REAL-CODED GENETIC ALGORITHMS

Nedim TUTKUN
nedimtutkun@gmail.com
Outlines

• Variables and Cost Function
• Variable Encoding
• Initial Function
• Cost Function
• Natural Selection
• Pairing
• Crossover
• Mutation
Variables and Cost Function

- The goal is to solve some optimization problems where we search for an optimal (minimum) solution in terms of the variables of the problem.
- 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 has $N_{\text{var}}$ variables, that is, an $N$-dimensional optimization problem given by $p_1, p_2, \ldots, p_{N_{\text{var}}}$ then the chromosome is written as an array with $1 \times N_{\text{var}}$ elements so that $\text{chromosome} = [p_1, p_2, p_3, \ldots, p_{N_{\text{var}}}]$
Variables and Cost Function

- 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 $p_1, p_2, \ldots, p_{N\text{var}}$.
- Cost or fitness values can be found by

$$\text{Cost} = f(\text{chromosome}) = f (p_1, p_2, p_3, \ldots, p_{N\text{var}})$$
Consider the cost function.

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

Subject to the constraints: \(0 \leq x, y \leq 10\)

Since \(f(x, y)\) is a function of \(x\) and \(y\) only, the variables of the function are

\[
\text{chromosome} = [x, y]
\]
Variables and Cost Function

- A mesh and contour plot of the cost function appears as Figs 1 and 2. This cost function is considerably more complex than any cost function with single variable.
- As seen from the Fig 1, there are several peaks and valleys in the cost function mesh plot.
- Assume that our goal is to find the **global minimum** value of $f(x, y)$ because traditional minimum-seeking methods are usually unable to find it.
Mesh Plot of the Cost Function

Fig. 1. \( f(x_1, x_2) = x_1 \sin(4x_1) + 1.1x_2 \sin(2x_2) \)
Fig. 2. Contour plot of $f(x_1,x_2)=x_1 \sin(4x_1) + 1.1x_2 \sin(2x_2)$
Variable Encoding

- Here is where we begin to see the differences from the binary encoding because 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 given.
- However although the values are continuous, a digital computer represents numbers by a finite number of bits.
Variable Encoding

- When we refer to the continuous GA, we mean the computer uses its internal precision and roundoff to define the precision of the value.
- We should be aware that the algorithm is always limited in precision to the roundoff 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 the initial search region is not known, there must be enough diversity in the initial population to explore a reasonably sized variable space.
Initial Population

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

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

- All variables are normalized to have values between 0 and 1, which is the range of a uniform random number generator.
- If the range of values is between \( a \) and \( b \), then the unnormalized values are given by

\[
p = (b - a) \times \text{rand} + a
\]

- \( a \): lowest number in the variable range
- \( b \): highest number in the variable range
- \( \text{rand} \): a uniformly distributed random number between 0 and 1
Initial Population

- The individual chromosomes are not all created equal and each one’s worth is assessed by the cost function.
- For this stage, the chromosomes are passed to the cost function for evaluation.
- We begin solving $f(x,y)=x\sin(4x)+1.1y\sin(2y)$ by filling a $N_{pop} \times N_{var}$ matrix with uniform random numbers between 0 and 10.
Initial Population

- Figure 3 shows the initial random population for the $N_{\text{pop}} = 10$ chromosomes.
- Population values are listed in Table 1. We see widely scattered population members that well sample the values of the cost function.
- None of the initial guesses are particularly close to the global minimum.
Contour Plot of the Cost Function

Fig. 3. Initial population of 10 on the contour plot of the cost function.
## Natural Selection

Table 1. Initial Population of 8 Random Chromosomes & Their Cost

<table>
<thead>
<tr>
<th>Number</th>
<th>x</th>
<th>y</th>
<th>F(x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6.9745</td>
<td>0.8342</td>
<td>3.4766</td>
</tr>
<tr>
<td>2</td>
<td>0.30359</td>
<td>9.6828</td>
<td>5.5408</td>
</tr>
<tr>
<td>3</td>
<td>2.402</td>
<td>9.3359</td>
<td>-2.2528</td>
</tr>
<tr>
<td>4</td>
<td>0.18758</td>
<td>8.9371</td>
<td>-8.0108</td>
</tr>
<tr>
<td>5</td>
<td>2.6974</td>
<td>6.2647</td>
<td>-2.8957</td>
</tr>
<tr>
<td>6</td>
<td>5.613</td>
<td>0.1289</td>
<td>-2.4601</td>
</tr>
<tr>
<td>7</td>
<td>7.7246</td>
<td>5.5655</td>
<td>-9.8884</td>
</tr>
<tr>
<td>8</td>
<td>6.8537</td>
<td>9.8784</td>
<td>13.752</td>
</tr>
</tbody>
</table>
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 $N_{\text{pop}}$ costs and associated chromosomes are ranked from lowest cost to highest cost.
- Take the half of it and leave the rest die off.
Natural Selection

- 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 of the survivors are considered fit enough to mate.
- Of the $N_{\text{pop}}$ chromosomes in a given generation, only the top $N_{\text{keep}}$ are kept for mating and the rest are discarded to make room for the new offspring.
**Natural Selection**

- In our example the mean of the cost function for the population of 8 was -0.3423 and the best cost was -9.8884.
- After discarding the bottom half the mean of the population is -5.8138.
- The natural selection results represented to five significant digits are shown in Table 2.
Natural Selection

Table 2. Surviving Chromosomes after a 50% Selection Rate

<table>
<thead>
<tr>
<th>Number</th>
<th>x</th>
<th>y</th>
<th>F(x,y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7.7246</td>
<td>5.5655</td>
<td>-9.8884</td>
</tr>
<tr>
<td>2</td>
<td>0.1876</td>
<td>8.9371</td>
<td>-8.0108</td>
</tr>
<tr>
<td>3</td>
<td>2.6974</td>
<td>6.2647</td>
<td>-2.8957</td>
</tr>
<tr>
<td>4</td>
<td>5.6130</td>
<td>0.12885</td>
<td>-2.4601</td>
</tr>
</tbody>
</table>
The example presented here uses rank weighting with the probabilities given by the above equation.

A random number generator produced the following two pairs of random numbers: (0.6710, 0.8124) and (0.7930, 0.3039). Using these random pairs the following chromosomes were randomly selected to mate:

- mothers = [2 3] and fathers = [3 1].
Pairing

Table 3a. Rank Weighting

<table>
<thead>
<tr>
<th>n</th>
<th>Chromosome</th>
<th>$P_n$</th>
<th>$\sum_{i=1}^{n} P_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00110010001100</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>2</td>
<td>1110110000000001</td>
<td>0.3</td>
<td>0.7</td>
</tr>
<tr>
<td>3</td>
<td>00101111001000</td>
<td>0.2</td>
<td>0.9</td>
</tr>
<tr>
<td>4</td>
<td>00101111000110</td>
<td>0.1</td>
<td>1.0</td>
</tr>
</tbody>
</table>

- Starting at the top of the list, the first chromosome with a cumulative probability that is greater than the random number is selected for the mating pool.
Pairing

For instance, if the random number is $r = 0.577$, then $0.4 < r \leq 0.7$, so the chromosome is selected. If a chromosome is paired with itself, there are several alternatives. First, let it go. It just means there are three of these chromosomes in the next generation.

### Table 3b. Rank Weighting

<table>
<thead>
<tr>
<th>$n$</th>
<th>Chromosome</th>
<th>$P_n$</th>
<th>$\sum_{i=1}^{n} P_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00110010001100</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>2</td>
<td>11101100000001</td>
<td>0.3</td>
<td>0.7</td>
</tr>
<tr>
<td>3</td>
<td>00101111001000</td>
<td>0.2</td>
<td>0.9</td>
</tr>
<tr>
<td>4</td>
<td>00101111000110</td>
<td>0.1</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Pairing

- The $N_{\text{keep}} = 4$ most-fit chromosomes form the mating pool. Two mothers and fathers pair in some random fashion such as mothers = $[2 \ 3]$ and fathers = $[3 \ 1]$.

- Each pair produces two offspring that contain traits from each parent. In addition, the parents survive to be part of the next generation.

- It can be said that the offsprings carry the traits of the parents for the next generation.
Pairing

- Thus chromosome2 mates with chromosome3, and so forth.
- The ma and pa vectors contain the numbers corresponding to the chromosomes selected for mating.
- Table 4 summarizes the results.
Table 4. Pairing and Mating Process of Single-Point Crossover Chromosome Family Binary String Cost

<table>
<thead>
<tr>
<th>Number</th>
<th>x</th>
<th>y</th>
<th>F(x,y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>ma1</td>
<td>0.18758</td>
<td>8.9371</td>
</tr>
<tr>
<td>3</td>
<td>pa1</td>
<td>2.6974</td>
<td>6.2647</td>
</tr>
<tr>
<td>5</td>
<td>offspring1</td>
<td>0.2558</td>
<td>6.2647</td>
</tr>
<tr>
<td>6</td>
<td>offspring2</td>
<td>2.6292</td>
<td>8.9371</td>
</tr>
<tr>
<td>3</td>
<td>ma2</td>
<td>2.6974</td>
<td>6.2647</td>
</tr>
<tr>
<td>1</td>
<td>pa2</td>
<td>7.7246</td>
<td>5.5655</td>
</tr>
<tr>
<td>7</td>
<td>offspring3</td>
<td>6.6676</td>
<td>5.5655</td>
</tr>
<tr>
<td>8</td>
<td>offspring4</td>
<td>3.7544</td>
<td>6.2646</td>
</tr>
</tbody>
</table>
Mating

- As for the binary algorithm, two parents are chosen, and the offspring are some combination of these parents.
- Many different approaches have been tried for crossing over in continuous GAs.
- Adewuya (1996) reviews some of the methods.
- Several interesting methods are demonstrated by Michalewicz (1994).
Mating

- 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.

\[
\text{parent}_1 = [p_{m1}, p_{m2}, p_{m3}, p_{m4}, p_{m5}, p_{m6}, \ldots, p_{mN_{\text{var}}}] \\
\text{parent}_2 = [p_{d1}, p_{d2}, p_{d3}, p_{d4}, p_{d5}, p_{d6}, \ldots, p_{dN_{\text{var}}}] 
\]
Mating

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

$$\text{offspring}_1 = [p_{m1}, p_{m2}, \hat{p}_{d3}, p_{d4}, \hat{p}_{m5}, p_{m6}, \ldots, p_{mN_{var}}]$$

$$\text{offspring}_2 = [p_{d1}, p_{d2}, \hat{p}_{m3}, p_{m4}, \hat{p}_{d5}, p_{d6}, \ldots, p_{dN_{var}}]$$

- 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 information between the two parents.
This method is called uniform crossover:

\[
\text{offspring}_1 = [p_{m1}, p_{d2}, p_{d3}, p_{d4}, p_{d5}, p_{m6}, \ldots, p_{dN_{var}}]
\]

\[
\text{offspring}_2 = [p_{d1}, p_{m2}, p_{m3}, p_{m4}, p_{m5}, p_{d6}, \ldots, p_{mN_{var}}]
\]

The problem with these point crossover methods is that no new information is introduced, that is, each continuous value that was randomly initiated in the initial population is propagated to the next generation, only in different combinations.

Although this strategy worked fine for binary representations, there is now a continuum of values, and in this continuum we are merely interchanging two data points.
Mating

- These approaches totally rely on mutation to introduce new genetic material.

- The blending methods solves 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, $p_{\text{new}}$, comes from a combination of the two corresponding offspring variable values (Radcliff, 1991).
**Mating**

\[ p_{\text{new}} = \beta p_{mn} + (1 - \beta)p_{dn} \]

\( \beta = \) 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 (i.e., replacing \( \beta \) by \( 1 - \beta \)). If \( \beta = 1 \), then \( p_{mn} \) 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 $\beta = 0.5$ (Davis, 1991), the result is an average of the variables of the two parents. This method is demonstrated to work well on several interesting problems by Michalewicz (1994).

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 $N_{\text{var}}$ values where all variables are linear combinations of those of the two parents.
Mating

- The variables can be blended by using the same $\beta$ for each variable or by choosing different $\beta$’s for each variable.

- These blending methods effectively combine the information from the two parents and choose values of the variables between the values bracketed by the parents.

- However, they do not allow introduction of values beyond the extremes already represented in the population. To do this requires an extrapolating method. The simplest of these methods is linear crossover (Wright, 1991).
**Mating**

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

\[
\begin{align*}
    p_{new1} &= 0.5p_{mn} + 0.5p_{dn} \\
    p_{new2} &= 1.5p_{mn} - 0.5p_{dn} \\
    p_{new3} &= -0.5p_{mn} + 1.5p_{dn}
\end{align*}
\]

- 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.
**Mating**

- Heuristic crossover (Michalewicz, 1991) is variation where some random number, $\beta$, 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 $\beta$ for each variable. This method also allows generation of offspring outside of the values of the two parent variables.
Mating

- Sometimes values are generated outside of the allowed range. If this happens, the offspring is discarded and the algorithm tries another $\beta$.

- The blend crossover (BLX-$\alpha$) method (Eshelman and Shaffer, 1993) begins by choosing some parameter $\alpha$ that determines the distance outside the bounds of the two parent variables that the offspring variable may lie.

- This method allows new values outside the range of the parents without letting the algorithm stray too far. Many codes combine the various methods to use the strengths of each.
Mating

- New methods, such as quadratic crossover (Adewuya, 1996), do a numerical fit to the fitness function. Three parents are necessary to perform a quadratic fit. The algorithm used here is a combination of an extrapolation method with a crossover method.

- Sometimes it is better 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

\[
\alpha = \text{roundup}\{\text{random} \times N_{\text{var}}\}
\]
Mating

\[
\begin{align*}
\text{parent}_1 &= [p_{m1}p_{m2} \cdots p_{m\alpha} \cdots p_{mN_{\text{var}}}] \\
\text{parent}_2 &= [p_{d1}p_{d2} \cdots p_{d\alpha} \cdots p_{dN_{\text{var}}}] 
\end{align*}
\]

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:

\[
\begin{align*}
\text{new}_1 &= p_{m\alpha} - \beta(p_{m\alpha} - p_{d\alpha}) \\
\text{new}_2 &= p_{d\alpha} + \beta(p_{m\alpha} - p_{d\alpha})
\end{align*}
\]

where \( \beta \) is also a random value between 0 and 1.
Mating

- The final step is to complete the crossover with the rest of the chromosome as before:

\[
\text{offspring}_1 = [p_{m1} p_{m2} \cdots p_{new1} \cdots p_{dN_{var}}]
\]

\[
\text{offspring}_2 = [p_{d1} p_{d2} \cdots p_{new2} \cdots p_{mN_{var}}]
\]

- If the first variable of the chromosomes is selected, then only the variables to the right of 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 \( \beta > 1 \).
Mating

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

\[
\text{chromosome}_2 = [0.1876, 8.9371] \\
\text{chromosome}_3 = [2.6974, 6.2647]
\]

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

\[
\text{offspring}_1 = [0.18758 - 0.0272 \times 0.18758 + 0.0272 \times 2.6974, 6.2647]  \\
\quad = [0.2558, 6.2647]  \\
\text{offspring}_2 = [2.6974 + 0.0272 \times 0.18758 - 0.0272 \times 2.6974, 8.9371]  \\
\quad = [2.6292, 8.9371]
\]
Mating

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

\[
\text{offspring}_3 = [2.6974 - 0.7898 \times 2.6974 + 0.7898 \times 7.7246, 6.2647] = [6.6676, 5.5655]
\]

\[
\text{offspring}_4 = [7.7246 + 0.7898 \times 2.6974 - 0.7898 \times 7.7246, 8.9371] = [3.7544, 6.2647]
\]
Mutations

- Here, we can sometimes find our method working too well. If care is not taken, the GA can converge too quickly into one region of the cost surface. If this area is in the region of the global minimum, that is good.

- However, some functions, such as the one we are modeling, have many local minima. If we do nothing to solve this tendency to converge quickly, we could end up in a local rather than a global minimum.

- To avoid this problem of 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.
Mutations

- For the binary GA, this amounted to just changing a bit from a 0 to a 1, and vice versa. The basic method of mutation is not much more complicated for the continuous GA. For more complicated methods, see Michalewicz (1994).

- 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 columns of the variables to be mutated. A mutated variable is replaced by a new random variable.
Mutations

- The following pairs were randomly selected:
  
  \[
  \text{mrow} = [4 4 7] \\
  \text{mcol} = [1 2 1]
  \]

- 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:
  
  $$5.6130 \implies 9.8190$$
Mutations

- Mutations occur two more times. The first two columns in Table 5 show the population after mating. The next two columns display the population after mutation. Associated costs after the mutations appear in the last column. The mutated values in Table 5 appear in italics.

- Note that the first chromosome is not mutated due to elitism. The mean for this population is -3.202. The third offspring (row 7) has the best cost due to the crossover and mutation.

- If the x-value were not mutated, then the chromosome would have a cost of 0.6 and would have been eliminated in the natural selection process. Figure 4 shows the distribution of chromosomes after the first generation.
## Mutations

### Table 5. Mutating Population

<table>
<thead>
<tr>
<th>Population after Mating</th>
<th>Population after Mutations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$y$</td>
</tr>
<tr>
<td>7.7246</td>
<td>5.5655</td>
</tr>
<tr>
<td>0.18758</td>
<td>8.9371</td>
</tr>
<tr>
<td>2.6974</td>
<td>6.2647</td>
</tr>
<tr>
<td>5.613</td>
<td>0.12885</td>
</tr>
<tr>
<td>0.2558</td>
<td>6.2647</td>
</tr>
<tr>
<td>2.6292</td>
<td>8.9371</td>
</tr>
<tr>
<td>6.6676</td>
<td>5.5655</td>
</tr>
<tr>
<td>3.7544</td>
<td>6.2647</td>
</tr>
</tbody>
</table>
Fig. 4. Contour plot of the cost function with the population after the first generation.
Mutations

- Most users of the continuous GA add a normally distributed random number to the variable selected for mutation.

\[ p_n' = p_n + \sigma N_n(0, 1) \]

\( \sigma \) = standard deviation of the normal distribution, 
\( N_n(0, 1) \) = standard normal distribution (mean = 0 and variance = 1).

- We do not use this technique because a good value for \( \sigma \) must be chosen, the addition of the random number can cause the variable to exceed its bounds, and it takes more computational time.
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 bottom four chromosomes are discarded and replaced by offspring from the top four parents. Another three random variables are selected for mutation from the bottom 7 chromosomes.

- The population at the end of generation 2 is shown in Table 6 and Figure 5. Table 7 is the ranked population at the beginning of generation 3. After mating, mutation, and ranking, the final population after three generations is shown in Table 8 and Figure 6.
### The Next Generation

Table 6. New Ranked Population at the Start of the Second Generation

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1602</td>
<td>5.5655</td>
<td>−14.05</td>
</tr>
<tr>
<td>2.6292</td>
<td>8.9371</td>
<td>−10.472</td>
</tr>
<tr>
<td>7.7246</td>
<td>5.5655</td>
<td>−9.8884</td>
</tr>
<tr>
<td>0.18758</td>
<td>8.9371</td>
<td>−8.0108</td>
</tr>
<tr>
<td>2.6974</td>
<td>6.2647</td>
<td>−2.8957</td>
</tr>
<tr>
<td>0.2558</td>
<td>6.2647</td>
<td>−0.03688</td>
</tr>
<tr>
<td>3.7544</td>
<td>6.2647</td>
<td>2.1359</td>
</tr>
<tr>
<td>9.819</td>
<td>7.1315</td>
<td>17.601</td>
</tr>
</tbody>
</table>
The Next Generation

Table 7. Population after Crossover and Mutation in the Third Generation

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
<td>Cost</td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>9.1602</td>
<td>5.5655</td>
<td>-14.05</td>
</tr>
<tr>
<td>2.6292</td>
<td>8.9371</td>
<td>-10.472</td>
</tr>
<tr>
<td>7.7246</td>
<td>6.4764</td>
<td>-1.1376</td>
</tr>
<tr>
<td>0.18758</td>
<td>8.9371</td>
<td>-8.0108</td>
</tr>
<tr>
<td>2.6292</td>
<td>5.8134</td>
<td>-7.496</td>
</tr>
<tr>
<td>9.1602</td>
<td>8.6892</td>
<td>-17.494</td>
</tr>
<tr>
<td>4.4042</td>
<td>7.969</td>
<td>-6.1528</td>
</tr>
</tbody>
</table>
# The Next Generation

Table 8. New Ranked Population at the Start of the Third Generation

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1602</td>
<td>8.6892</td>
<td>–17.494</td>
</tr>
<tr>
<td>9.1602</td>
<td>5.5655</td>
<td>–14.05</td>
</tr>
<tr>
<td>2.6292</td>
<td>8.9371</td>
<td>–10.472</td>
</tr>
<tr>
<td>0.18758</td>
<td>8.9371</td>
<td>–8.0108</td>
</tr>
<tr>
<td>2.6292</td>
<td>5.8134</td>
<td>–7.496</td>
</tr>
<tr>
<td>4.4042</td>
<td>7.969</td>
<td>–6.1528</td>
</tr>
<tr>
<td>7.7246</td>
<td>6.4764</td>
<td>–1.137</td>
</tr>
</tbody>
</table>
The Next Generation

Table 9. Ranking of Generation 3 from Least to Most Cost

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.0215</td>
<td>8.6806</td>
<td>−18.53</td>
</tr>
<tr>
<td>9.1602</td>
<td>8.6892</td>
<td>−17.494</td>
</tr>
<tr>
<td>9.1602</td>
<td>8.323</td>
<td>−15.366</td>
</tr>
<tr>
<td>9.1602</td>
<td>5.5655</td>
<td>−14.05</td>
</tr>
<tr>
<td>9.1602</td>
<td>8.1917</td>
<td>−13.618</td>
</tr>
<tr>
<td>2.6292</td>
<td>8.9371</td>
<td>−10.472</td>
</tr>
<tr>
<td>7.7246</td>
<td>1.8372</td>
<td>−4.849</td>
</tr>
<tr>
<td>7.8633</td>
<td>3.995</td>
<td>4.6471</td>
</tr>
</tbody>
</table>
Fig. 5. Contour plot of the cost function with the population after the third generation.
The Next Generation

Fig. 6. Contour plot of the cost function with the population after the second generation.
Fig. 7. Plot of the minimum and mean costs as a function of generation. The algorithm converged in three generations.
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 3 to 6.

- 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 7 is a plot of the mean and minimum cost for each generation. The GA was able to find the global minimum, unlike the Nelder-Mead and other similar algorithms.
```matlab
clear;clc;
% Step 1 : Initialization
a=0;b=10;n=1;
G=50;pm=0.3;pc=.5;N=64;fmax=[];maxfit=0;fave=[];fmin=[];
x=rand(N,n);
for k=1:n
    x(:,k)=linmap(x(:,k),a,b)
% convert chromosome to real number in a range from a to b
end
```
Simple Genetic Algorithms (Matlab Code)

```matlab
for g=1:G
    fprintf('g:%.0f\n',g);
    \% Step 2 : Selection
    f=fitval(x)
    s=selpop(x,f);
    \% Step 3 : Crossover
    c=artxover(s,pc)
    \% Step 4 : Mutation
    x=pertmutate(c,pm,a,b)
    \[maxfit x]=elit(x,maxfit);
    f=fitval(x);
    fmax=[fmax maxfit];
    fave=[fave mean(f)];
    fmin=[fmin min(f)];
end \% end the generation
```
f = fitval(x);
[fmax ind] = max(f);
optx = x(ind, :)
yoptx = fun(optx)
g = 1:G;
plot(g, fmax, g, fave, g, fmin);
xlabel('Generation'); ylabel('Fitness Value');
legend('Max', 'Ave', 'Min', 'location', 'best'); legend boxoff; figure(2);
x1 = [0:.0001:2];
y1 = fun(x1);
plot(x1, y1);
xlabel('x'); ylabel('f(x)');
grid;
function y=artxover(x,pc)
[ro co]=size(x);N=2*ro;
y=zeros(N,co);mateind=[];
for k=1:ro
mateind(k)=ceil(rand*ro); % epleme if
mateind(k)==0,mateind(k)=1;
end
for i=1:2:ro
alfa=rand;
i1=mateind(i);i2=mateind(i+1);
y(ro+i,:)=alfa*x(i1,:)+(1-alfa)*x(i2,:);
y(ro+i+1,:)=alfa*x(i2,:)+(1-alfa)*x(i1,:);
end
y(1:ro,:)=x;
function y=pertmutate(x,pm,a,b)
    y=x;
    [N,n]=size(x);
    nm=ceil((N-1)*n*pm);
    r=ceil(N*rand(nm,1));
    c=ceil(n*rand(nm,1));
    for k=1:length(r)
        if r(k)==1, r(k)=2; end
    end
    for k=1:length(r)
        y(r(k),c(k))=(b-a)*rand+a;
    end
function  y=fun(x)
y=100*exp(-x).*sin(10*x);

function  y=fitval(x)
y=fun(x);

function  y=linmap(x,a,b)
y=x;
xmin=0;
xmax=1;
m=(b-a)/(xmax-xmin);
y=m*(x-xmin)+a;
function [maxfit y]=elit(x,maxfit)
    y=x;
    f=fitval(y);
    [bestfit bestind]=max(f);
    if bestfit>maxfit    % finding maximum
        maxfit=bestfit;
        bestx=x(bestind,:);
        y(1,:)=bestx;
    end

function y=selpop(x,f)
    [s ix]=sort(f,'descend'); % finding maximum
    y=x(ix(1:length(s)/2,:),:);
Simple Genetic Algorithms (Simple Matlab Code)

```matlab
function [maxfit y]=elit(x,maxfit)
y=x;
f=fitval(y);
[bestfit bestind]=max(f);
if bestfit>maxfit  \% finding maximum
    maxfit=bestfit;
    bestx=x(bestind,:);
    y(1,:)=bestx;
end

function y=selpop(x,f)
[s ix]=sort(f, 'descend'); \% finding maximum
y=x(ix(1:length(s)/2,:),:);
```
Simple Genetic Algorithms Results

$$y = 100\cdot \sin(10\cdot x) \cdot \exp(-x)$$

opt\_x = 0.1472

opt\_y = 85.8913

Global maximum

Global minimum
Simple Genetic Algorithms Convergence

$X_{opt}=14.72$, $F_{max}=85.8913$, $P_c=0.5$, $P_m=0.3$
clear;clc;
% Step 1 : Initialization
a=0;b=10;n=2;
G=100;pm=0.03;pc=.8;N=128;maxfit=-1e9;fmax=[];fave=[];fmin=[];
x=rand(N,n);
for k=1:n
  x(:,k)=linmap(x(:,k),a,b);
  % convert chromosome to real number in a range from a to b
end
for g=1:G
    fprintf('g:%.0f\n',g);
    f=fitval(x(:,1),x(:,2));
    s=selpop(x,f);
    % Step 3 : Crossover
    c=artxover(s,pc);
    % Step 4 : Mutation
    x=pertmutate(c,pm,a,b);
    [maxfit x]=elit(x(:,1),x(:,2),maxfit);
    f=fitval(x(:,1),x(:,2));
    fmax=[fmax maxfit];
    fave=[fave mean(f)];
    fmin=[fmin min(f)];
end % end the generation
g=1:G;
plot(g,fmax,g,fave,g,fmin);
xlabel('Generation');ylabel('Fitness');
legend('Max','Ave','Min','location','best');legend boxoff;
axis([0 G min(fmin) 1.1*max(fmax)])
% plot(fittest)
f=fitval(x(:,1),x(:,2));
[fmx ind]=max(f);
optx=x(ind(1),:)
yoptx=fun(optx(:,1),optx(:,2))
Simple Genetic Algorithms Convergence

\[ f(x,y) = x \sin(4x) + 1.1y \sin(2y) \]
Simple Genetic Algorithms Convergence

\[ X_{\text{opt}} = (9.8237, 9.9731), \quad F_{\text{max}} = 19.5804, \quad P_c = 0.8, \quad P_m = 0.03 \]
Simple Genetic Algorithms Convergence

$X_{opt}=(9.7825, 9.9873)$, $F_{max}=19.5996$, $P_c=0.65$, $P_m=0.3$
ASSIGNMENT-3
(16/03/2018-30/03/2018)
Find effects of $P_c$ and $P_m$ parameters on finding global maxima using the real coded Genetic Algorithms.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$P_c$</th>
<th>$P_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>0.6, 0.7, 0.8</td>
<td>0.003, 0.03, 0.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Function</th>
<th>Search Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f = 100 + \sin(x)$</td>
<td>$0 \leq x_j \leq \pi$</td>
</tr>
<tr>
<td>$f = 79 - (x^2 + y^2)$</td>
<td>$-5.12 \leq x, y \leq 5.12$</td>
</tr>
<tr>
<td>$f = 4000 - 100(x_1^2 - x_2)^2 + (1 - x_1)^2$</td>
<td>$-2.048 \leq x_j \leq 2.048$</td>
</tr>
</tbody>
</table>