deterministic Crowding

If the parents will be replaced or not with their childeren will be determined under a criteria of the distance between parents and children.

Algorithm Assuming crossover, mutation and fitness function are already defiened

1. *Choose two parents,* p_1 *and* p_2*, at random, with no parent being chosen more than once.*
2. *Produce two children,* c'_1 *and* c'_2*.*
3. *Mutate the children yielding* c_1 *and* c_2*, with a crossover.*
4. *Replace parent with child as follows:*
   - *IF* $d(p_1, c_1) + d(p_2, c_2) > d(p_1, c_2) + d(p_2, c_1)$
     - *IF* $f(c_1) > f(p_1)$ *THEN replace* p_1 *with* c_1
     - *IF* $f(c_2) > f(p_2)$ *THEN replace* p_2 *with* c_2
   - *ELSE*
     - *IF* $f(c_2) > f(p_1)$ *THEN replace* p_1 *with* c_2
     - *IF* $f(c_1) > f(p_2)$ *THEN replace* p_2 *with* c_1

where $d(\zeta_1, \zeta_2)$ *is the Hamming distance between two points* (ζ_1, ζ_2) *in pattern configuration space. The process of producing child is repeated until all the population have taken part in the process. Then the cycle of reconstructing a new population and restarting the search is repeated until all the global optima are found or a set maximum number of generation has been reached.*

4.4 Results you should show.

Hopefully you apply two algorithms to each of two test functions. Besides fitness-generation graph, as usual, you try visualize how your individual change their location as generation goes.

That is to say, show all points of individuals in, say, every 20 generations in order to see how theyconverge to the peaks.