Module 5
Design for Reliability and Quality

IIT, Bombay

Lecture 5
Design for Optimization

IIT, Bombay

Instructional Objectives

By the end of this lecture, the students are expected to learn,

1. What is an optimization?
2. Different optimization tools utilized to optimize single and multi-variable problems.

Defining Optimum Design

In principle, an optimum design means the best or the most suitable of all the feasible conceptual designs. The optimization is the process of maximizing of a desired quantity or the minimizing of an undesired one. For example, optimization is often used by the mechanical engineers to achieve either a minimum manufacturing cost or a maximum component life. The aerospace engineers may wish to minimize the overall weight of an aircraft. The production engineers would like to design the optimum schedules of various machining operations to minimize the idle time of machines and overall job completion time, and so on.

Tools for Design Optimization

No single optimization method is available for solving all optimization problems in an unique efficient manner. Several optimization methods have been developed till date for solving different types of optimization problems. The optimization methods are generally classified under different groups such as (a) single variable optimization, (b) multi-variable optimization, (c) constrained optimization, (d) specialized optimization, and (e) non-traditional optimization. We would concentrate only single- and multi-variable optimization methods in this lecture and the interested readers can undertake further studies in appropriate references to understand the more advanced optimization techniques.

Single Variable Optimization Methods

This methods deal with the optimization of a single variable. *Figure 5.5.1* depicts various types of extremes that can occur in an objective function, f(x) curve, where x is the design variable. It can be observed from the curve that both the points A and C are mathematical minima. The point A, which is the larger of the two minima, is called a local minimum, while the point C is the global minimum. The point D is referred to a point of inflection. To check whether the objective function is local minimum or local maximum or the inflection point, the optimality criteria is as shown below.

Suppose at point x^*,

$$\frac{\partial f(x)}{\partial x} = 0; \quad \frac{\partial^2 f(x)}{\partial^2 x} = n \qquad (1)$$

If n is odd, x^* is an inflection point while if n is even, x^* is a local optimum. In the case, x^* is a local minima, there are two possibilities. If the second derivative is positive, x^* is a local minimum. However, if the second derivative is negative, x^* is a local maximum.

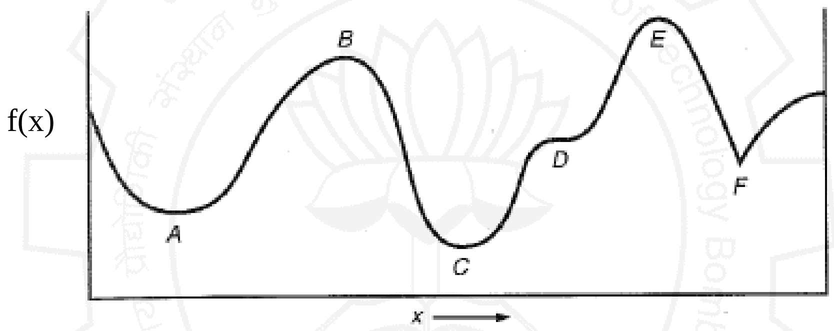


Figure 5.5.1 Different types of extermes in objective function curve [1].

The Single variable optimization is classified into two categories – *direct* and *gradient based search methods*.

Direct search methods

The *direct search methods* use only the objective function values to locate the minimum point. The typical direct search methods include *uniform search*, *uniform dichotomous search*, *sequential dichotomous search*, *Fibonacci search* and *golden section search* methods.

Uniform search

In the uniform search method, the trial points are spaced equally over the allowable range of values. Each point is evaluated in turn in an exhaustive search. For example, the designer wants to optimize the yield of a chemical reaction by varying the concentration of a catalyst, x and x lies over the range 0 to 10. Four experiments are available, and the same are distributed at equivalent spacing over the range $L = 10$ (*Figure 5.5.2*). This divides L into intervals each of width L/n+1, where n is the number of experiments. From inspection of the results at the experimental points, we can conclude that the optimum will not lie in the ranges $x < 2$ or $x > 6$. Therefore, we know the optimum will lie in between the range $2 < x < 6$. So, the range of values that require further search is reduced to 40% of the total range with only four experiments.

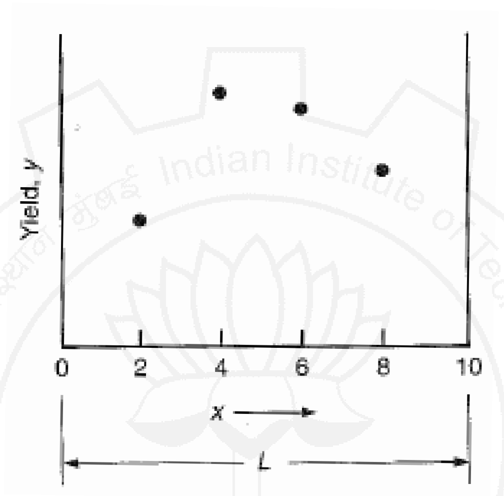


Figure 5.5.2 Example of the uniform search method [1].

Uniform dichotomous search

In the *uniform dichotomous search* method, the experiments are performed in pairs to establish whether the function is increasing or decreasing. Since the search procedure is uniform, the experiments will be spaced evenly over the entire range of values. Using the same example as in *uniform search method*, we place the first pair of experiments at $x = 3.33$ and the other pair at $x = 6.67$. There will be in total n/2 numbers of pairs for n number of experiments. Hence, the range L has to be divided into $(n/2)+1$ intervals each of width $L/\{(n/2)+1)\}$. In this case, the pairs will be $x = 3.30, 3.36$ and $x = 6.64, 6.70$. Since $y_a < y_b$, as observed in *Figure 5.5.3*, the region $0 < x < 3.33$ is excluded from the search region. Furthermore, since $y_c > y_d$, the region $6.67 < x < 10.0$ can also be excluded from the region of search. Hence, the optimum would lie in the interval $3.33 \le x \le 6.67$. Thus, the range of values that require further search is reduced to

30% of the total range with only four experiments which is certainly a better situation that the previous method.

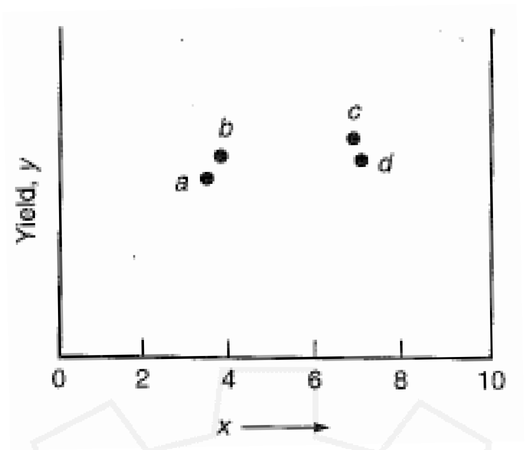


Figure 5.5.3 Example of the uniform dichotomous search methods [1].

Sequential dichotomous search

In the *sequential dichotomous search*, the experiments are done in sequence with each taking benefit from the preceding experiment. Using the same example as in previous methods, we can first run a pair of experiments near to the center of the range i.e. $x = 5.0$. Two experiments are separated just enough in x to be sure that outcomes, if any, are distinguishable. *Figure 5.5.4* depicts that $y_b > y_a$ and hence, we can get rid of the range $x > 5.0$ from any further consideration. The second pair of experiments are run at the center of the remaining interval, $x = 2.5$. Since $y_d > y_c$, we can eliminate the region $0 < x < 2.50$ and the optimum will be examined only in the interval $2.5 \le x \le 5.0$. Thus, four experiments have narrowed the range of values that require further search to 25% of the total range with this search technique.

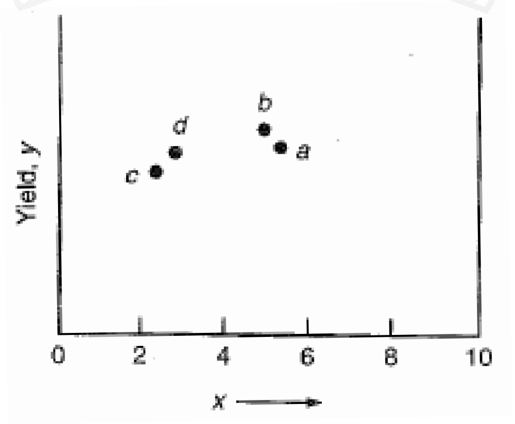


Figure 5.5.4 Example of the sequential dichotomous search methods [1].

Fibonacci search

The *Fibonacci search* is based on the use of the Fibonacci number series. A Fibonacci series is given by $F_n = F_{n-2} + F_{n-1}$, where $F_0 = 1$ and $F_1 = 1$. Thus, the series is as given in *Table 5.5.1*.

Table 5.5.1 Fibonacci number series [1]

n	0	1	2	3	4	5	6	7	8	9 ….
F_n	1	1	2	3	5	8	13	21	34	55 ….

Each nth Fibonacci number is the sum of the preceding two numbers. The *Fibonacci search* begins by placing the experiments at a distance $d_1 = (F_{n-1}/F_n)L$ from each end of the range of values (*Figure 5.5.5*). So, for $n = 4$, $d_1 = (2/5)\times 10 = 4$. Since $y_4 > y_6$, the interval $6 < x < 10$ is eliminated from further consideration. Next, we should set $n_2 = n - 1$ and calculate d_2 as $d_2 = (F_{n-3}/F_{n-1})L = (1/3)\times 6 = 2$. Hence, we should run experiments at $x = 2$ and $x = 4$. Since $y_4 > y_2$, the region $0 < x < 2$ should be eliminated for further consideration. So, the region between $x = 2$ and $x = 6$ is only remaining. Since only three data points are used, the last experiment is performed just to the right of $x = 4$ to determinate whether the optimum is $2 < x < 4$ or $4 < x < 6$. This experiment eliminates the former interval and narrows the optimum within the region $4 < x < 6$ only. In this method, the range of values that require further search is reduced to 20% of the total range with four experiments only.

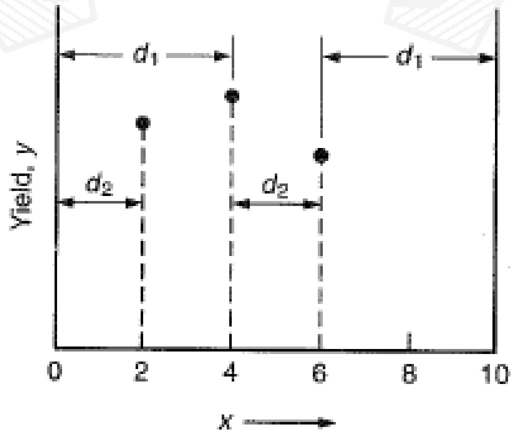


Figure 5.5.5 Example of the Fibonacci search method [1].

Golden section search

The Fibonacci search is very efficient. However, it requires an advance decision on the number of experiments before we have any information about the behavior of the function near the maximum. The *golden section search* technique is based on the ratio of two successive *Fibonacci numbers*, $F_{n-1}/F_n = 0.618$ for all values of $n > 8$. This ratio is referred to as the golden mean. In using the *golden section search* the first two trials are located at 0.618L from either end of the range. Based on the value of y_1 and y_2 an interval is eliminated. With the new reduced interval additional experiments are performed at $\pm 0.618L_2$. This procedure is continued until the maximum is located to with in as small an interval as desired.

Gradient based search method

Gradient based search method use the first and/or the second order derivatives of the objective function to locate the minimum point. One important gradient based search method is the Newton-Raphson-method.

Newton-Raphson method

The *Newton-Raphson method* considers a linear approximation to the first derivative of the function using the Taylor's series expansion. Subsequently, this expression is equated to zero to find the initial guess. If the current point at iteration t is $x^{(t)}$, the point in the next iteration is governed by the nature of the following expression.

$$x^{(t+1)} = x^{(t)} - \frac{f'(x^{(t)})}{f''(x^{(t)})} \qquad (2)$$

The iteration process given by equation (2) is assumed to have converged when the derivative, $f'(x^{(t+1)})$, is close to zero.

$$\left|f'(x^{(t+1)})\right| \le \varepsilon \qquad (3)$$

where ε is a small quantity. Figure 5.5.6 depicts the convergence process in the Newton-Raphson method, where x^* is the true solution.

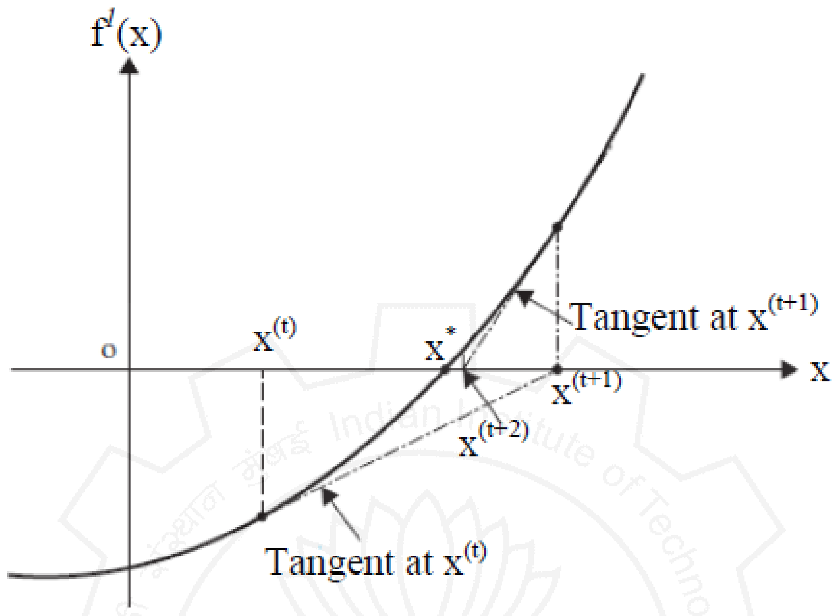


Figure 5.5.6 Iterative process of Newton-Raphson method [2].

Example

Find the minimum of the function

$$f(x) = 0.65 - \frac{0.75}{1+x^2} - \left[0.65 * x * \tan^{-1}\left(\frac{1}{x}\right)\right] \qquad (4)$$

using the Newton-Raphson method with the starting point $x^{(1)} = 0.1$. Use $\varepsilon = 0.01$ in equation (3) for checking the convergence.

Solution

The first and second derivatives of the function f(x) are given by

$$f'(x) = \frac{1.5*x}{(1+x^2)^2} + \frac{0.65*x}{1+x^2} - 0.65*\tan^{-1}\frac{1}{x}$$

$$f''(x) = \frac{1.5*(1-3*x^2)}{(1+x^2)^3} + \frac{0.65*(1-x^2)}{(1+x^2)^2} + \frac{0.65}{1+x^2} = \frac{2.8-3.2*x^2}{(1+x^2)^3} \qquad (5, 6)$$

Iteration 1

$$x^{(1)} = 0.1,\ f(x^{(1)}) = -0.188197,\ f'(x^{(1)}) = -0.744832,\ f''(x^{(1)}) = 2.68659 \qquad (7)$$

We can write using equation (2)

$$x^{(2)} = x^{(1)} - \frac{f'(x^{(1)})}{f''(x^{(1)})} = 0.377241 \tag{8}$$

Convergence check: $|f'(x^{(2)})| = |-0.138230| > \varepsilon$ and hence, further iteration is necessary.

Iteration 2

$$f(x^{(2)}) = -0.303279,\ f'(x^{(2)}) = -0.138230,\ f''(x^{(2)}) = 1.57296 \tag{9}$$

Now, using equation (2), we can write

$$x^{(3)} = x^{(2)} - \frac{f'(x^{(2)})}{f''(x^{(2)})} = 0.465119 \tag{10}$$

Convergence check: $|f'(x^{(3)})| = |-0.0179078| > \varepsilon$ and hence, we need further iteration.

Iteration 3

$$f(x^{(3)}) = -0.09881,\ f'(x^{(3)}) = -0.0179078,\ f''(x^{(3)}) = 1.17126 \tag{11}$$

Using equation (2), we can write

$$x^{(4)} = x^{(3)} - \frac{f'(x^{(3)})}{f''(x^{(3)})} = 0.480409 \tag{12}$$

Convergence check: $|f'(x^{(4)})| = |-0.0005033| < \varepsilon$. Since the process has now converged, the optimum solution is taken as $x^* \approx x^{(4)} = 0.480409$.

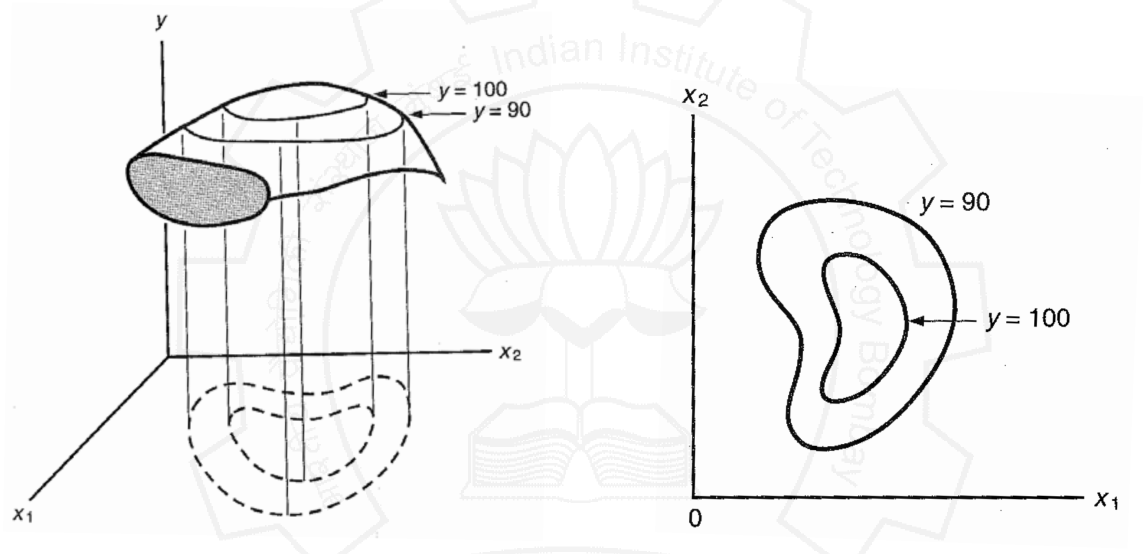
Multi-variable optimization methods

The multi-variable optimization methods deal with the optimization of multi-variable design. When an objective function is associated with two or more design or independent variables, the geometric representation becomes a response surface [*Figure 5.5.7*(*a*)]. *Figure 5.5.7*(*b*) depicts the contour line produced by the intersection of planes of constant y with the response surface and projected on the x_1x_2 plane [where y refers to the response surface function, and x_1 and x_2 are two independent variables]. Similar to the single variable optimization, the multi-variable optimization methods are also classified into two categories – *direct* and *gradient based search methods*.

Direct search method

Lattice search

In the lattice search, a two-dimensional grid lattice is superimposed over the contour as shown in *Figure 5.5.8*. In search of the location of the maximum point, the starting point is picked near to the center of the region i.e. at point 1. The objective function is evaluated for points 1 through 9. If the response at point 5 turns out to be the largest, it would be considered as the center point for the next search. The procedure continues until the location reached is one at which the central point is greater than any of the other eight neighboring points. Frequently, a coarse grid is used initially and a finer grid is used when the maximum location appears to be in approach.



(a) (b)

Figure 5.5.7 Representation of two variables for contour lines [1].

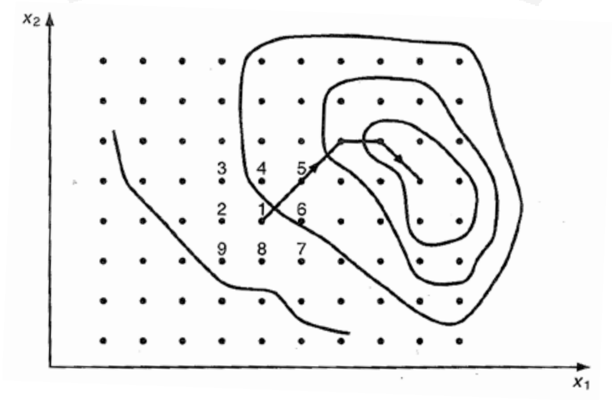


Figure 5.5.8 Example of the lattice search method [1].

Gradient based search method

Steepest descent (Cauchy's) method

This method is generally used to optimize a multi-variable design. The search direction used in this method is the negative of the gradient at any particular point $X^{(t)}$

$$S^{(t)} = -\nabla f(X^{(t)}) \tag{13}$$

Since this direction provides maximum descent in function values, it is called steepest descent method. At every iteration, the derivative is computed at the current point and a unidirectional search is performed in the negative to this derivative direction to find the minimum point along that direction. The minimum point becomes the current point and the search is continued from this point. The procedure continues until a point having a small enough gradient vector is found. The steps followed in the present method is mentioned sequentially below

1. Start with an arbitrary initial point $X^{(t)}$ and set the iteration number as $t = 1$.
2. Find the search direction $S^{(t)}$ as
$$S^{(t)} = -\nabla f^{(t)} = -\nabla f(X^{(t)}) \tag{14}$$
3. Determine the optimal step length $\lambda^{(t)}$ in the direction $S^{(t)}$ and set
$$X^{(t+1)} = X^{(t)} + [\lambda^{(t)} * S^{(t)}] = X^{(t)} - [\lambda^{(t)} * \nabla f^{(t)}] \tag{15}$$
4. Test the new point, $X^{(t+1)}$, for optimality. If $X^{(t+1)}$ is the optimum, stop the process. Alternately, go to step 5.
5. Set the new iteration number $t = t + 1$ and go to step 2.

Example

Minimize $f(x_1, x_2) = x_1 - x_2 + 2 * x_1^2 + 2 * x_1 * x_2 + x_2^2$ starting from the point $X^1 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$.

Solution

Iteration1

The gradient of 'f' is given by

$$\nabla f = \begin{Bmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \end{Bmatrix} = \begin{Bmatrix} 1 + 4 * x_1 + 2 * x_2 \\ -1 + 2 * x_1 + 2 * x_2 \end{Bmatrix}$$

$$\nabla f^{(1)} = \nabla f(X^{(1)}) = \begin{Bmatrix} 1 \\ -1 \end{Bmatrix} \tag{16, 17}$$

Therefore,

$$S^{(1)} = -\nabla f^{(1)} = \begin{Bmatrix} -1 \\ 1 \end{Bmatrix} \tag{18}$$

Next, we need to find the optimal step length $\lambda^{(1)}$ to find $X^{(2)}$. For this, we should minimize $f(X^{(1)} + \lambda^{(1)} S^{(1)}) = f(-\lambda^{(1)}, \lambda^{(1)}) = (\lambda^1)^2 - 2 * \lambda^{(1)}$ with respect to $\lambda^{(1)}$. Since $d(f)/d(\lambda^{(1)}) = 0$ at $\lambda^{(1)} = 1$, we would obtain

$$X^{(2)} = X^{(1)} + \lambda^{(1)} * S^{(1)} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} + 1 * \begin{Bmatrix} -1 \\ 1 \end{Bmatrix} = \begin{Bmatrix} -1 \\ 1 \end{Bmatrix} \tag{19}$$

Since $\nabla f^{(2)} = \nabla f(X^{(2)}) = \begin{Bmatrix} -1 \\ -1 \end{Bmatrix} \neq \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$, the $X^{(2)}$ is not optimum and hence, further iterations are required.

Iteration 2

$$S^{(2)} = -\nabla f^{(2)} = \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} \tag{20}$$

To minimize, we should first evaluate,

$$f(X^{(2)} + \lambda^{(2)} S^{(2)}) = f(-1 + \lambda^{(2)}, 1 + \lambda^{(2)}) = 5 * (\lambda^2)^2 - 2 * \lambda^{(2)} - 1 \tag{21}$$

Next, we set $d(f)/d(\lambda^{(2)}) = 0$ that leads to $\lambda^{(2)} = \frac{1}{5}$, and hence

$$X^{(3)} = X^{(2)} + \lambda^{(2)} * S^{(2)} = \begin{Bmatrix} -1 \\ 1 \end{Bmatrix} + \frac{1}{5} * \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} = \begin{Bmatrix} -0.8 \\ 1.2 \end{Bmatrix} \tag{22}$$

Since the components of the gradient at $X^{(3)}, \nabla f^{(3)} = \begin{Bmatrix} 0.2 \\ -0.2 \end{Bmatrix} \neq \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$, we proceed to the next iteration.

Iteration 3

$$S^{(3)} = -\nabla f^{(3)} = \begin{Bmatrix} -0.2 \\ 0.2 \end{Bmatrix} \tag{23}$$

Since,

$$\begin{aligned} f(X^{(3)} + \lambda^{(3)} S^{(3)}) &= f(-0.8 - \lambda^{(3)}, 1.2 + 0.2 * \lambda^{(3)}) \\ &= 0.04 * (\lambda^{3})^{2} - 0.08 * \lambda^{(3)} - 1.20 \end{aligned} \tag{24}$$

we set, $d(f)/d(\lambda^{(3)}) = 0$ that leads to $\lambda^{(3)} = 1.0$. Therefore

$$X^{(4)} = X^{(3)} + \lambda^{(3)} * S^{(3)} = \begin{Bmatrix} -0.8 \\ 1.2 \end{Bmatrix} + 1.0 * \begin{Bmatrix} -0.2 \\ 0.2 \end{Bmatrix} = \begin{Bmatrix} -1.0 \\ 1.4 \end{Bmatrix} \tag{25}$$

The gradient at $X^{(4)}$ is given by

$$\nabla f^{(4)} = \begin{Bmatrix} -0.2 \\ -0.2 \end{Bmatrix} \tag{26}$$

Since $\nabla f^{(4)} \neq \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$, $X^{(4)}$ is not the optimum and hence we have to proceed to the next iteration.

This process is continued until the optimum point, $X^{*} = \begin{Bmatrix} -1.0 \\ 1.5 \end{Bmatrix}$, is found.

Conjugate gradient (Fletcher-Reeves) method

The convergence characteristics of the steepest descent method can be improved greatly by modifying it into a conjugate method (which can be considered as a conjugate gradient method involving the use of the gradient of the function). Any minimization method that makes use of the conjugate directions is quadratically convergent. This property of quadratic convergence is very useful because it ensures that the method will minimize a quadratic function in *n* steps or less. Since any general function can be approximated reasonably well by a quadratic near the optimum point, any quadratically convergent method is expected to find the optimum point in a finite number of iterations. The iterative procedure of conjugate gradient is stated below

1. Start with an arbitrary initial point $X^{(t)}$.
2. Set the first search direction $S^{(1)} = -\nabla f(X^{(1)}) = -\nabla f^{(1)}$. (27)
3. Find the point $X^{(2)}$ using the relation

   $$X^{(2)} = X^{(1)} + \lambda^{(1)} * S^{(1)} \tag{28}$$

   where $\lambda^{(1)}$ is the optimal step length in the direction $S^{(1)}$. Set t = 2 and go to the next step.
4. Find $\nabla f^{(t)} = \nabla f(X^{(t)})$, and set (29)

   $$S^{(t)} = -\nabla f^{(t)} + \frac{[\nabla f^{(t)}]^T * [\nabla f^{(t)}]}{[\nabla f^{(t-1)}]^T * [\nabla f^{(t-1)}]} * S^{(t-1)} \tag{30}$$

5. Calculate the optimum step length $\lambda^{(t)}$ in the direction $S^{(t)}$, and find the new point

   $$X^{(t+1)} = X^{(t)} + \lambda^{(t)} * S^{(t)} \tag{31}$$

6. Test for the optimality of the point $X^{(t+1)}$. If $X^{(t+1)}$ is optimum, stop the process. Otherwise, set the value of $t = t + 1$ and go to step 4.

Example

Minimize $f(x_1, x_2) = x_1 - x_2 + 2 * x_1^2 + 2 * x_1 * x_2 + x_2^2$ starting from the point $X^{(1)} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$.

Solution

Iteration 1

The gradient of ‘f’ is given by

$$\nabla f = \begin{Bmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \end{Bmatrix} = \begin{Bmatrix} 1 + 4 * x_1 + 2 * x_2 \\ -1 + 2 * x_1 + 2 * x_2 \end{Bmatrix}$$

$$\nabla f^{(1)} = \nabla f(X^{(1)}) = \begin{Bmatrix} 1 \\ -1 \end{Bmatrix} \tag{32, 33}$$

The search direction is taken as $S^{(1)} = -\nabla f^{(1)} = \begin{Bmatrix} -1 \\ 1 \end{Bmatrix}$. (34)

To find the optimum step length $\lambda^{(1)}$ along $S^{(1)}$, we minimize $f(X^{(1)} + \lambda^{(1)} * S^{(1)})$ with respect to $\lambda^{(1)}$ as shown below

$$f(X^{(1)}+\lambda^{(1)}*S^{(1)}) = f(-\lambda^{(1)}, +\lambda^{(1)}) = (\lambda^{1})^2 - 2*\lambda^{(1)} \tag{35}$$

$$\frac{\partial f}{\partial \lambda^{(1)}} = 0 \text{ at } \lambda^{(1)} = 1 \tag{36}$$

Therefore,

$$X^{(2)} = X^{(1)} + \lambda^{(1)}*S^{(1)} = \begin{Bmatrix}0\\0\end{Bmatrix} + 1*\begin{Bmatrix}-1\\1\end{Bmatrix} = \begin{Bmatrix}-1\\1\end{Bmatrix} \tag{37}$$

Iteration 2

Since $\nabla f^{(2)} = \nabla f(X^{(2)}) = \begin{Bmatrix}-1\\-1\end{Bmatrix}$, equation (30) gives the next search direction as

$$S^{(2)} = -\nabla f^{(2)} + \frac{[\nabla f^{(2)}]^T * [\nabla f^{(2)}]}{[\nabla f^{(1)}]^T * [\nabla f^{(1)}]} * S^{(1)} \tag{38}$$

Where $[\nabla f^{(1)}]^T * [\nabla f^{(1)}] = 2$ and $[\nabla f^{(2)}]^T * [\nabla f^{(2)}] = 2$

Therefore,

$$S^{(2)} = -\begin{Bmatrix}-1\\-1\end{Bmatrix} + \left(\frac{2}{2}\right) * \begin{Bmatrix}-1\\1\end{Bmatrix} = \begin{Bmatrix}0\\+2\end{Bmatrix} \tag{39}$$

To find $\lambda^{(2)}$, we minimize

$$\begin{aligned} f(X^{(2)}+\lambda^{(2)}*S^{(2)}) &= f(-1, 1+2*\lambda^2) \\ &= -1-(1+2*\lambda^{(2)}) + 2 - 2*(1+2*\lambda^{(2)}) + (1+2*\lambda^{(2)})^2 \\ &= 4*(\lambda^2)^2 - 2*\lambda^2 - 1 \end{aligned} \tag{40}$$

With respect to $\lambda^{(2)}$. As $\frac{\partial f}{\partial \lambda^{(2)}} = 8*\lambda^{(2)} - 2 = 0$ at $\lambda^{(2)} = \frac{1}{4}$, we obtain

$$X^{(3)} = X^{(2)} + \lambda^{(2)}*S^{(2)} = \begin{Bmatrix}-1\\1\end{Bmatrix} + \frac{1}{4}*\begin{Bmatrix}0\\2\end{Bmatrix} = \begin{Bmatrix}-1\\1.5\end{Bmatrix} \tag{41}$$

Iteration 3

Now

$$\nabla f^{(3)} = \nabla f(X^{(3)}) = \begin{Bmatrix}0\\0\end{Bmatrix}, \ [\nabla f^{(2)}]^T * [\nabla f^{(2)}] = 2 \text{ and } [\nabla f^{(3)}]^T * [\nabla f^{(3)}] = 0$$

Thus

$$S^{(3)} = -\nabla f^{(3)} + \frac{[\nabla f^{(3)}]^T * [\nabla f^{(3)}]}{[\nabla f^{(2)}]^T * [\nabla f^{(2)}]} * S^{(2)} = -\begin{Bmatrix}0\\0\end{Bmatrix} + \left(\frac{0}{2}\right) * \begin{Bmatrix}0\\0\end{Bmatrix} = \begin{Bmatrix}0\\0\end{Bmatrix} \tag{42}$$

This shows that there is no search direction to reduce ‘f’ further, and hence $X^{(3)}$ is optimum.

Non-traditional optimization methods

The common difficulties associated with most classical direct and gradient-based techniques, are as follows.

- The convergence to an optimal solution depends on the chosen initial solution.
- Most algorithms tend to get stuck to a suboptimal solution.
- An algorithm efficient in solving one optimization problem may not be efficient in solving a different optimization problem.
- Algorithms are not efficient in handling problems having a discrete search space.
- Algorithms cannot be efficient used on a parallel machine.

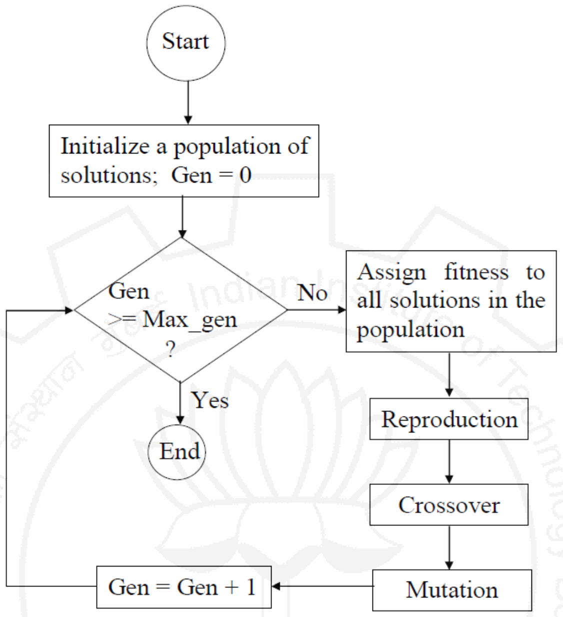
Since nonlinearities and complex interactions among problem variables often exist in real-world optimization problems, the search space usually contains more than one optimal solution, of which most are undesired locally optimal solutions having inferior objective function values. Non-traditional optimization methods like Genetic algorithms (GA) and simulated annealing are well suited for solving such problems, and in most cases they can find the global optimum solution with a high probability. We would concentrate only on GA optimization method in this lecture and the interested readers can undertake further studies in appropriate references to understand the simulated annealing optimization technique.

Genetic algorithm

Genetic algorithm is a population based probabilistic search and optimization technique, which works based on the mechanism of natural genetics and Darwin’s theory of natural selection. Several versions of the GA’s are available in the open literature, such as binary-coded GA, real-coded GA, micro-GA, messy-GA, and others. An attempt is made to explain the working principle of a binary-coded GA in this section.

Binary-coded GA

Figure 5.5.9 depicts the working principle of GA. It starts with a population of initial solutions generated at random. The fitness value of each solution in the population is calculated. The population of solutions is then modified using genetic operators, namely reproduction, crossover, and mutation. The role of each of these operators is discussed in detail in subsequent sections. After the reproduction, crossover and mutation are applied to the whole population of solutions, one generation of a GA is completed. Different criteria are used to terminate the program, such as the maximum number of generations, a desired accuracy in the solutions, and others.



Start
Initialize a population of solutions; Gen = 0
Gen >= Max_gen ?
No
Assign fitness to all solutions in the population
Yes
End
Reproduction
Crossover
Mutation
Gen = Gen + 1

Figure 5.5.9 A schematic diagram showing the working cycle of a GA.

Representation of objective function

The GA's are generally designed to solve maximization problems. For example

Maximization of f(x), $x_i^{(L)} \le x_i \le x_i^{(U)}$, i = 1, 2, ….n. (43)

where the terms x, L, U and n refer to variable, lower limit and upper limit of the variable, and the number of variables, respectively. GA is also used for minimization problem, by proper conversion of the maximization problem. Let us suppose that a function f(x) is to be minimized. It can be treated as a maximization problem as follows.

Either maximize – f(x), using duality principle,

or, maximize $1/f(x)$, for $f(x) \neq 0$,

or, maximize $1/[1+f(x)]$, for $f(x) \ge 0$, (44-46)

or, maximize $1/\{1+[f(x)]^2\}$, and so on, The steps utilized to solve the above mentioned maximization or minimization problems are given below.

Step 1 Generation of a population of solutions

Consider the equation (43) with two variable x_1 and x_2. An initial population of solutions of size N (say N =100, 200… depending on the complexity of the problem) is selected at random. The solutions are represented in the form of binary strings composed of 1's and 0's. A binary string can be compared to a biological chromosome and each bit of the string is nothing but a gene value. The length of a binary string is decided based on the desired accuracy in the values of the variables. For example, if we need an accuracy level of ε in the values of a variable x_1, we have to assign l bits to represent it, where l is determined using the expression given below.

$$l = \log_2 \frac{x_i^U - x_i^L}{\varepsilon} \qquad (47)$$

Let us assume that 10 bits are assigned to represent each variable. Thus, in this problem, the GA-string is only 20-bits (10 bits each for x_1 and x_2) long. *Table 5.5.2* depicts the initial population of GA-strings created at random.

Table 5.5.2 Initial population of GA-strings created at random [4]

1	0	0	.	.	.	1
0	1	1	.	.	.	0
1	1	1	.	.	.	1
0	0	1	.	.	.	1
.	.	.	.	.	.	.
.	.	.	.	.	.	.
1	0	1	.	.	.	1

Step 2 Fitness evaluation

To determine the fitness value of each solution (that is string), the real values of the variables x_1 and x_2 are to be calculated first. Knowing the minimum and maximum limits of a variable (say

x_1) and decoded value of the binary sub-string assigned to represent x_1, its real value is determined using the linear mapping rule given below

$$x_1 = x_1^L + \frac{x_1^U - x_1^L}{2^l - 1} * D \tag{48}$$

where l is the length of the sub-string used to represent the variable x_1 and D represents the decoded value of the binary sub-string. The decoded value of a binary number is calculated as follows: Let us consider a binary number 10110. Its decoded value will be determined as $(1*2^4)+(0*2^3)+(1*2^2)+(1*2^1)+(0*2^0)=22$. The real value of the second variable x_2 is determined following the same procedure. Knowing the real values of the variable- x_1 and x_2, the function value $f(x_1, x_2)$ is calculated, which is considered as the fitness value of the binary-string. It is noteworthy to mention that, in case of maximization problems the fitness value is equal to the function value, $f(x_1, x_2)$, while in the minimization problem the fitness is any of the equations (44-46). Hence to avoid the confusion in terminology the fitness is represented generally as $F(x_1, x_2)$.

Step 3 Reproduction

All the GA-strings contained in the population may not be equally good in terms of their fitness values calculated in step 2. In this step, reproduction operator is used to select the good ones from the population of strings based on their fitness information. Several reproduction schemes have been developed by various investigators. Some of them include proportionate selection (Roulette-wheel selection), ranking selection, tournament selection and elitism. For simplicity proportionate selection method is explained in the present section.

Proportionate selection (Roulette-wheel selection)

In this scheme, the probability of a string for being selected in the mating pool is considered to be proportional to its fitness. It is implemented with the help of a Roulette-wheel shown in *Figure 5.5.10*. The top surface area of the wheel is divided into N parts (where N is the population size), in proportion to the fitness values – F_1, F_2, …., F_N. The wheel is rotated in a particular direction (either clockwise or anti-clockwise) and a fixed pointer is used to indicate the winning area, after it stops. A particular sub-area representing a GA-solution is selected to be winner probabilistically and the probability that i–th area will be declared so, is given by the following expression

$$p = \frac{F_i}{\sum_{i=1}^{N} F_i} \tag{49}$$

The wheel is rotated for N times and each time, only one area is identified by the pointer to be the winner. The procedure is shown below in *Table 5.5.3*.

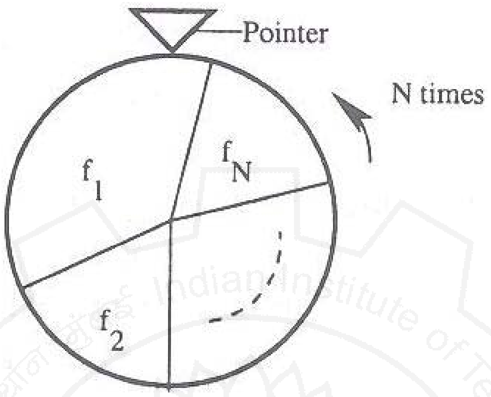


Figure 5.5.10 A Roulette-wheel used to implement proportionate selection scheme [4].

Table 5.5.3 Proportionate selection scheme [4]

GA-strings	Fitness	Probability of being selected
1 0 0 …. 1	F_1	$F_1 / \sum_{i=1}^{N} F_i$
0 1 1 …. 0	F_2	$F_2 / \sum_{i=1}^{N} F_i$
1 1 1 …. 0	F_3	$F_3 / \sum_{i=1}^{N} F_i$
0 0 1 …. 1	F_4	$F_4 / \sum_{i=1}^{N} F_i$
. . . ….. .	.	.
. . . …. .	.	.
. . . …. .	.	.
1 0 1 …. 1	F_N	$F_N / \sum_{i=1}^{N} F_i$

In this scheme, a good string may be selected for a number of times. Thus, the resulting mating pool (of size N) may look as in *Table 5.5.4*.

Table 5.5.4 Mating pool generated by proportionate selection scheme [4]

1	0	1	.	.	.	1
1	1	1	.	.	.	0
1	1	1	.	.	.	1
.	.	.	.	.	.	.
.	.	.	.	.	.	.
1	0	1	.	.	.	1

Step 4 Crossover

In crossover, there is an exchange of properties between two parents and as a result of which, two children solutions (off-springs) are produced. To carry out this operation, the parents or mating pairs (each pair consists of two strings) are selected at random from the mating pool. Thus, N/2 mating pairs are formed from a population of strings of size N. The parents are checked, whether they will participate in crossover by tossing a coin, whose probability of appearing head is p_c. If the head appears, the parent will participate in crossover to produce two children. Otherwise, they will remain intact in the population. It is important to mention that p_c is generally kept near to 1.0, so that almost all the parents can participate in crossover. Once a particular mating pair is selected for crossover, the crossing site is decided using a random number generator generating an integer lying between 1 and L–1, where L is the length of the string. The number of individuals participating in crossover can be probabilistically determined. There exist a large number of crossover schemes in the GA-literature. Some of these schemes are explained below.

Single-point crossover

The crossover site lying between 1 and L-1 is selected at random. The left side of the crossover site is generally kept unaltered and swapping is done between the two sub-strings lying on right side of the crossover site. *Table 5.5.5* shows the parents participating in a single-point crossover and *Table 5.5.6* depicts the two children produced due to the crossover.

Table 5.5.5 Parent strings participating in a single-point crossover [4]

0	1	0	1	1	0	1	0	1	1	1	0	0	1	1	0	1	0	0	1
0	0	1	1	0	1	0	0	1	0	1	1	1	0	1	0	0	1	0	1

Table 5.5.6 The children strings produced due to single-point crossover of the parent strings [4]

0	1	0	1	1	0	1	0	1	1	1	1	1	0	1	0	0	1	0	1
0	0	1	1	0	1	0	0	1	0	1	0	0	1	1	0	1	0	0	1

Two-point crossover

We select two different crossover sites lying between 1 and L-1, at random. *Table 5.5.7* depicts the parent strings' participating in this crossover and *Table 5.5.8* shows the two children strings produced due to the crossover.

Table 5.5.7 Parent strings participating in a two-point crossover [4]

1	0	1	0	1	1	1	0	0	1	1	0	1	0	0	1	0	1	0	0
0	1	0	0	1	0	1	1	1	0	1	0	0	0	1	1	0	0	1	1

Table 5.5.8 The children strings produced due to two-point crossover of the parent strings [4]

1	0	1	0	1	1	1	1	1	0	1	0	1	0	0	1	0	1	0	0
0	1	0	0	1	0	1	0	0	1	1	0	0	0	1	1	0	0	1	1

Step 5 Mutation

The children strings obtained from crossover are placed in the new population and the process is continued. Finally, the mutation operator is applied to the strings with a specified mutation probability (p_m). A mutation is the occasional random alteration of a binary digit. Thus in mutation a '0' is changed to '1' and vice-versa, at a random location. When used sparingly with the reproduction and crossover operators, mutation serves as a safeguard against a premature loss of important genetic material at a particular position.

Example

The objective is to minimize the function

$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$ in the interval $0 \le x_1, x_2 \le 6$.

Solution

Step 1

10-bits binary code is chosen for each variable i.e. x_1 and x_2. Hence the total string length equal to 20. The solution accuracy with 10 bit string is $(6\text{-}0)/(2^{10}-1)$ or 0.006 in the interval (0, 6). The population is chosen to be of 20 points. The random population created using Knuth's random number generator with a random seed equal to 0.760 is shown in *Table 5.5.9* [3]. The maximum number of generations are chosen as t_{max} = 30 and initialize the generator counter t=0.

Table 5.5.9 Evaluation and reproduction phases on a population [3]

	String		x_1	x_2	f(x)	F(x)	A	B	C	D	E	F	Mating pool	
	Substring 2	Substring 1											Substring 2	Substring 1
1	1110010000	1100100000	5.349	4.692	959.68	0.001	0.13	0.007	0.007	0.472	10	0	0010100100	1010101010
2	0001001101	0011100111	0.452	1.355	105.52	0.009	1.10	0.055	0.062	0.108	3	1	1010100001	0111001000
3	1010100001	0111001000	3.947	2.674	126.69	0.008	0.98	0.049	0.111	0.045	2	1	0001001101	0011100111
4	1001000110	1000010100	3.413	3.120	65.03	0.015	1.85	0.93	0.204	0.723	14	2	1110011011	0111000010
5	1100011000	1011100011	4.645	4.334	512.20	0.002	0.25	0.013	0.217	0.536	10	0	0010100100	1010101010
6	0011100101	0011111000	1.343	1.455	70.87	0.014	1.71	0.086	0303	0.931	19	2	0011100010	1011000011
7	0101011011	0000000111	2.035	0.041	88.27	0.011	1.34	0.067	0.370	0.972	19	1	0011100010	1011000011
8	1110101000	1110101011	5.490	5.507	1436.6	0.001	0.12	0.006	0.376	0.817	17	0	0111000010	1011000110
9	1001111101	1011100111	3.736	4.358	265.56	0.004	0.49	0.025	0.401	0.363	7	1	0101011011	0000000111
10	0010100100	1010101010	0.962	4.000	39.85	0.024	2.96	0.148	0.549	0.189	4	3	1001000110	1000010100
11	1111101001	0001110100	5.871	0.680	814.12	0.001	0.14	0.007	0.556	0.220	6	0	0011100101	0011111000
12	0000111101	0110011101	0.358	2.422	42.60	0.023	2.84	0.142	0.698	0.288	6	3	0011100101	0011111000
13	0000111110	1110001101	0.364	5.331	318.75	0.003	0.36	0.018	0.716	0.615	12	1	0000111101	0110011101
14	1110011011	0111000010	5.413	2.639	624.16	0.002	0.24	0.012	0.728	0.712	13	1	0000111110	1110001101
15	1010111010	1010111000	4.094	4.082	286.80	0.003	0.37	0.019	0.747	0.607	12	0	0000111101	0110011101
16	0100011111	1100111000	1.683	4.833	197.56	0.005	0.61	0.030	0.777	0.192	4	0	1001000110	1000010100

17	0111000010	1011000110	2.639	4.164	97.70	0.010	1.22	0.060	0.837	0.386	9	1	1001111101	1011100111
18	1010010100	0100001001	3.871	1.554	113.20	0.009	1.09	0.054	0.891	0.872	18	1	1010010100	0100001001
19	0011100010	1011000011	1.326	4.147	57.75	0.017	2.08	0.103	0.994	0.589	12	2	0000111101	0110011101
20	1011100011	1111010000	4.324	5.724	987.96	0.001	0.13	0.006	1.000	0.413	10	0	0010100100	1010101010

A: Expected count; C: Cumulative probability of selection; E: String number

B: Probability of selection D: Random number between 0 and 1; F: True count in the mating pool

Step 2

The next step is to calculate the fitness of each string in the population. The first substring (1100100000) decodes a value equal to ($2^9+2^8+2^5$) or 800. The corresponding parameter value equal to 0+(6-0)*800/1023 or 4.692 [using equation (48)]. Similarly the second substring decodes to a value equal to ($2^9+2^8+2^7+2^4$) or 912. The corresponding parameter value equal to 0+(6-0) *912/1023 or 5.349. Thus, the first string corresponds to the point $x^{(1)}=(4.692, 5.349)^T$. These values are substituted in the objective function expression to obtain the function value. It is found that the function value at this point is equal to $f(x^{(1)})=959.68$. We now calculate the fitness function value at this point using the transformation rule: $F(x^{(1)}) = 1.0 / (1.0+959.68) = 0.001$. This value is used in the reproduction operation. Similarly, other strings in the population are evaluated and fitness values are calculated. *Table 5.5.9* shows the objective function value and fitness value for all 20 strings in the initial population.

Step 3

Since $t = 0 < t_{max} = 30$, proceed to step 4.

Step 4

In this step, good strings in the population are selected to form the mating pool. Roulette-wheel selection procedure is used for this purpose. First calculate the average fitness ($\overline{F}$) of the population. By adding the fitness values of all strings and dividing the sum by the population size, we obtain $\overline{F}=0.008$. The next step is to compute the expected count of each string as $F(x)/\overline{F}$. The values are calculated and shown in column A of *Table 5.5.9*. Next, calculate the probability [using equation (49)] of each string being copied in the mating pool (column B). Once these probabilities are calculated, the cumulative probability is also calculated. These distributions are also shown in column C of *Table 5.5.9*. In order to form the mating pool, random numbers between zero and one (given in column D) are created. Further, the particular string which is specified by each of these random numbers is identified. For example, if the random number 0.472 is created, the tenth string gets a copy in the mating pool, because that string occupies the interval (0.401, 0.549), as shown in column C. Column E refers to the selected string. Similarly, other strings are selected according to the random number shown in column D. After this selection procedure is repeated N times (N is the population size), the number of selected copies for each string is counted. This number is shown in column F. *Table 5.5.9* also shows the complete mating pool. Figure 5.5.10 shows the initial random population and the mating pool after reproduction. The points marked with an enclosed box are the points in the mating pool. The inferior points are probabilistically eliminated from further consideration.

Step 5

In this step, the strings in the mating pool are used in the cross over operation. Single-point cross over is used for this purpose. Since the mating pool contains strings at random, we pick pairs of strings from the top of the list. Thus, strings 3 and 10 participate in the first cross over operation. When two strings are chosen for crossover, first a coin is flipped with a probability p_c=0.8 to check whether a crossover is desired or not. If the outcome of the coin-flipping is true, the crossover is performed, otherwise the strings are directly placed in an intermediate population for subsequent genetic operation. It turns out that the outcome of the first coin-flipping is true, meaning that the crossover is required to be performed. The next step is to find a cross site at random. We choose a site by creating a random number between (0, L-1) or (0, 19). It turns out that the obtained random number is 11. Thus, we cross the strings at the site 11 and create two new strings. After crossover, the children strings are placed in the intermediate population. Strings 14 and 2 (selected at random) are used in the crossover operation. This time the coin flipping comes true again and the crossover operation is performed at the site 8 found at random. The new children strings are put into the intermediate population. Figure 5.5.11 shows how points crossover and form new points. The points marked with a small box are the points in the mating pool and the points marked with a small circle are children points created after crossover operation. With the flipping of a coin with a probability p_c = 0.8, it turns out that fourth, seventh and tenth crossovers come out to be false. Thus, in these cases, the strings are copied directly into the intermediate population. The complete population at the end of the crossover operation is shown in *Table 5.5.10*.

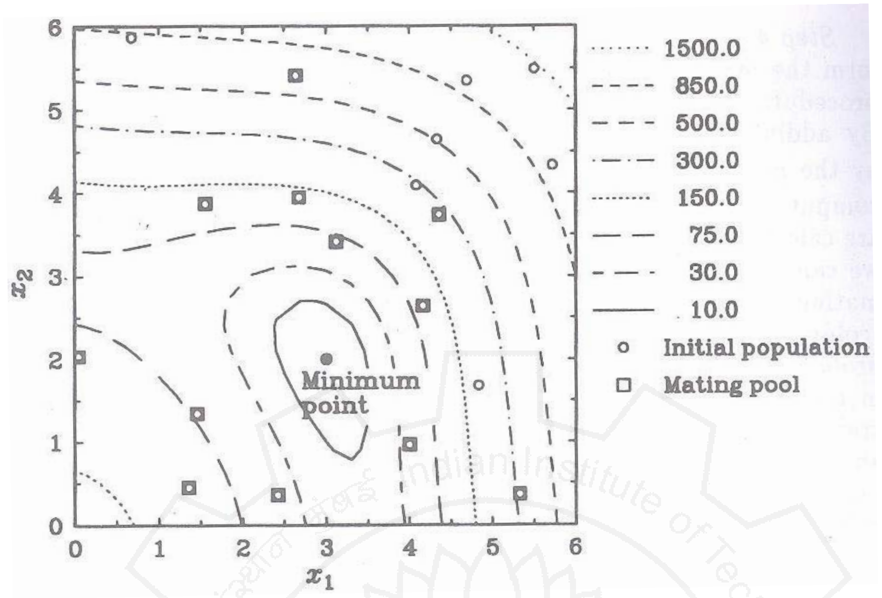


Figure 5.5.10 The initial population (marked with empty circles) and the mating pool (marked with boxes) on a contour plot of the objective function. The best point in the population has a function value 39.85 and the average function value of the initial population is 360.54 [3].

Table 5.5.10 Crossover and mutation operators [3]

Mating pool		G	H	Intermediate population		Mutation		x_1	x_2	f(x)	F(x)
Substring-2	Substring-1			Substring-2	Substring-1	Substring-2	Substring-1				
0010100100	1010101010	Y	9	0010100101	0111001000	001010**1**101	0111001000	1.015	2.674	18.89	0.050
1010100001	0111001000	Y	9	1010100000	1010101010	101010000**1**	1010101010	3.947	4.000	238.32	0.004
0001001101	0011100111	Y	12	0001001101	0011000010	0001001101	00**0**1000010	0.452	0.387	149.20	0.007
1110011011	0111000010	Y	12	1110011011	0111100111	1110011011	01**0**1100**0**11	5.413	2.082	596.34	0.002
0010100100	1010101010	Y	5	0010100010	1011000011	0010100010	1011000011	0.950	4.147	54.851	0.018
0011100010	1011000011	Y	5	0011100100	1010101010	0011100100	1**1**10101010	1.337	5.501	424.58	0.002
0011100010	1011000011	N		0011100010	1011000011	001110001**1**	1011**1**00011	1.331	4.334	83.93	0.012
0111000010	1011000110	N		0111000010	1011000110	01**0**10**1**0010	1011000110	1.982	4.164	70.47	0.014
0101011011	0000000111	Y	14	0101011011	0000010100	0101011011	0000010100	2.035	0.117	87.63	0.011
1001000110	1000010100	Y	14	1001000110	1000000111	10010**1**0110	1000000111	3.507	3.044	72.79	0.014
0011100101	0011111000	Y	1	0011100101	0011111000	0011100101	0011111000	1.343	1.455	70.87	0.014
0011100101	0011111000	Y	1	0011100101	0011111000	0011100101	0011111000	1.343	1.455	70.87	0.014
0000111101	0110011101	N		0000111101	0110011101	00001**0**1101	011**1**01110**0**	0.264	2.792	25.78	0.037
0000111110	1110001101	N		0000111110	1110001101	0000111110	1110001101	0.364	5.331	318.75	0.003
0000111101	0110011101	Y	18	0000111101	0110011100	0000111101	0110011100	0.358	2.416	42.92	0.023

1001000110	1000010100	Y	18	1001000110	1000010101	1001000110	**0**000010101	3.413	0.123	80.127	0.012
1001111101	1011100111	Y	10	1001111101	0100001001	1001111101	0100001001	3.736	1.554	95.97	0.010
1010010100	0100001001	Y	10	1010010100	1011100111	1010010100	101**0**100111	3.871	3.982	219.43	0.005
0000111101	0110011101	N		0000111101	0110011101	0000111101	0110011101	0.358	2.422	42.60	0.023
0010100100	1010101010	N		0010100100	1010101010	0010100100	1010101010	0.962	4.000	39.85	0.024

G: Whether crossover (Y-yes, N-no), H: Crossing site

Figure 5.5.12 shows the effect of mutation on the intermediate population. The points marked with a small circle are points in the intermediate population. The points marked with a small box constitute the new population (obtained after reproduction, crossover, and mutation). It is interesting to note that if only one bit is mutated in a string, the point is moved along a particular variable only. Like the crossover operator, the mutation operator has created some points better and some points worse than the original points. This flexibility enables GA operators to explore the search space properly before converging to a region prematurely and essential to solve global optimization problems.

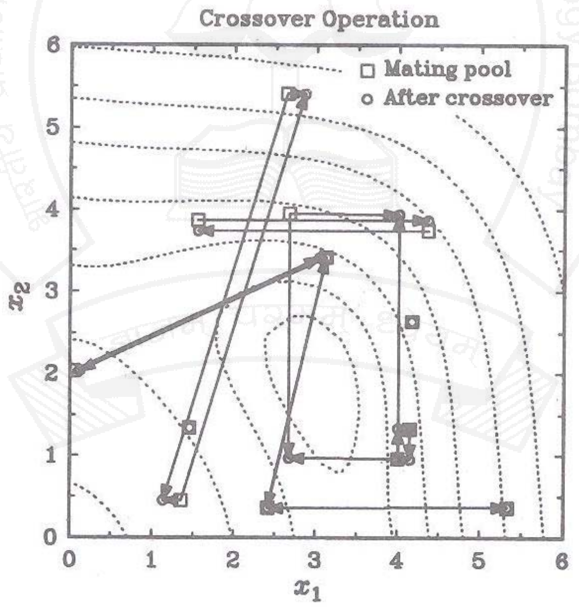


Figure 5.5.11 The population after the crossover operation. Two points are crossed over to form two new points. Of ten pairs of strings, seven pairs are crossed [3].

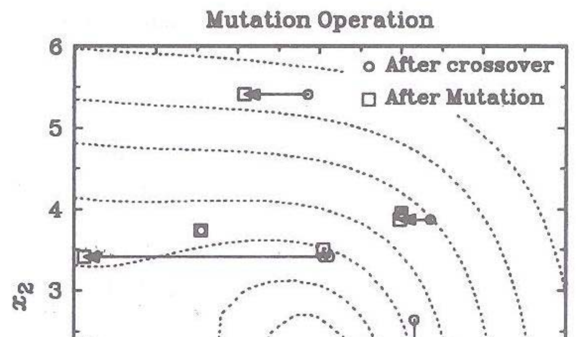


Figure 5.5.12 The population of mutation operation. Some point's don't get mutated and remain unaltered. The best point in the population has a function value 18.89 and the average function value of the population is 140.21, an improvement of over 60% [3].

Step 7

The resulting population becomes the new population. We now evaluate each string as before by first identifying the substrings for each variable and mapping the decoded values of the substrings in the chosen intervals. This completes one iteration of genetic algorithm. Next, increment the generation counter to t =1 and proceed to step 3 for the next iteration. The new population after one iteration of GA is shown in Fig. 5.5.12 (marked with empty boxes). It is observed that in one iteration, some good points are found. *Table 5.5.10* also shows the fitness values and objective function values of the new population members.

The average fitness of the new population is calculated as 0.015, a remarkable improvement from that in the initial population (the average in the initial population was 0.008). This process continues until the maximum allowable generation is reached or some other termination criterion is met. Figure 5.5.13 depicts that the population after 25 generation. At this generation the best point is found to be $(3.003, 1.994)^T$ with a function value 0.001. The fitness value at this point is equal to 0.999 and the average population fitness of the population is 0.474. Figure 5.5.13 also shows the points clustered around the true minimum of the function in this generation.

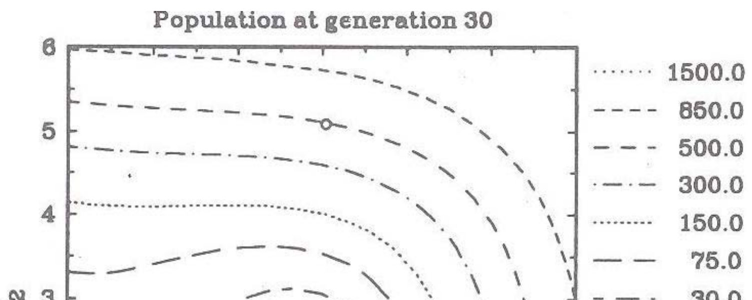


Figure 5.5.13 All 20 points in the population at generation 25 shown on the contour plot of the objective function. The figure shows that most points are clustered around the true minimum [3].

Exercise

1. State few non-traditional optimization methods.
2. Explain the methodology followed in the secant method to find out the minimum value of the variable in a single variable objective function.

Reference

1. G. Dieter, “Engineering design: A materials and processing approach”, 3rd edition, McGraw hill education, 2001.
2. S. S. Rao, “Engineering optimization”, 3rd edition, New age international (P) Ltd., 2004.
3. K. Deb, “Optimization for engineering design: Algorithms and examples”, PHI learning (P) Ltd., 2010.
4. D. K. Pratihar, “Soft computing”, Narosa publishing house, 2008.