Introduction To Genetic Algorithms

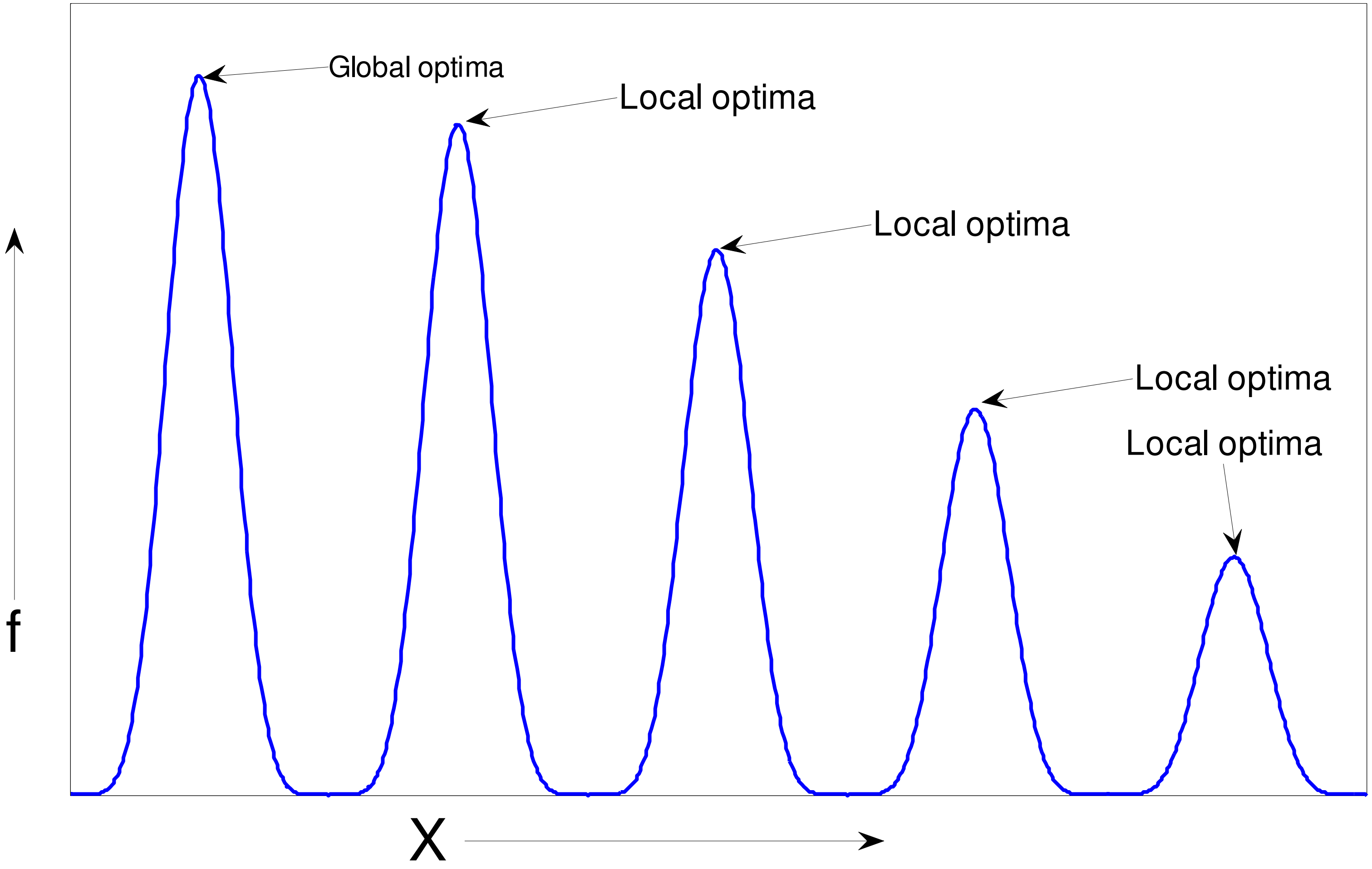
Dr. Rajib Kumar Bhattacharjya
Department of Civil Engineering
IIT Guwahati
Email: rkbc@iitg.ernet.in

7 November 2013

References

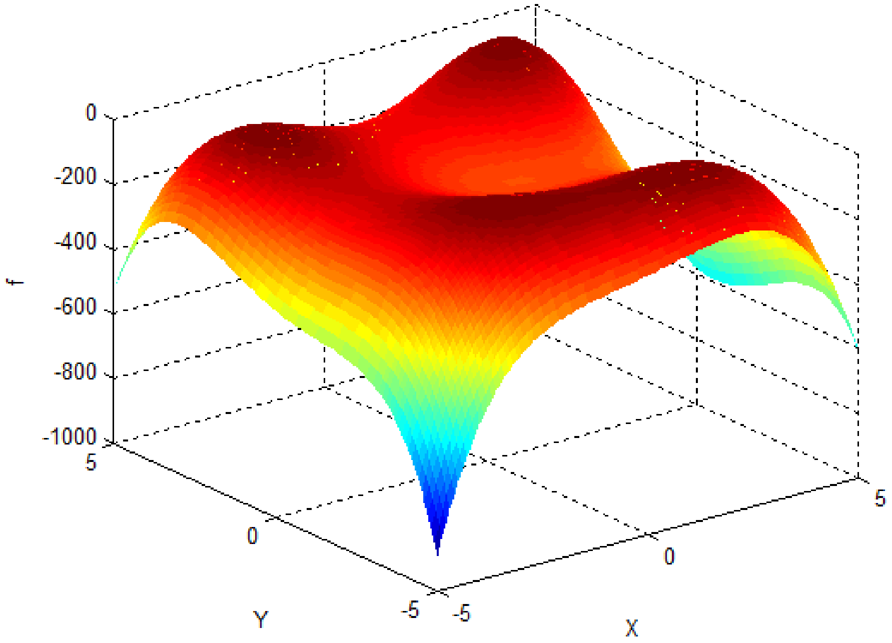
- D. E. Goldberg, ‘Genetic Algorithm In Search, Optimization And Machine Learning’, New York: Addison – Wesley (1989)
- John H. Holland ‘Genetic Algorithms’, Scientific American Journal, July 1992.
- Kalyanmoy Deb, ‘An Introduction To Genetic Algorithms’, Sadhana, Vol. 24 Parts 4 And 5.

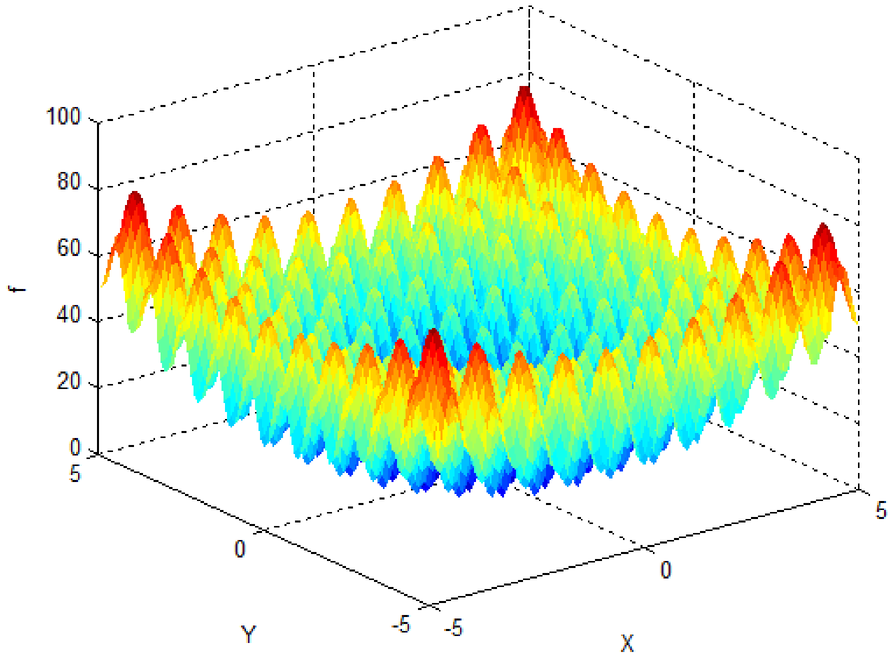
Introduction to optimization



Introduction to optimization

Multiple optimal solutions

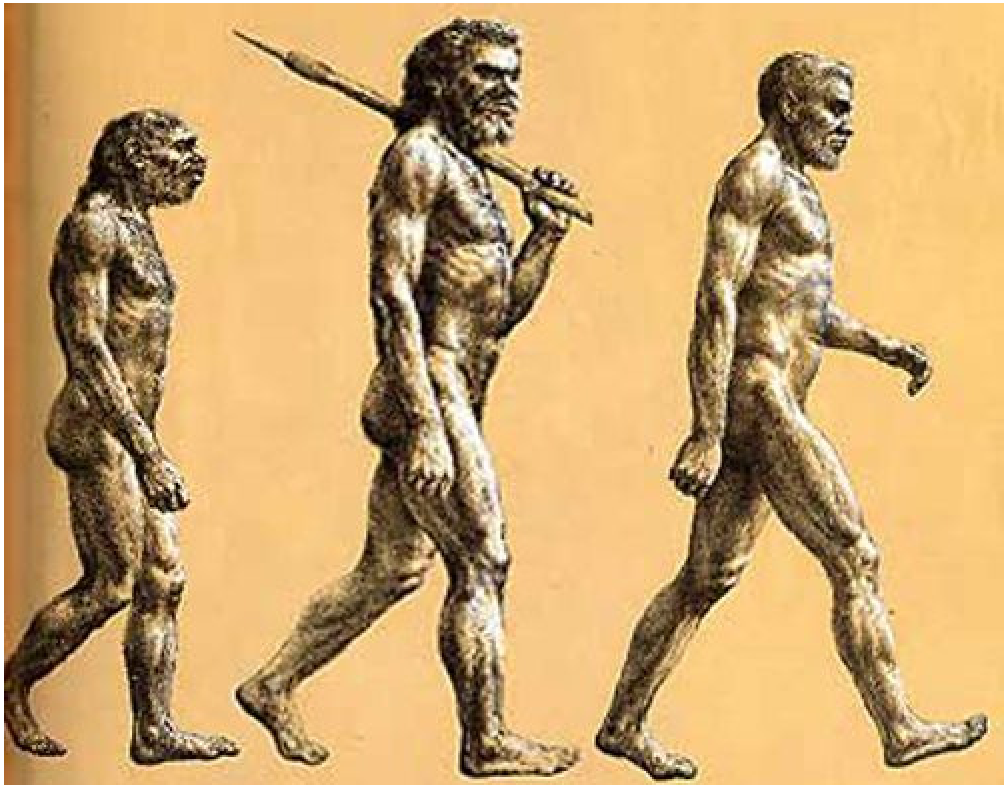




Genetic Algorithms

Genetic Algorithms are the heuristic search and optimization techniques that mimic the process of natural evolution.

Principle Of Natural Selection



"Select The Best, Discard The Rest"

An Example….

Giraffes have long necks

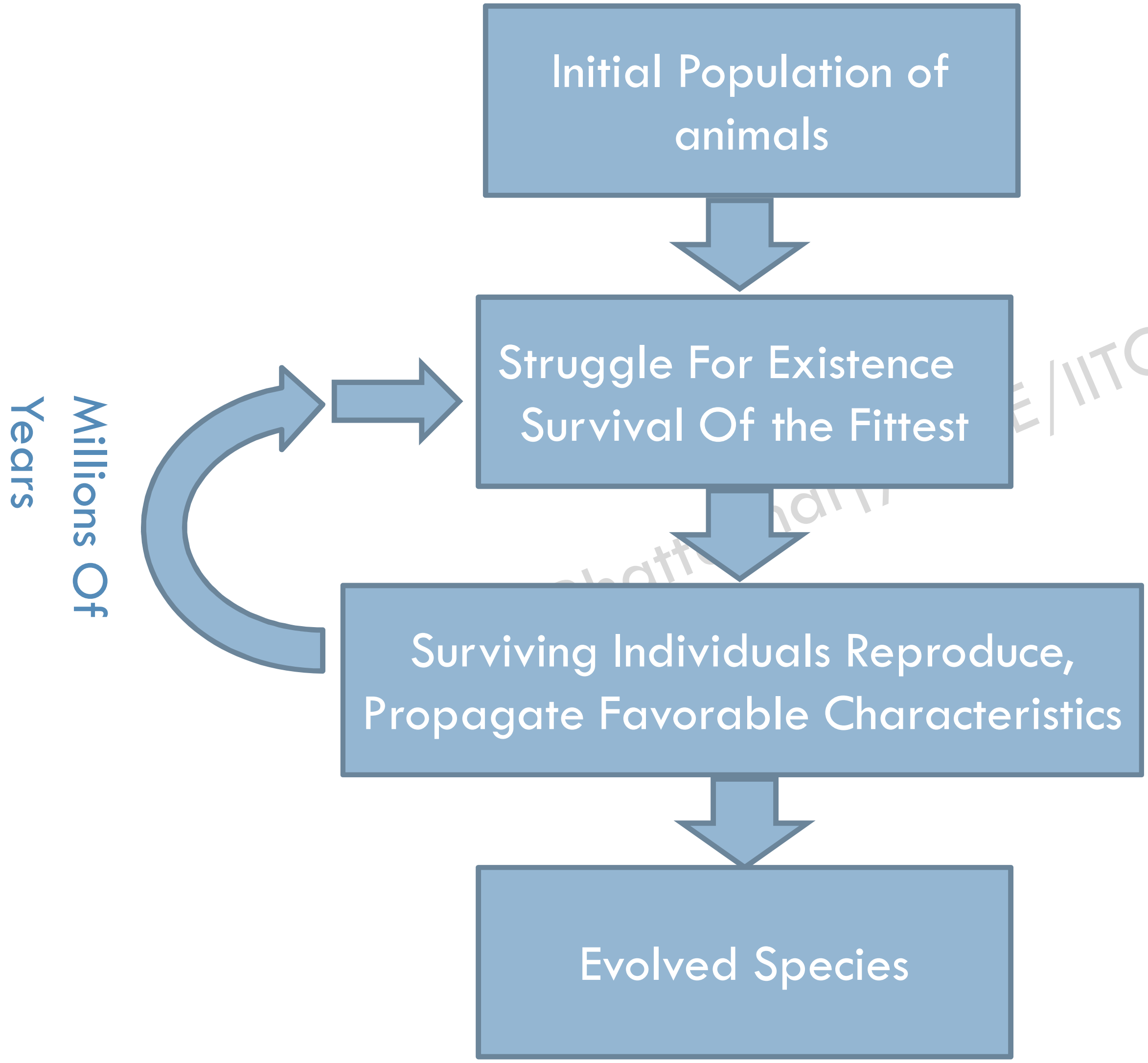
- Giraffes with slightly longer necks could feed on leaves of higher branches when all lower ones had been eaten off.
- They had a better chance of survival.
- Favorable characteristic propagated through generations of giraffes.

- Now, evolved species has long necks.

An Example….

This longer necks may have due to the effect of mutation initially. However as it was favorable, this was propagated over the generations.

Evolution of species



Initial Population of animals

Struggle For Existence Survival Of the Fittest

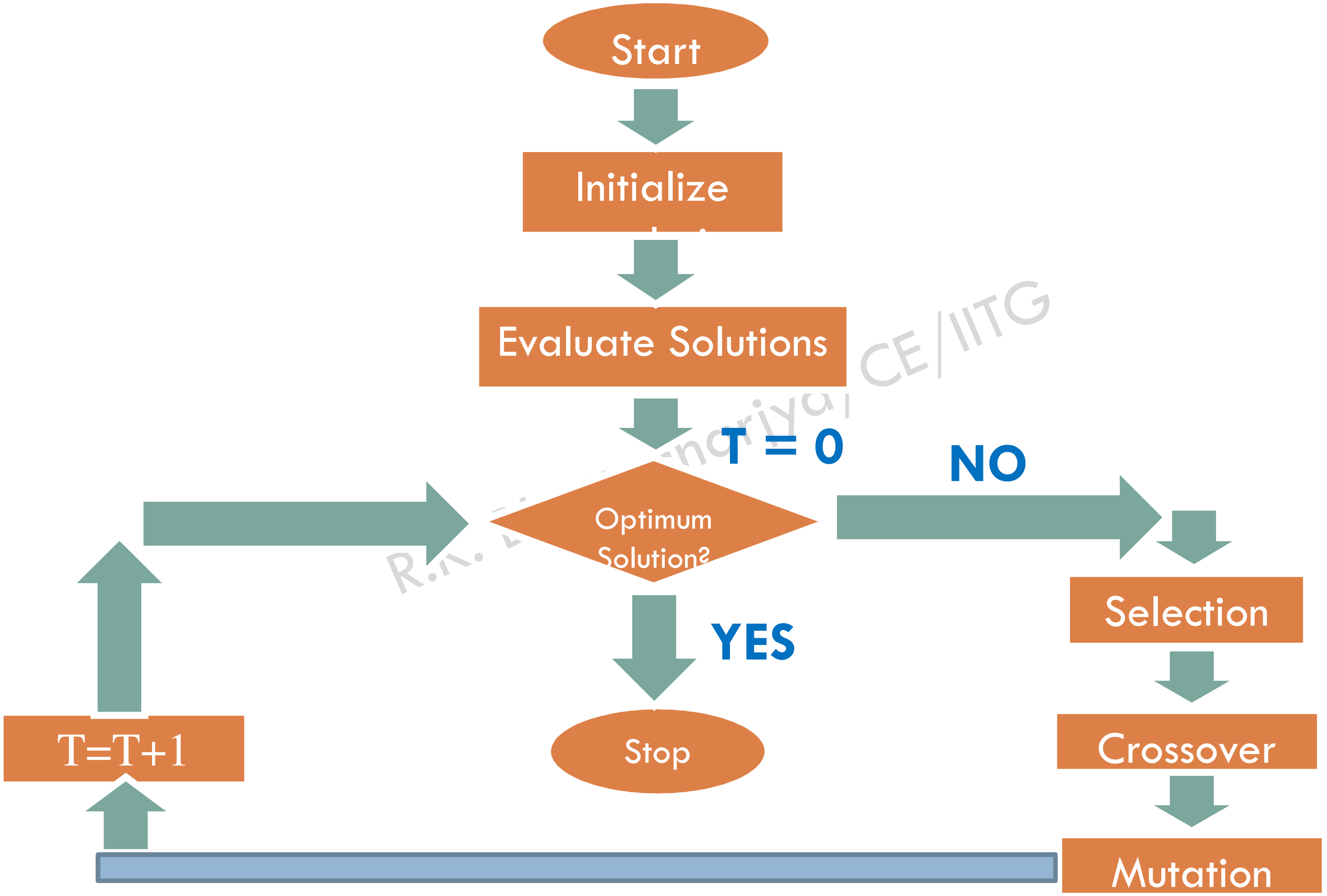
Surviving Individuals Reproduce, Propagate Favorable Characteristics

Evolved Species

Millions Of Years

Thus genetic algorithms implement the optimization strategies by simulating evolution of species through natural selection

Simple Genetic Algorithms



Start

Initialize

Evaluate Solutions

T = 0

Optimum Solution?

NO

Selection

Crossover

Mutation

YES

Stop

T=T+1

Simple Genetic Algorithm

```
function sga ()
        {
        Initialize population;
        Calculate fitness function;

        While(fitness value != termination criteria)
                {
                   Selection;

                   Crossover;

                   Mutation;

                Calculate fitness function;
                }
        }
```

GA Operators and Parameters

- Selection
- Crossover
- Mutation
- Now we will discuss about genetic operators

Selection

The process that determines which solutions are to be preserved and allowed to reproduce and which ones deserve to die out.

The primary objective of the selection operator is to emphasize the good solutions and eliminate the bad solutions in a population while keeping the population size constant.

"Selects the best, discards the rest"

Functions of Selection operator

Identify the good solutions in a population

Make multiple copies of the good solutions

Eliminate bad solutions from the population so that multiple copies of good solutions can be placed in the population

Now how to identify the good solutions?

Fitness function

A fitness value can be assigned to evaluate the solutions

A fitness function value quantifies the optimality of a solution. The value is used to rank a particular solution against all the other solutions

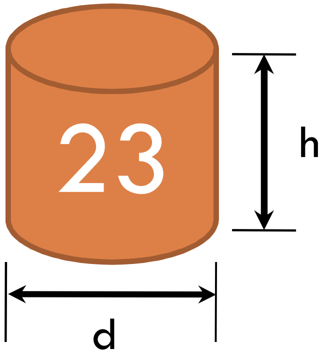
A fitness value is assigned to each solution depending on how close it is actually to the optimal solution of the problem

Assigning a fitness value

$$\begin{aligned} \text{Minimize} \quad & f(d,h) = c((\pi d^2/2) + \pi dh), \\ \text{Subject to} \quad & g_1(d,h) \equiv (\pi d^2 h/4) \geq 300, \\ \text{Variable bounds} \quad & d_{\min} \leq d \leq d_{\max}, \\ & h_{\min} \leq h \leq h_{\max}. \end{aligned}$$

Considering c = 0.0654

$$\begin{aligned} F(s) &= 0.0654(\pi(8)^2/2 + \pi(8)(10)), \\ &= 23, \end{aligned}$$



Selection operator

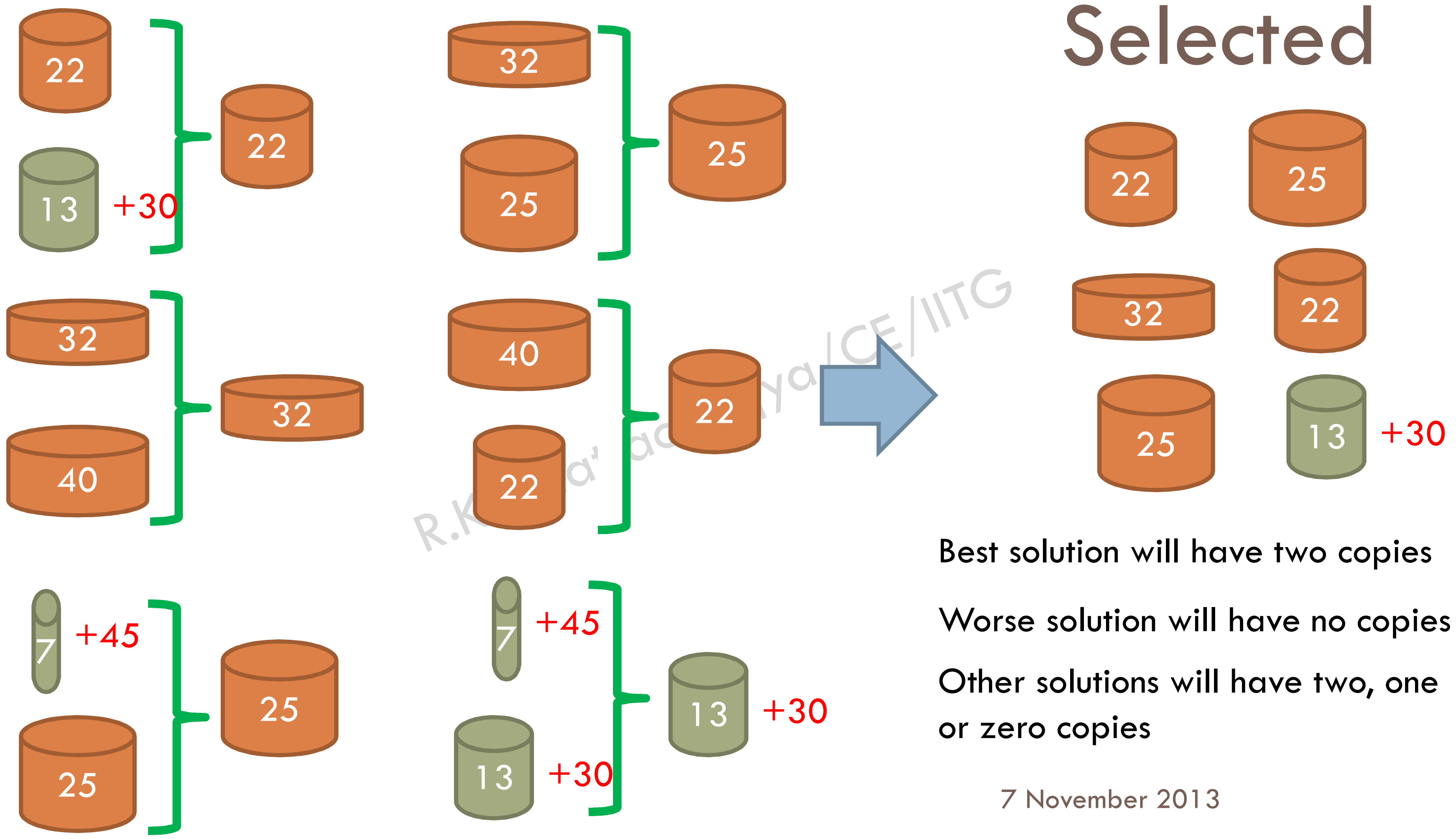
- There are different techniques to implement selection in Genetic Algorithms.

- They are:
  - Tournament selection
  - Roulette wheel selection
  - Proportionate selection
  - Rank selection
  - Steady state selection, *etc*

Tournament selection

- In tournament selection several tournaments are played among a few individuals. The individuals are chosen at random from the population.
- The winner of each tournament is selected for next generation.
- Selection pressure can be adjusted by changing the tournament size.
- Weak individuals have a smaller chance to be selected if tournament size is large.

Tournament selection



Best solution will have two copies

Worse solution will have no copies

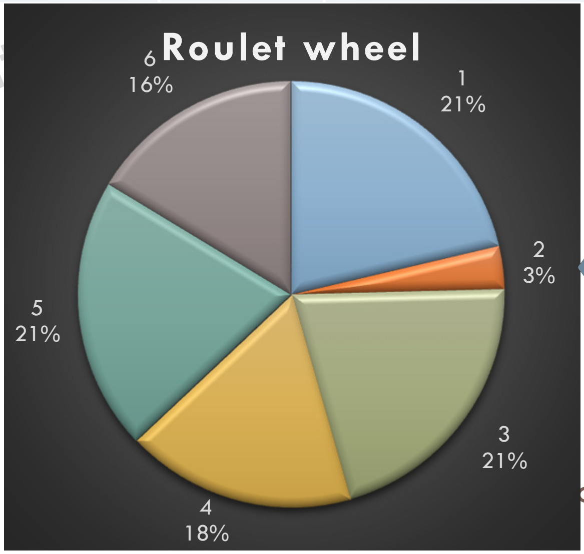
Other solutions will have two, one or zero copies

Roulette wheel and proportionate selection

Parents are selected according to their fitness values

The better chromosomes have more chances to be selected

Chrom #	Fitness	% of RW	EC	AC
1	50	26.88	1.61	2
2	6	3.47	0.19	0
3	36	20.81	1.16	1
4	30	17.34	0.97	1
5	36	20.81	1.16	1
6	28	16.18	0.90	1
	186	100.00	6	6



Rank selection

Chrom #	Fitness
1	37
2	6
3	36
4	30
5	36
6	28

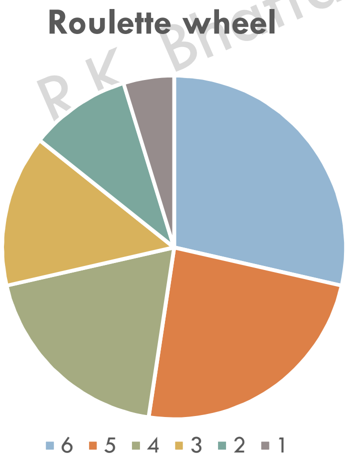
Sort according to fitness

Chrom #	Fitness
1	37
3	36
5	36
4	30
6	28
2	6

Assign raking

Chrom #	Rank
1	6
3	5
5	4
4	3
6	2
2	1

Chrom #	% of RW
1	29
3	24
5	19
4	14
6	10
2	5



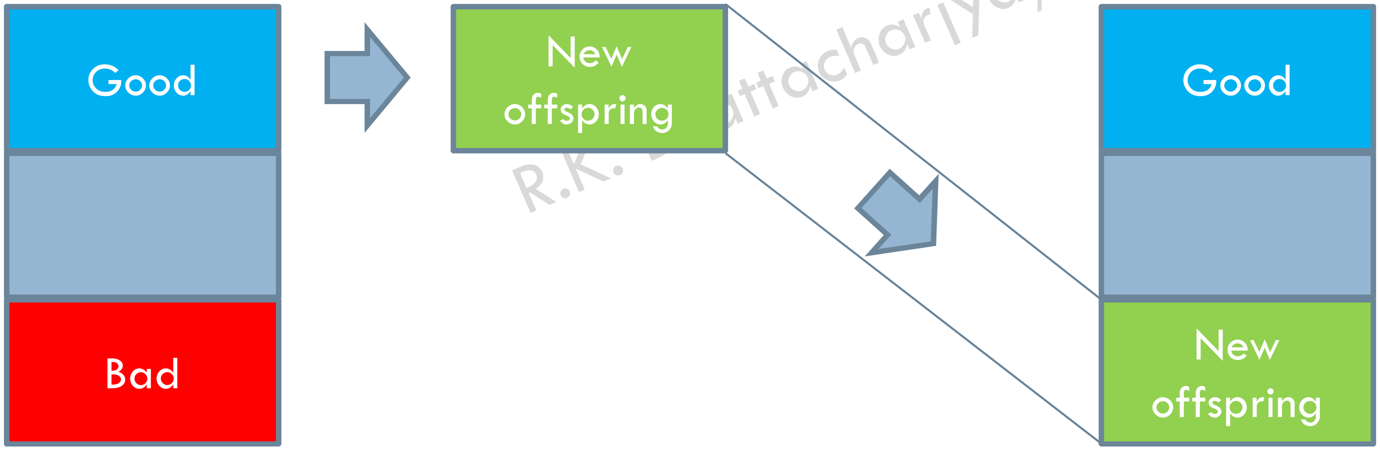
Chrom #	EC	AC
1	1.714	2
3	1.429	1
5	1.143	1
4	0.857	1
6	0.571	1
2	0.286	0

Steady state selection

In this method, a few good chromosomes are used for creating new offspring in every iteration.

Then some bad chromosomes are removed and the new offspring is placed in their places

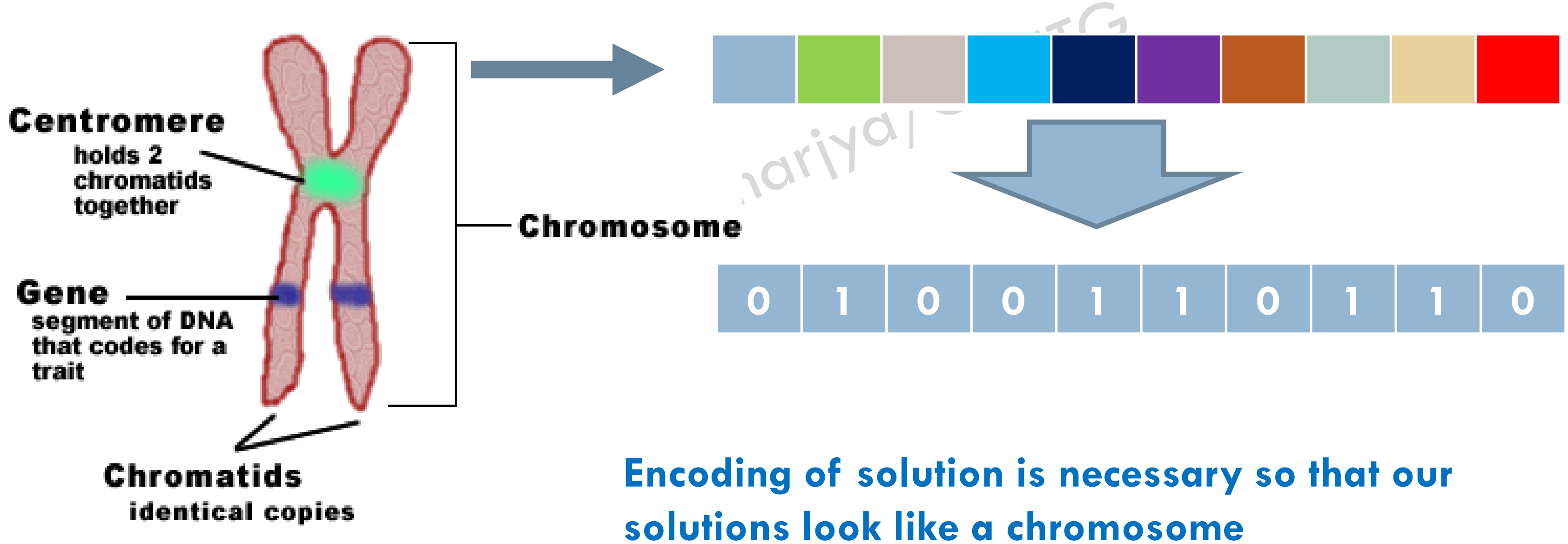
The rest of population migrates to the next generation without going through the selection process.



How to implement crossover

The crossover operator is used to create new solutions from the existing solutions available in the mating pool after applying selection operator.

This operator exchanges the gene information between the solutions in the mating pool.



0	1	0	0	1	1	0	1	1	0

Encoding of solution is necessary so that our solutions look like a chromosome

Source: http://www.biologycorner.com/bio1/celldivision-chromosomes.html

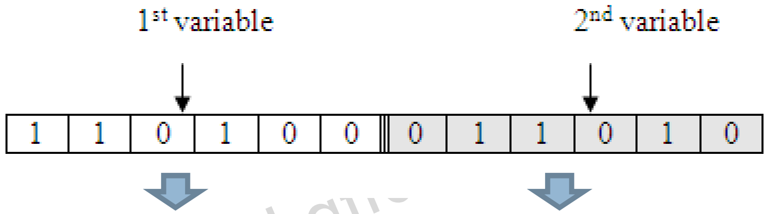
Encoding

The process of representing a solution in the form of a string that conveys the necessary information.

Just as in a chromosome, each gene controls a particular characteristic of the individual, similarly, each bit in the string represents a characteristic of the solution.

Encoding Methods

☐ Most common method of encoding is binary coded. Chromosomes are strings of 1 and 0 and each position in the chromosome represents a particular characteristic of the problem



Decoded value 52 26

Mapping between decimal and binary value

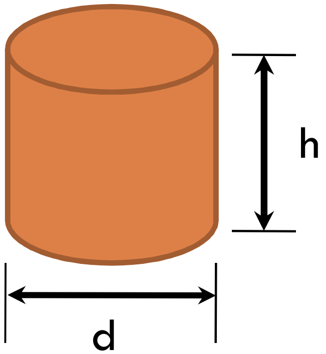
$$x_i = x_i^{\min} + \frac{x_i^{\max} - x_i^{\min}}{2^{l_i} - 1} \mathrm{DV}(s_i)$$

Encoding Methods

$$\begin{aligned} \text{Minimize} \quad & f(d,h) = c((\pi d^2/2) + \pi dh), \\ \text{Subject to} \quad & g_1(d,h) \equiv (\pi d^2 h/4) \geq 300, \\ \text{Variable bounds} \quad & d_{\min} \leq d \leq d_{\max}, \\ & h_{\min} \leq h \leq h_{\max}. \end{aligned}$$

Defining a string [0100001010]

d h



(d,h) = (8,10) cm

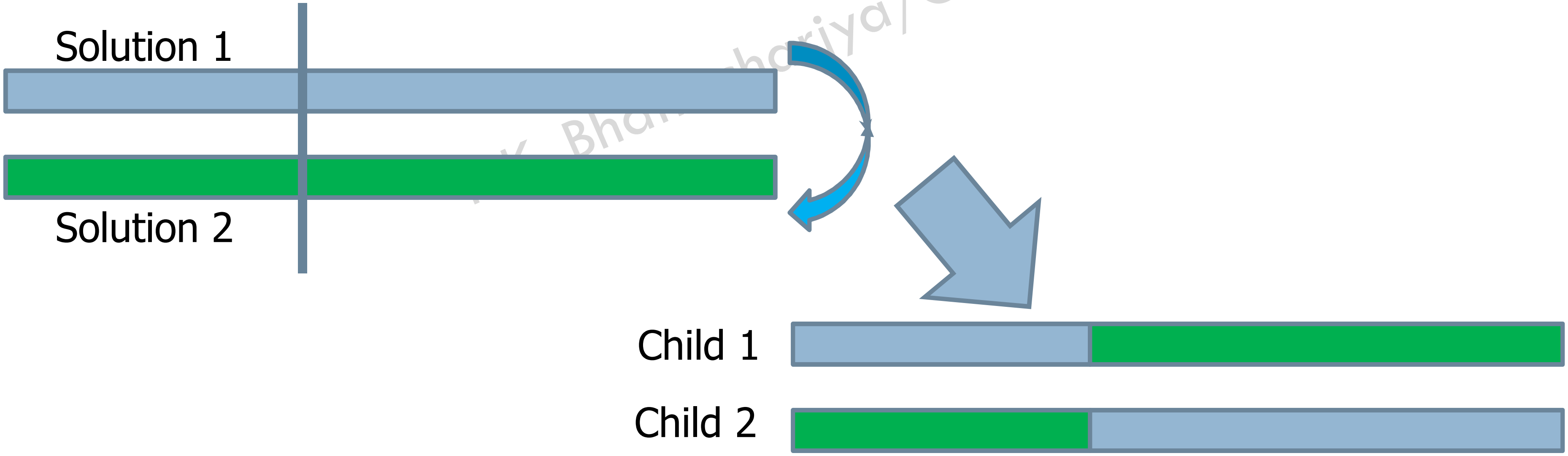
Chromosome = [0100001010]

Crossover operator

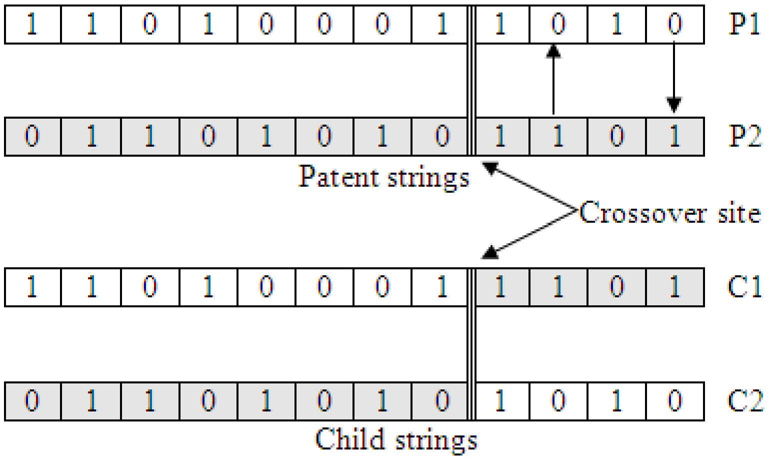
The most popular crossover selects any two solutions strings randomly from the mating pool and some portion of the strings is exchanged between the strings.

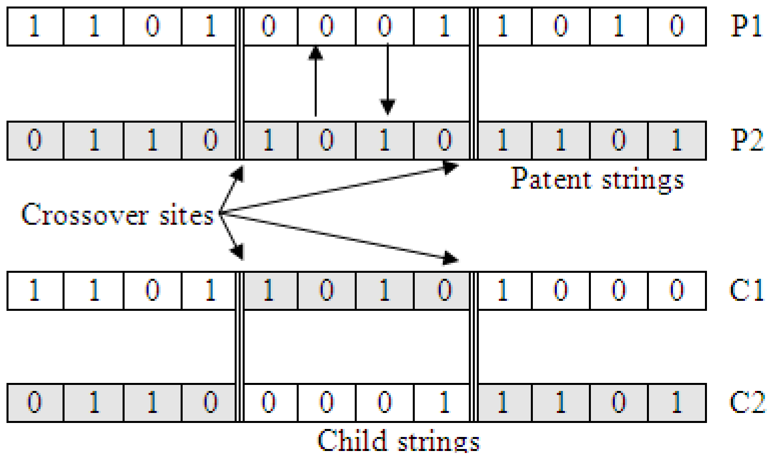
The selection point is selected randomly.

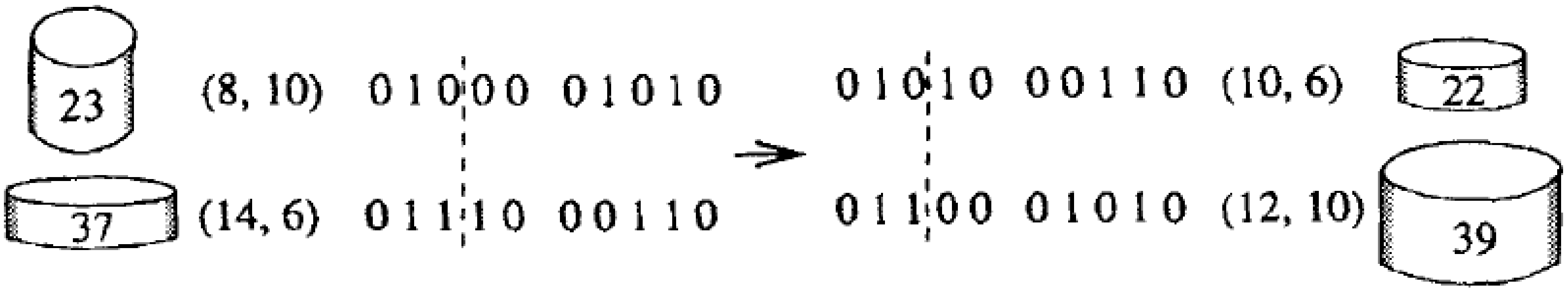
A probability of crossover is also introduced in order to give freedom to an individual solution string to determine whether the solution would go for crossover or not.



Binary Crossover







Source: Deb 1999

Mutation operator

Mutation is the occasional introduction of new features in to the solution strings of the population pool to maintain diversity in the population.

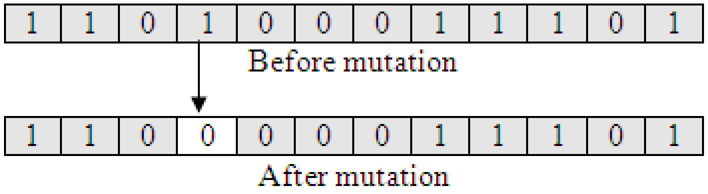
Though crossover has the main responsibility to search for the optimal solution, mutation is also used for this purpose.

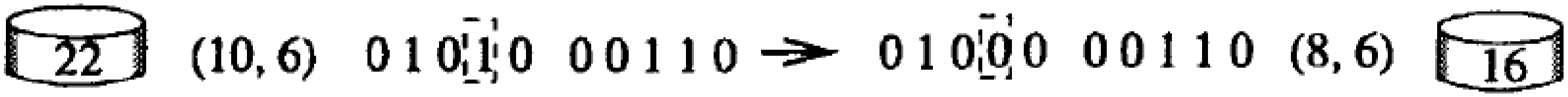
Before mutation

After mutation

Binary Mutation

- Mutation operator changes a 1 to 0 or vise versa, with a mutation probability of .
- The mutation probability is generally kept low for steady convergence.
- A high value of mutation probability would search here and there like a random search technique.





Source: Deb 1999

Elitism

- Crossover and mutation may destroy the best solution of the population pool
- Elitism is the preservation of few best solutions of the population pool
- Elitism is defined in percentage or in number

Nature to Computer Mapping

Nature	Computer
Population	Set of solutions
Individual	Solution to a problem
Fitness	Quality of a solution
Chromosome	Encoding for a solution
Gene	Part of the encoding solution

An example problem

$$Maximize\ f(x) = \sin(x)$$

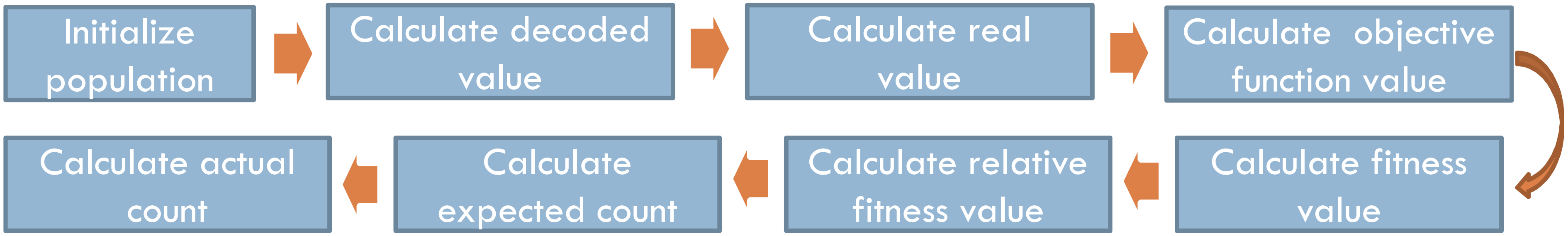
$$0 \leq x \leq \pi$$

Consider 6 bit string to represent the solution, then
000000 = 0 and 111111 = π

Assume population size of 4

Let us solve this problem by hand calculation

An example problem



Initialize population → Calculate decoded value → Calculate real value → Calculate objective function value → Calculate fitness value → Calculate relative fitness value → Calculate expected count → Calculate actual count

Initial population		Decoding		Fitness calculation		Selection: Proportionate selection		
Sol No	Binary String	DV	x value	f	F	Relative Fitness	Expected count	Actual count
1	100101	37	0.587	0.96	0.96	0.38	1.53	2
2	001100	12	0.19	0.56	0.56	0.22	0.89	1
3	111010	58	0.921	0.25	0.25	0.10	0.39	0
4	101110	46	0.73	0.75	0.75	0.30	1.19	1
				Avg	0.563			
				Max	0.96			

An example problem: Crossover

Matting pool → Random generation of crossover site → New population

Crossover: Single point

Sol No	Matting pool	CS	New Binary String	DV	x value	f	F
1	100101	3	100100	36	0.57	0.97	0.97
2	001100	3	001101	13	0.21	0.60	0.60
3	100101	2	101110	46	0.73	0.75	0.75
4	101110	2	100101	37	0.59	0.96	0.96
						Avg	0.8223
						Max	0.97

An example problem: Mutation

	Mutation					
Sol No	Population after crossover	Population after mutation	DV	x value	f	F
1	100100	100000	32	0.51	1.00	1.00
2	001101	101101	45	0.71	0.78	0.78
3	101110	100110	38	0.60	0.95	0.95
4	100101	101101	45	0.71	0.78	0.78
					Avg	0.878
					Max	1.00

Real coded Genetic Algorithms

- Disadvantage of binary coded GA
  - more computation
  - lower accuracy
  - longer computing time
  - solution space discontinuity
  - hamming cliff

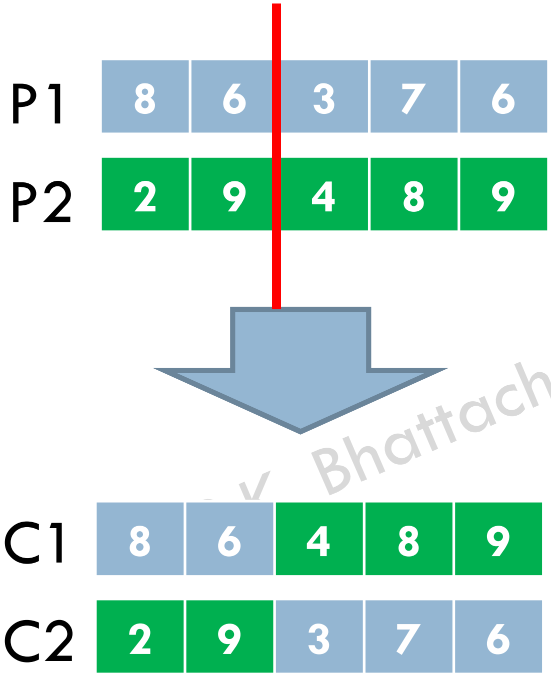
Real coded Genetic Algorithms

- The standard genetic algorithms has the following steps
  1. Choose initial population
  2. Assign a fitness function
  3. Perform elitism
  4. Perform selection
  5. Perform crossover
  6. Perform mutation

- In case of standard Genetic Algorithms, steps 5 and 6 require bitwise manipulation.

Real coded Genetic Algorithms

Simple crossover: similar to binary crossover



P1	8	6	3	7	6
P2	2	9	4	8	9
C1	8	6	4	8	9
C2	2	9	3	7	6

Real coded Genetic Algorithms

Linear Crossover

- Parents: (x_1,…,x_n) and (y_1,…,y_n)
- Select a single gene (*k*) at random
- Three children are created as,

$$(x_1, ..., x_k, 0.5 \cdot y_k + 0.5 \cdot x_k, ..., x_n)$$

$$(x_1, ..., x_k, 1.5 \cdot y_k - 0.5 \cdot x_k, ..., x_n)$$

$$(x_1, ..., x_k, -0.5 \cdot y_k + 1.5 \cdot x_k, ..., x_n)$$

- From the three children, best two are selected for the next generation

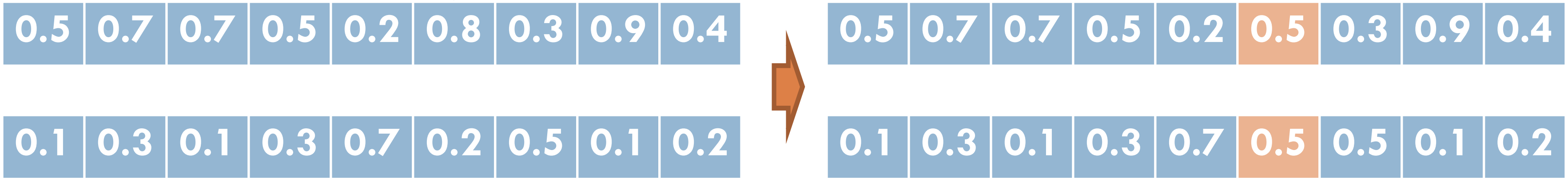
Real coded Genetic Algorithms

Single arithmetic crossover

- Parents: $(x_1,\ldots,x_n)$ and $(y_1,\ldots,y_n)$
- Select a single gene (*k*) at random
- child_1 is created as,

$$(x_1, \ldots, x_k, \alpha \cdot y_k + (1-\alpha) \cdot x_k, \ldots, x_n)$$

- reverse for other child. e.g. with $\alpha = 0.5$



0.5	0.7	0.7	0.5	0.2	0.8	0.3	0.9	0.4
0.1	0.3	0.1	0.3	0.7	0.2	0.5	0.1	0.2

0.5	0.7	0.7	0.5	0.2	0.5	0.3	0.9	0.4
0.1	0.3	0.1	0.3	0.7	0.5	0.5	0.1	0.2

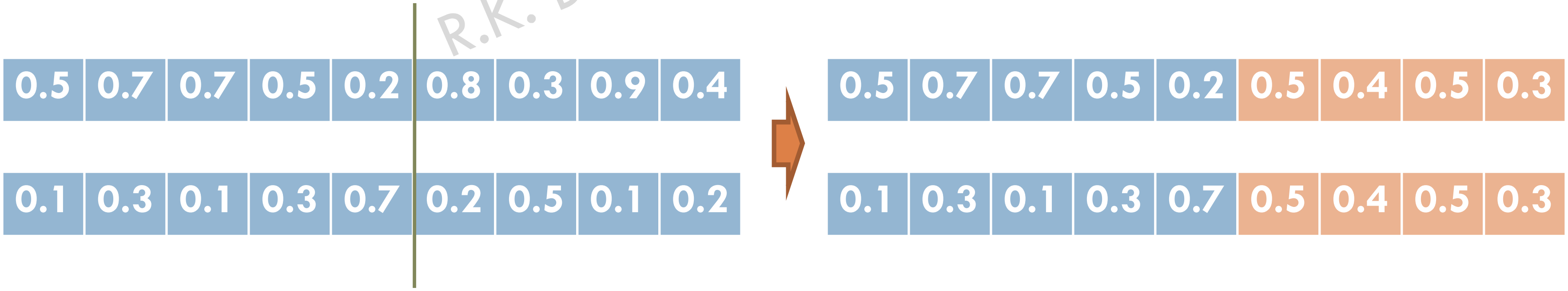
Real coded Genetic Algorithms

Simple arithmetic crossover

- Parents: $(x_1,\ldots,x_n)$ and $(y_1,\ldots,y_n)$
- Pick random gene *(k)* after this point mix values
- child_1 is created as:

$$(x_1, \ldots, x_k, \alpha \cdot y_{k+1} + (1-\alpha)\cdot x_{k+1}, \ldots, \alpha \cdot y_n + (1-\alpha)\cdot x_n)$$

- reverse for other child. e.g. with $\alpha = 0.5$



0.5	0.7	0.7	0.5	0.2	0.8	0.3	0.9	0.4
0.1	0.3	0.1	0.3	0.7	0.2	0.5	0.1	0.2

0.5	0.7	0.7	0.5	0.2	0.5	0.4	0.5	0.3
0.1	0.3	0.1	0.3	0.7	0.5	0.4	0.5	0.3

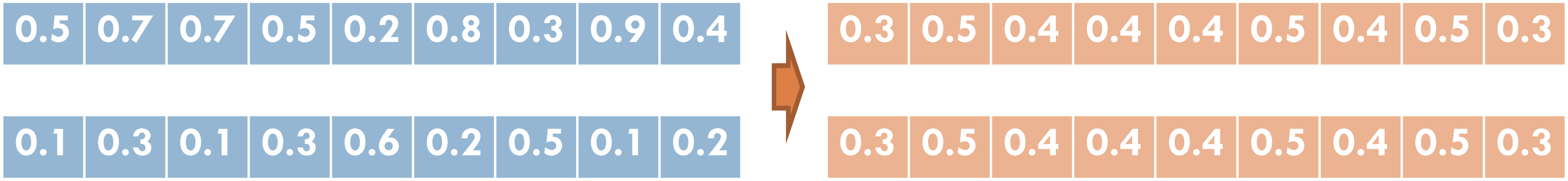
Real coded Genetic Algorithms

Whole arithmetic crossover

- Most commonly used
- Parents: $(x_1,\ldots,x_n)$ and $(y_1,\ldots,y_n)$
- child$_1$ is:

$$\alpha \cdot \overline{x} + (1-\alpha)\cdot \overline{y}$$

- reverse for other child. e.g. with $\alpha = 0.5$



0.5	0.7	0.7	0.5	0.2	0.8	0.3	0.9	0.4
0.1	0.3	0.1	0.3	0.6	0.2	0.5	0.1	0.2

0.3	0.5	0.4	0.4	0.4	0.5	0.4	0.5	0.3
0.3	0.5	0.4	0.4	0.4	0.5	0.4	0.5	0.3

Simulated binary crossover

- ☐ Developed by Deb and Agrawal, 1995)

$$x_i^{(1,t+1)} = 0.5\left[\left(1+\beta_{q_i}\right)x_i^{(1,t)} + \left(1-\beta_{q_i}\right)x_i^{(2,t)}\right]$$

$$x_i^{(2,t+1)} = 0.5\left[\left(1-\beta_{q_i}\right)x_i^{(1,t)} + \left(1+\beta_{q_i}\right)x_i^{(2,t)}\right]$$

$$\beta_{q_i} = \begin{cases} (2u_i)^{\frac{1}{n_c+1}} , & if\ u_i \le 0.5 \\ \left(\dfrac{1}{2(1-u_i)}\right)^{\frac{1}{n_c+1}} , & otherwise \end{cases}$$

Where, u_i a random number

n_c is a parameter that controls the crossover process. A high value of the parameter will create near-parent solution

Random mutation

$$y_i^{(1,t+1)} = u_i\left(x_i^u - x_i^l\right)$$

Where u_i is a random number between [0,1]

$$y_i^{(1,t+1)} = x_i^{1,t+1} + (u_i - 0.5)\Delta_i$$

Where, Δ_i is the user defined maximum perturbation

Normally distributed mutation

A simple and popular method

$$y_i^{(1,t+1)} = x_i^{1,t+1} + N(0,\sigma_i)$$

Where $N(0,\sigma_i)$ is the Gaussian probability distribution with zero mean

Polynomial mutation

Deb and Goyal,1996 proposed

$$y_i^{1,t+1} = x_i^{1,t+1} + \left(x_i^u - x_i^l\right)\delta_i$$

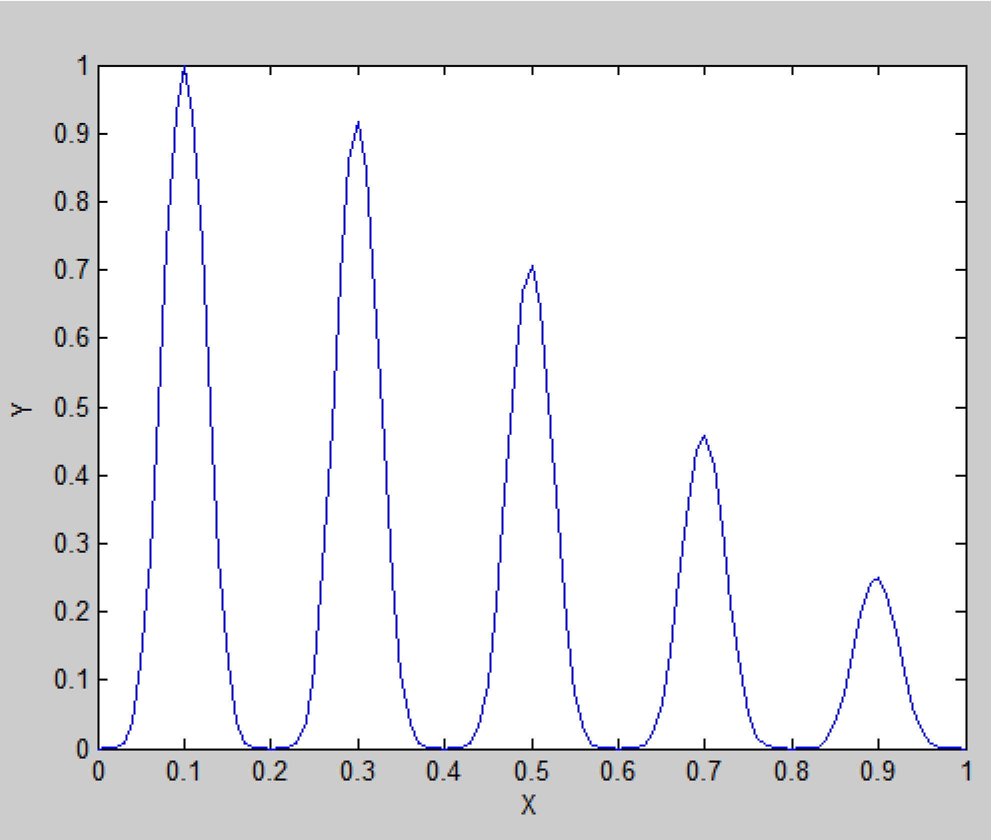
$$\delta_i = \begin{cases} (2u_i)^{1/(\eta_m+1)} - 1 & \text{If } u_i < 0.5 \\ 1 - \left(2(1-u_i)\right)^{1/(\eta_m+1)} & \text{If } u_i \geq 0.5 \end{cases}$$

Multi-modal optimization

$$Minimize\ f(x) = 2^{-2\left(\frac{(x-0.1)}{0.8}\right)^2} sin^6(5\pi x)$$

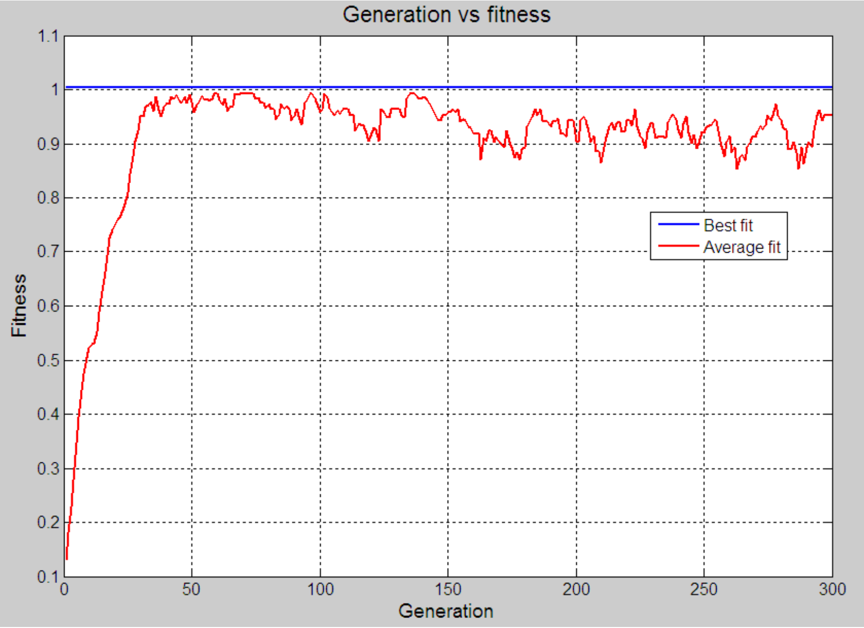
$$0 \leq x \leq 1$$

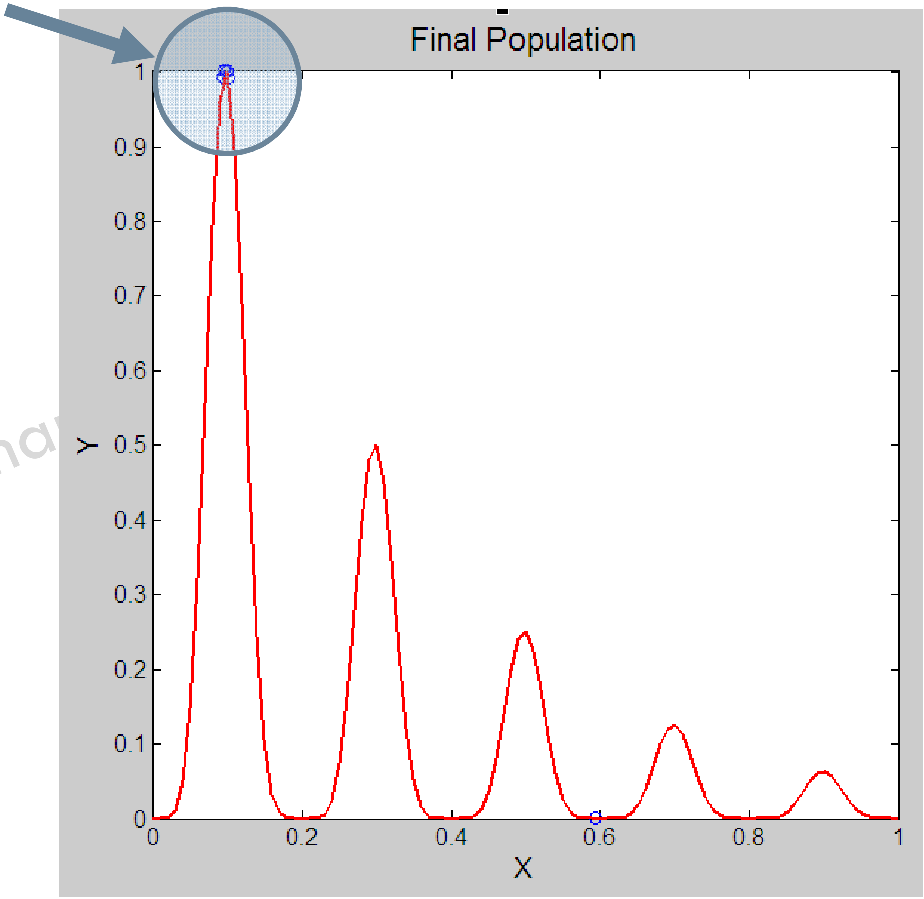
Solve this problem using simple Genetic Algorithms



After Generation 200

The population are in and around the global optimal solution





Multi-modal optimization

Simple modification of Simple Genetic Algorithms can capture all the optimal solution of the problem including global optimal solutions

Basic idea is that reduce the fitness of crowded solution, which can be implemented using following three steps.

Sharing function $$Sh(d_{ij}) = \begin{cases} 1 - (d_{ij}/\sigma), & \text{if } d_{ij} < \sigma\,; \\ 0, & \text{otherwise.} \end{cases}$$

Niche count $$nc_i = \sum_{j=1}^{N} Sh(d_{ij})$$

Modified fitness $$f_i' = \frac{f_i}{nc_i}$$

Hand calculation

Maximize $f(x) = |sin(\pi x)|$

$$0 \leq x \leq 2$$

Sol	String	Decoded value	x	f
1	110100	52	1.651	0.890
2	101100	44	1.397	0.942
3	011101	29	0.921	0.246
4	001011	11	0.349	0.890
5	110000	48	1.524	0.997
6	101110	46	1.460	0.992

Distance table

dij	1	2	3	4	5	6
1	0	0.254	0.73	1.302	0.127	0.191
2	0.254	0	0.476	1.048	0.127	0.063
3	0.73	0.476	0	0.572	0.603	0.539
4	1.302	1.048	0.572	0	1.175	1.111
5	0.127	0.127	0.603	1.175	0	0.064
6	0.191	0.063	0.539	1.111	0.064	0

Sharing function values

sh(dij)	1	2	3	4	5	6	nc
1	1	0.492	0	0	0.746	0.618	2.856
2	0.492	1	0.048	0	0.746	0.874	3.16
3	0	0.048	1	0	0	0	1.048
4	0	0	0	1	0	0	1
5	0.746	0.746	0	0	1	0.872	3.364
6	0.618	0.874	0	0	0.872	1	3.364

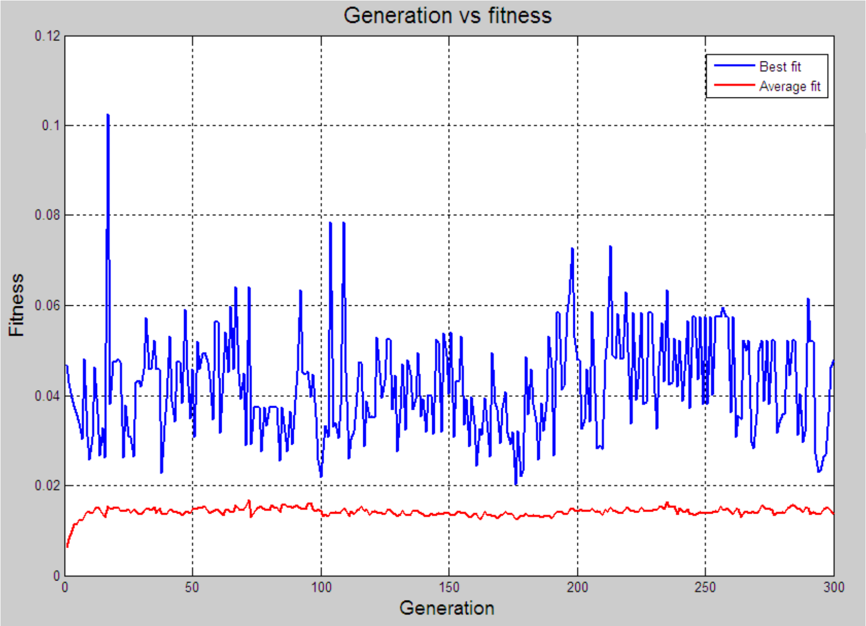
Sharing fitness value

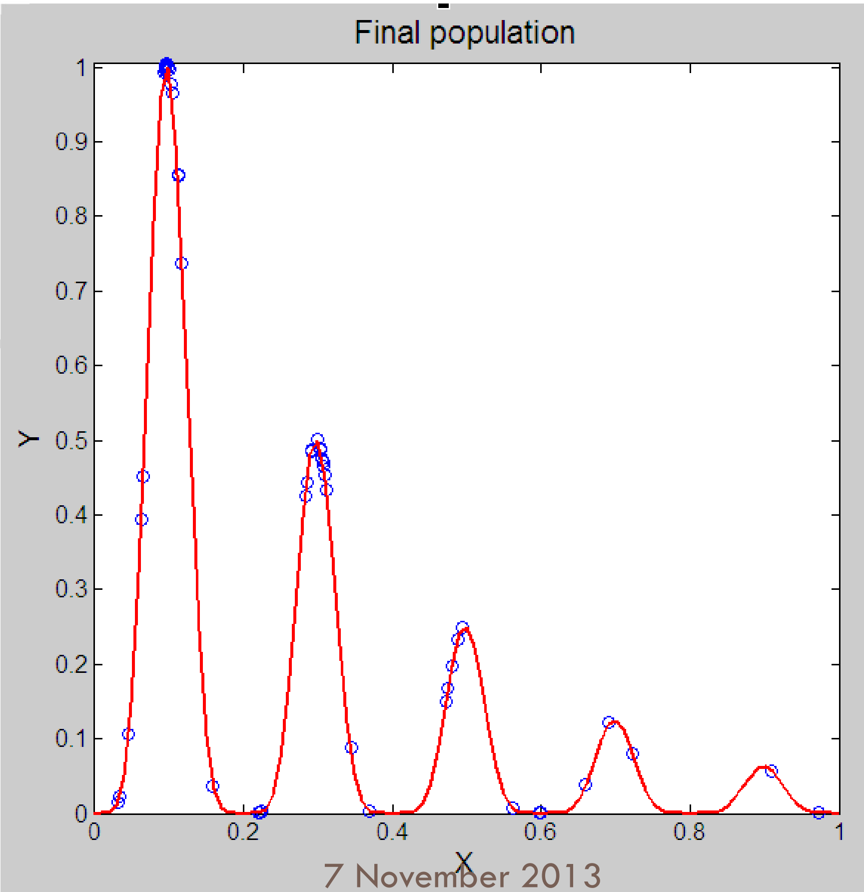
Sol	String	Decoded value	x	f	nc	f'
1	110100	52	1.651	0.890	2.856	0.312
2	101100	44	1.397	0.942	3.160	0.300
3	011101	29	0.921	0.246	1.048	0.235
4	001011	11	0.349	0.890	1.000	0.890
5	110000	48	1.524	0.997	3.364	0.296
6	101110	46	1.460	0.992	3.364	0.295

Solutions obtained using modified fitness value

$$Minimize\ f(x) = 2^{-2\left(\frac{(x-0.1)}{0.8}\right)^2} sin^6(5\pi x)$$

$$0 \leq x \leq 1$$





Evolutionary Strategies

- ES use real parameter value
- ES does not use crossover operator
- It is just like a real coded genetic algorithms with selection and mutation operators only

Two members ES: (1+1) ES

- In each iteration one parent is used to create one offspring by using Gaussian mutation operator

Two members ES: (1+1) ES

- Step1: Choose a initial solution x and a mutation strength σ
- Step2: Create a mutate solution

$$y = x + N(0, \sigma)$$

- Step 3: If $f(y) < f(x)$, replace x with y
- Step4: If termination criteria is satisfied, stop, else go to step 2

Two members ES: (1+1) ES

- Strength of the algorithm is the proper value of σ

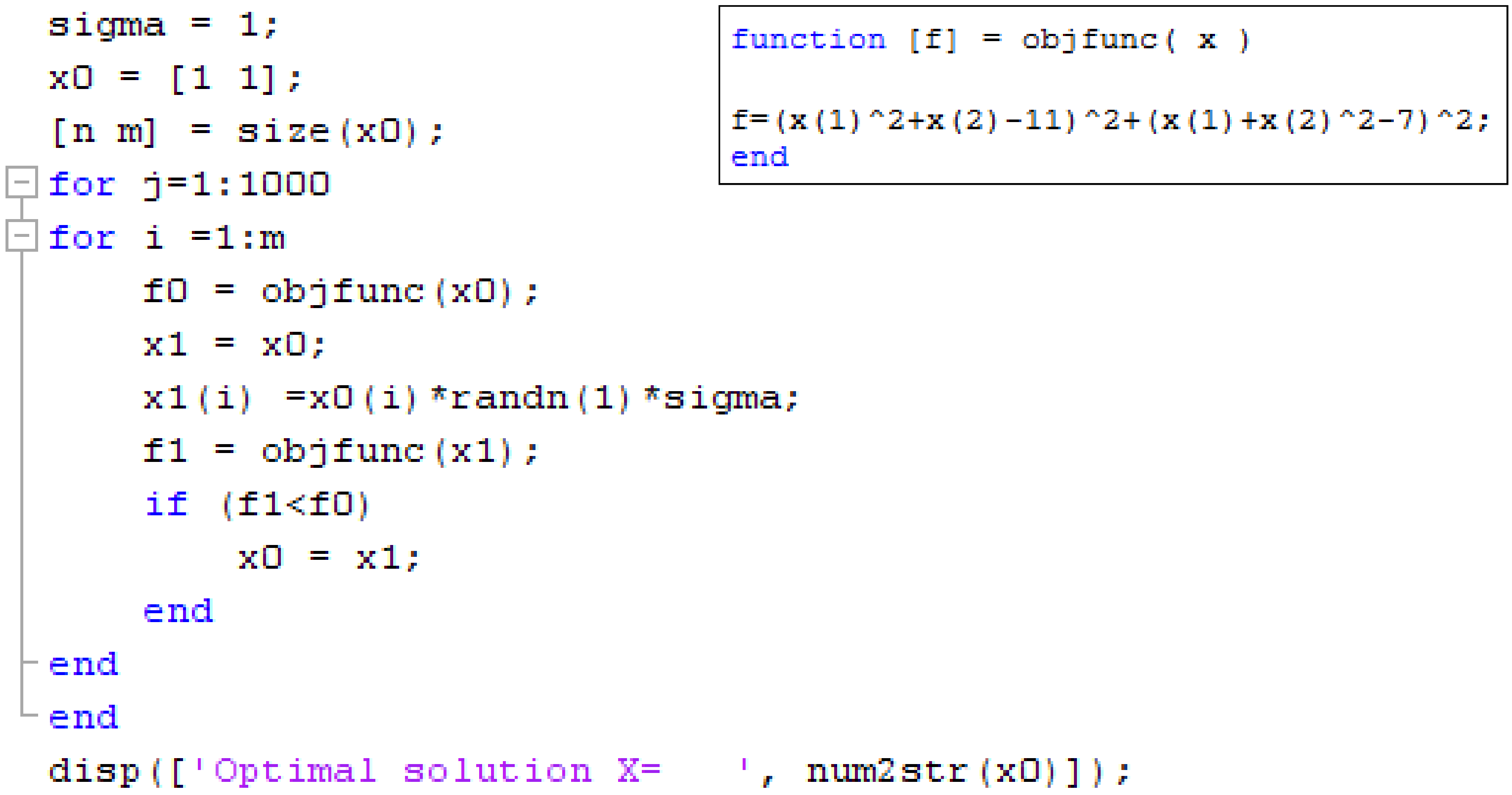
- Rechenberg postulate
    - The ratio of successful mutations to all the mutations should be 1/5. If this ratio is greater than 1/5, increase mutation strength. If it is less than 1/5, decrease the mutation strength.

Two members ES: (1+1) ES

- A mutation is defined as successful if the mutated offspring is better than the parent solution.
- If P_s is the ratio of successful mutation over *n* trial, Schwefel (1981) suggested a factor $C_d = 0.817$ in the following σ update rule

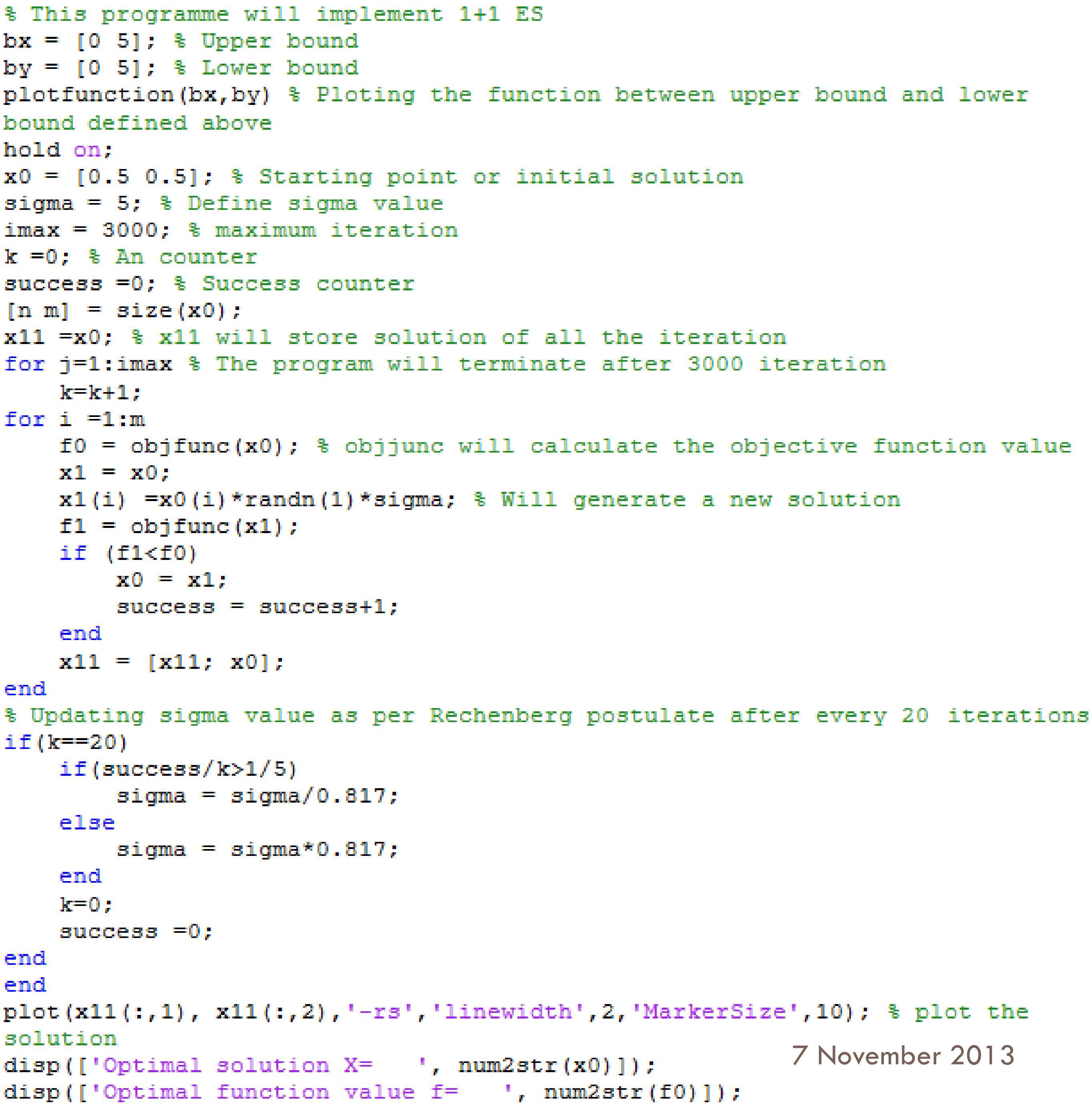
$$\sigma^{t+1} = \begin{cases} C_d \sigma^t & if\ P_s < 1/5 \\ \dfrac{1}{C_d} \sigma^t & if\ P_s < 1/5 \\ \sigma^t & if\ P_s = 1/5 \end{cases}$$

Matlab code



```
sigma = 1;
x0 = [1 1];
[n m] = size(x0);
for j=1:1000
for i =1:m
    f0 = objfunc(x0);
    x1 = x0;
    x1(i) =x0(i)*randn(1)*sigma;
    f1 = objfunc(x1);
    if (f1<f0)
        x0 = x1;
    end
end
end
disp(['Optimal solution X=   ', num2str(x0)]);
```

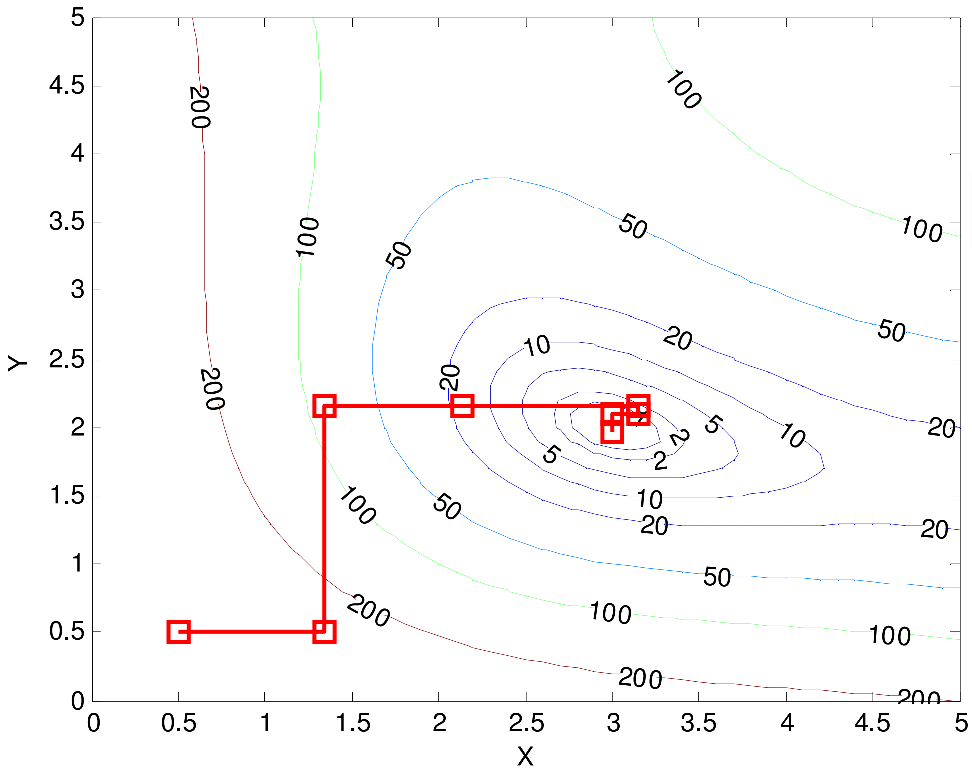
```
function [f] = objfunc( x )

f=(x(1)^2+x(2)-11)^2+(x(1)+x(2)^2-7)^2;
end
```



```
% This programme will implement 1+1 ES
bx = [0 5]; % Upper bound
by = [0 5]; % Lower bound
plotfunction(bx,by) % Ploting the function between upper bound and lower
bound defined above
hold on;
x0 = [0.5 0.5]; % Starting point or initial solution
sigma = 5; % Define sigma value
imax = 3000; % maximum iteration
k =0; % An counter
success =0; % Success counter
[n m] = size(x0);
x11 =x0; % x11 will store solution of all the iteration
for j=1:imax % The program will terminate after 3000 iteration
    k=k+1;
for i =1:m
    f0 = objfunc(x0); % objjunc will calculate the objective function value
    x1 = x0;
    x1(i) =x0(i)*randn(1)*sigma; % Will generate a new solution
    f1 = objfunc(x1);
    if (f1<f0)
        x0 = x1;
        success = success+1;
    end
    x11 = [x11; x0];
end
% Updating sigma value as per Rechenberg postulate after every 20  iterations
if(k==20)
    if(success/k>1/5)
        sigma = sigma/0.817;
    else
        sigma = sigma*0.817;
    end
    k=0;
    success =0;
end
end
plot(x11(:,1), x11(:,2),'-rs','linewidth',2,'MarkerSize',10); % plot the
solution
disp(['Optimal solution X=   ', num2str(x0)]);
disp(['Optimal function value f=   ', num2str(f0)]);
```

Some results of 1+1 ES

$$Minimize\ f = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$



Optimal Solution is
X*= [3.00 1.99]

Objective function value f
= 0.0031007

Multimember ES

<u>$(\mu + \lambda)$ ES</u>

Step1: Choose an initial population of μ solutions and mutation strength σ

Step2: Create λ mutated solution

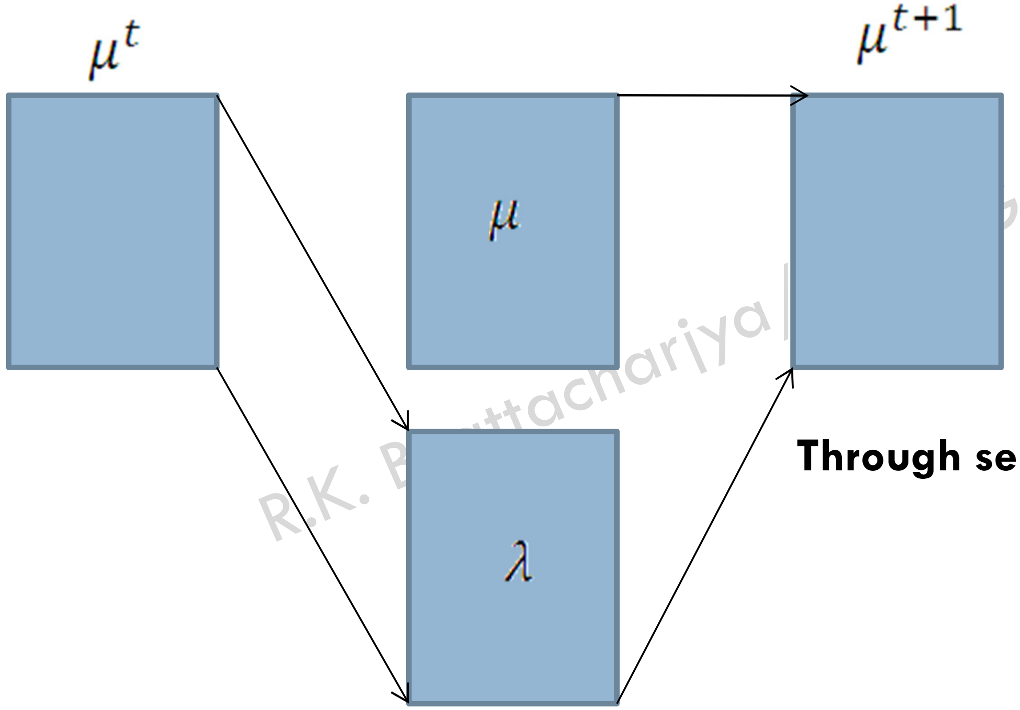
$$y^i = x^i + N(0, \sigma)$$

Step3: Combine x and y, and choose the best solutions μ

Step4: Terminate? Else go to step 2

Multimember ES

$(\mu + \lambda)$ ES



μ^t

μ^{t+1}

μ

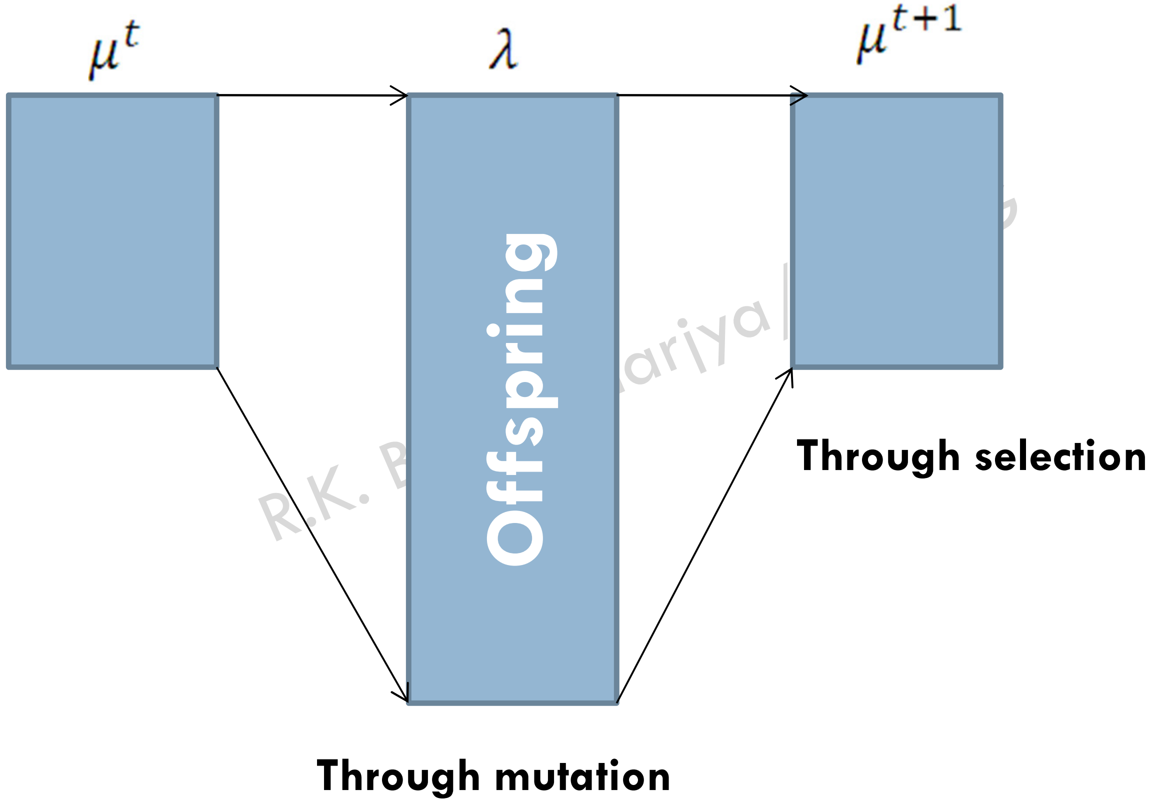
λ

Through selection

Through mutation

Multimember ES

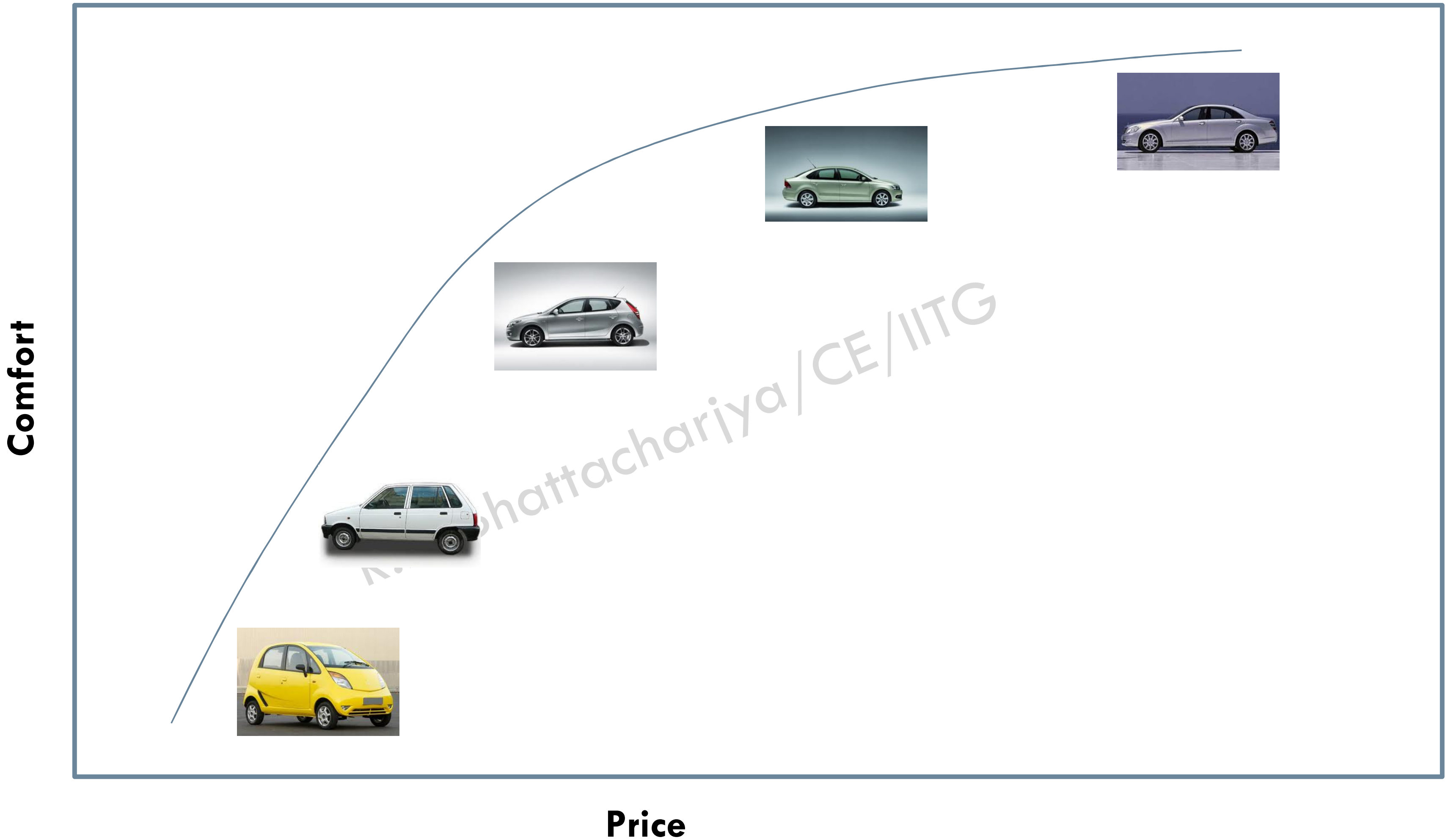
(μ, λ) ES



Through selection

Through mutation

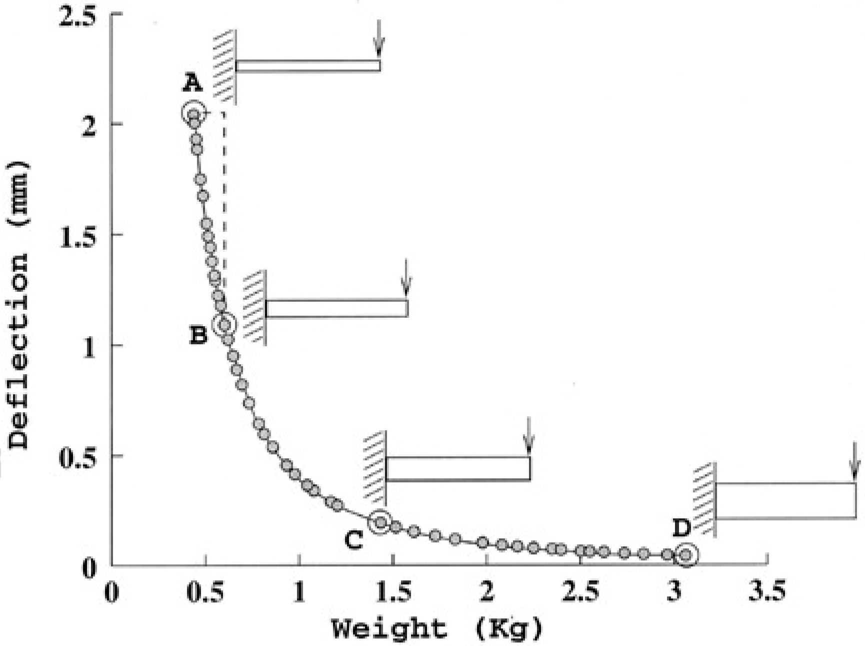
Multi-objective optimization



Multi-objective optimization

Two objectives are

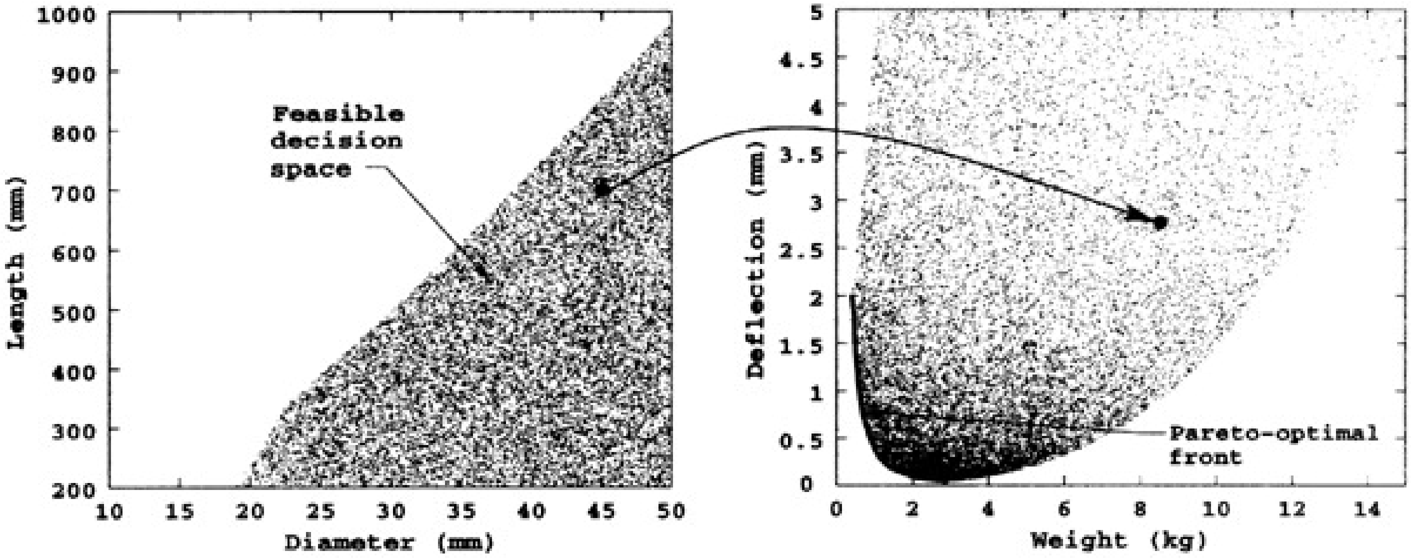
- Minimize weight
- Minimize deflection



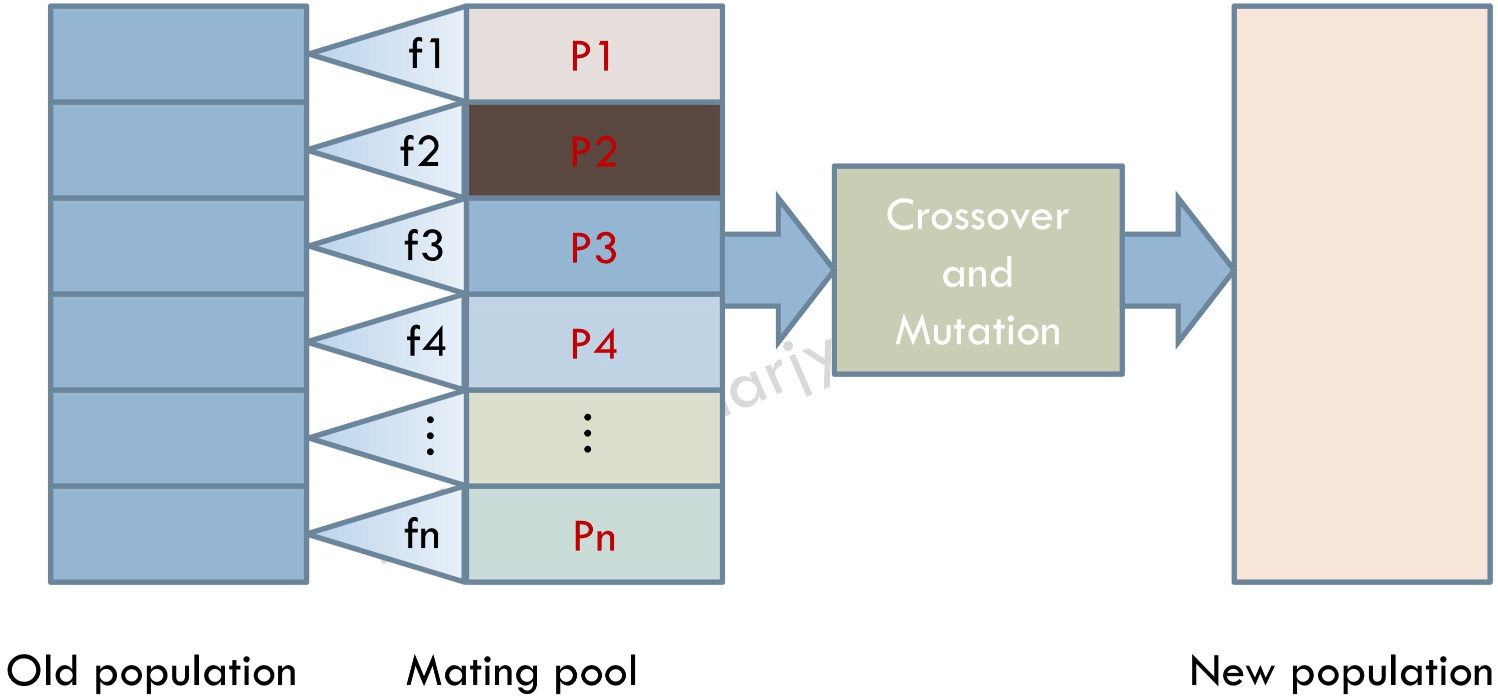
Multi-objective optimization

- More than one objectives
- Objectives are conflicting in nature
- Dealing with two search space
  - Decision variable space
  - Objective space
- Unique mapping between the objectives and often the mapping is non-linear
- Properties of the two search space are not similar
- Proximity of two solutions in one search space does not mean a proximity in other search space

Multi-objective optimization



Vector Evaluated Genetic Algorithm (VEGA)



Propose by Schaffer (1984)

Non-dominated selection heuristic

Give more emphasize on the non-dominated solutions of the population

This can be implemented by subtracting $\in$ from the dominated solution fitness value

Suppose N' is the number of sub-population and n' is the non-dominated solutions. Then total reduction is $(N' - n') \in$.

The total reduction is then redistributed among the non-dominated solution by adding an amount $(N' - n') \in / n'$

This method has two main implications

Non-dominated solutions are given more importance

Additional equal emphasis has been given to all the non-dominated solution

Weighted based genetic algorithm (WBGA)

- The fitness is calculated

$$F = \sum_{j=1}^{M} w_j \frac{f_i - f_j^{min}}{f_j^{max} - f_j^{min}}$$

- The spread is maintained using the sharing function approach

Sharing function $$Sh(d_{ij}) = \begin{cases} 1 - (d_{ij}/\sigma), & \text{if } d_{ij} < \sigma; \\ 0, & \text{otherwise.} \end{cases}$$

Niche count $$nc_i = \sum_{j=1}^{N} Sh(d_{ij})$$

Modified fitness $$F' = \frac{F}{nc}$$

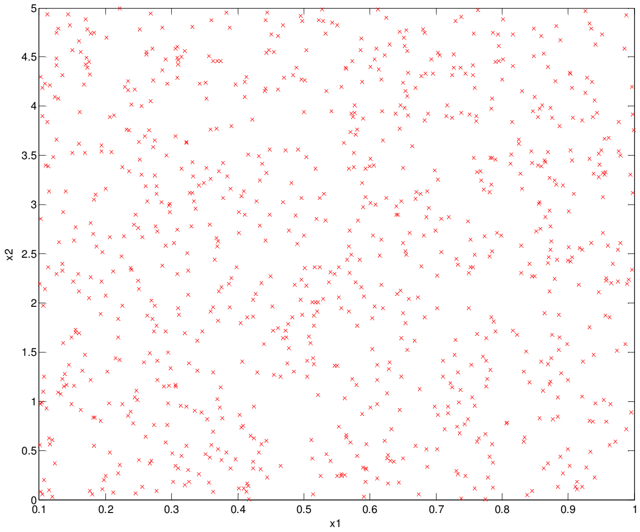
Multiple objective genetic algorithm (MOGA)

$$Maximize\ f_1 = 1.1 - x_1$$

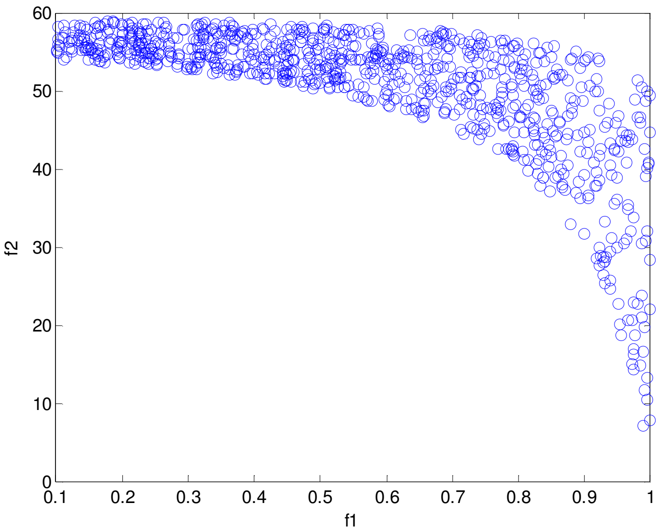
$$Maximize\ f_2 = 60 - \frac{1 + x_1}{x_2}$$

$$Subject\ to\quad 0.1 \le x_1 \le 1$$

$$0 \le x_2 \le 5$$

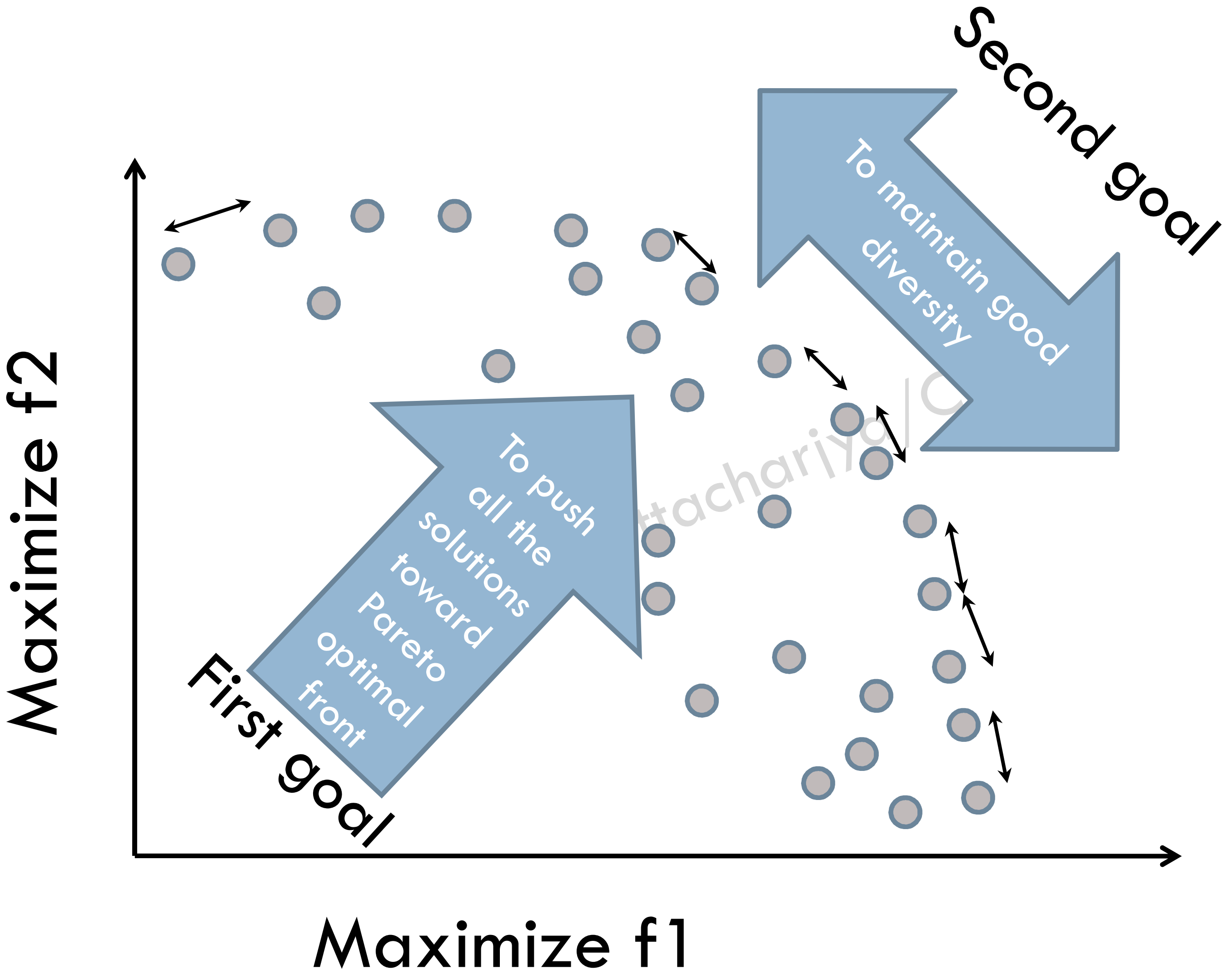


Solution space



Objective space

Multiple objective genetic algorithm (MOGA)

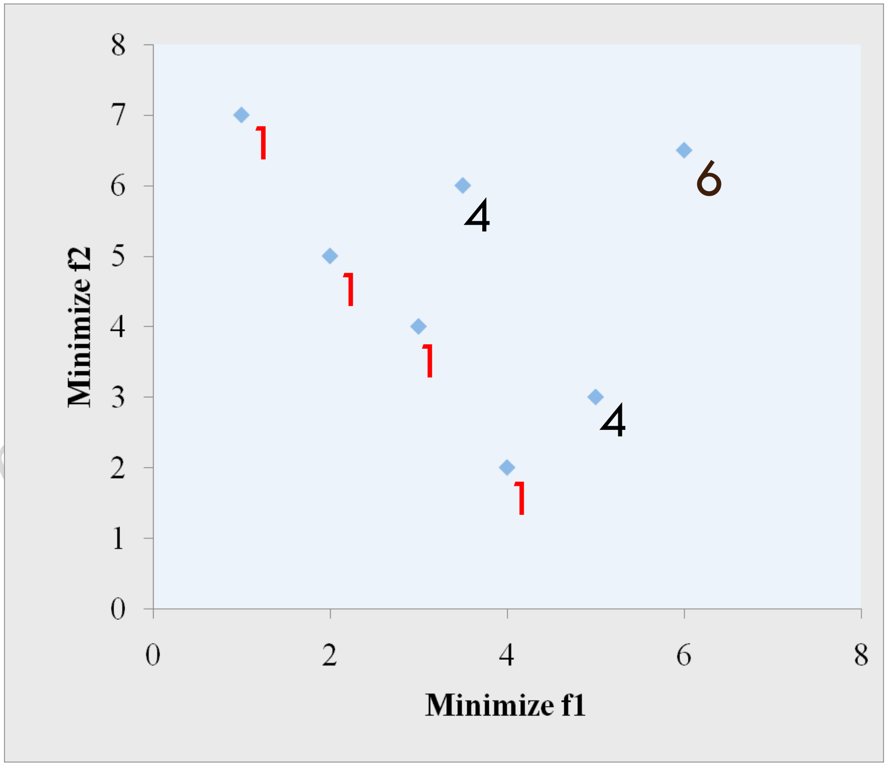


Multiple objective genetic algorithm (MOGA)

Fonseca and Fleming (1993) first introduced multiple objective genetic algorithm (MOGA)

The assigned fitness value based on the non-dominated ranking.

The rank is assigned as $r_i = 1 + n_i$ where r_i is the ranking of the i^{th} solution and n_i is the number of solutions that dominate the solution.



Multiple objective genetic algorithm (MOGA)

- Fonseca and Fleming (1993) maintain the diversity among the non-dominated solution using niching among the solution of same rank.
- The normalize distance was calculated as,

$$d_{i,j} = \sqrt{\sum_{k=1}^{M} \left(\frac{f_k^i - f_k^j}{f_k^{max} - f_k^{min}} \right)^2}$$

- The niche count was calculated as,

$$nc_i = \sum_{j=1}^{\mu(r_i)} Sh(d_{ij})$$

NSGA

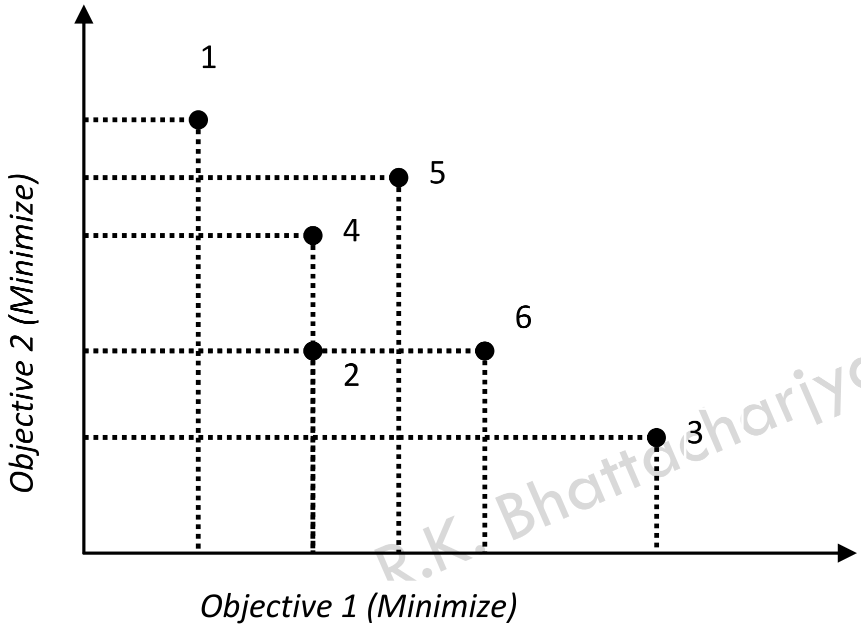
- Srinivas and Deb (1994) proposed NSGA
- The algorithm is based on the non-dominated sorting.
- The spread on the Pareto optimal front is maintained using sharing function

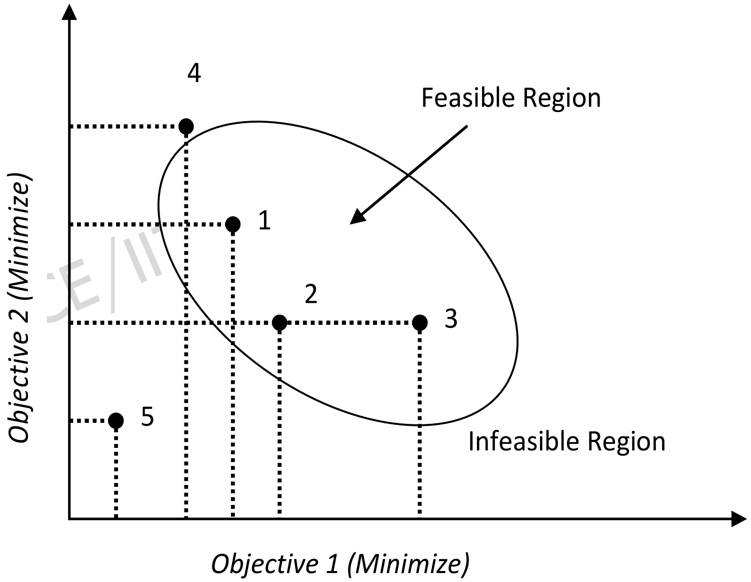
$$d_{i,j} = \sqrt{\sum_{k=1}^{P_1} \left(\frac{x_k^i - x_k^j}{x_k^{max} - x_k^{min}} \right)^2}$$

NSGA II

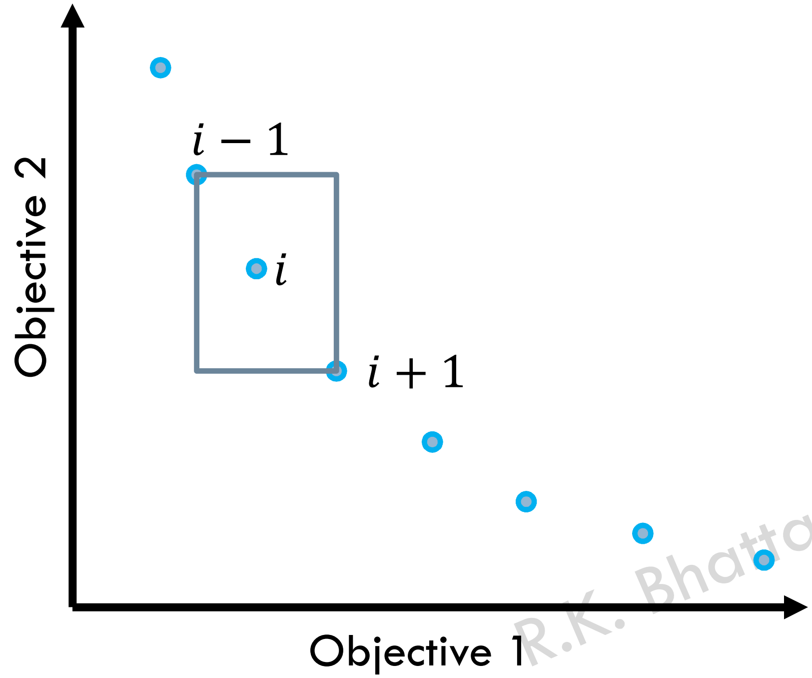
- Non-dominated Sorting Genetic Algorithms
  - NSGA II is an elitist non-dominated sorting Genetic Algorithm to solve multi-objective optimization problem developed by Prof. K. Deb and his student at IIT Kanpur.
  - It has been reported that NSGA II can converge to the global Pareto-optimal front and can maintain the diversity of population on the Pareto-optimal front

Non-dominated sorting





Calculation crowding distance



Cd, the crowded distance is the perimeter of the rectangle constituted by the two neighboring solutions

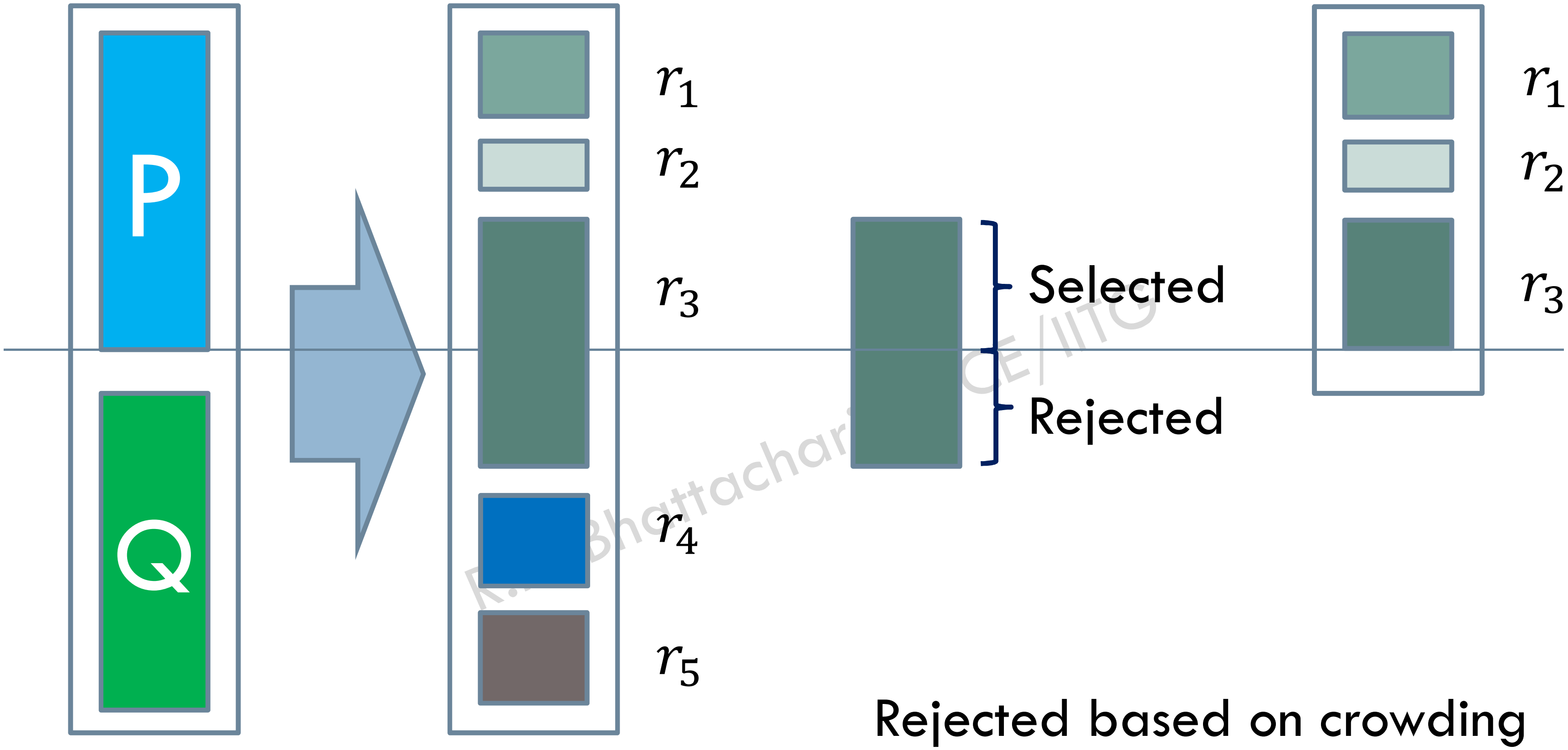
Cd value more means that the solution is less crowded

Cd value less means that the solution is more crowded

Crowded tournament operator

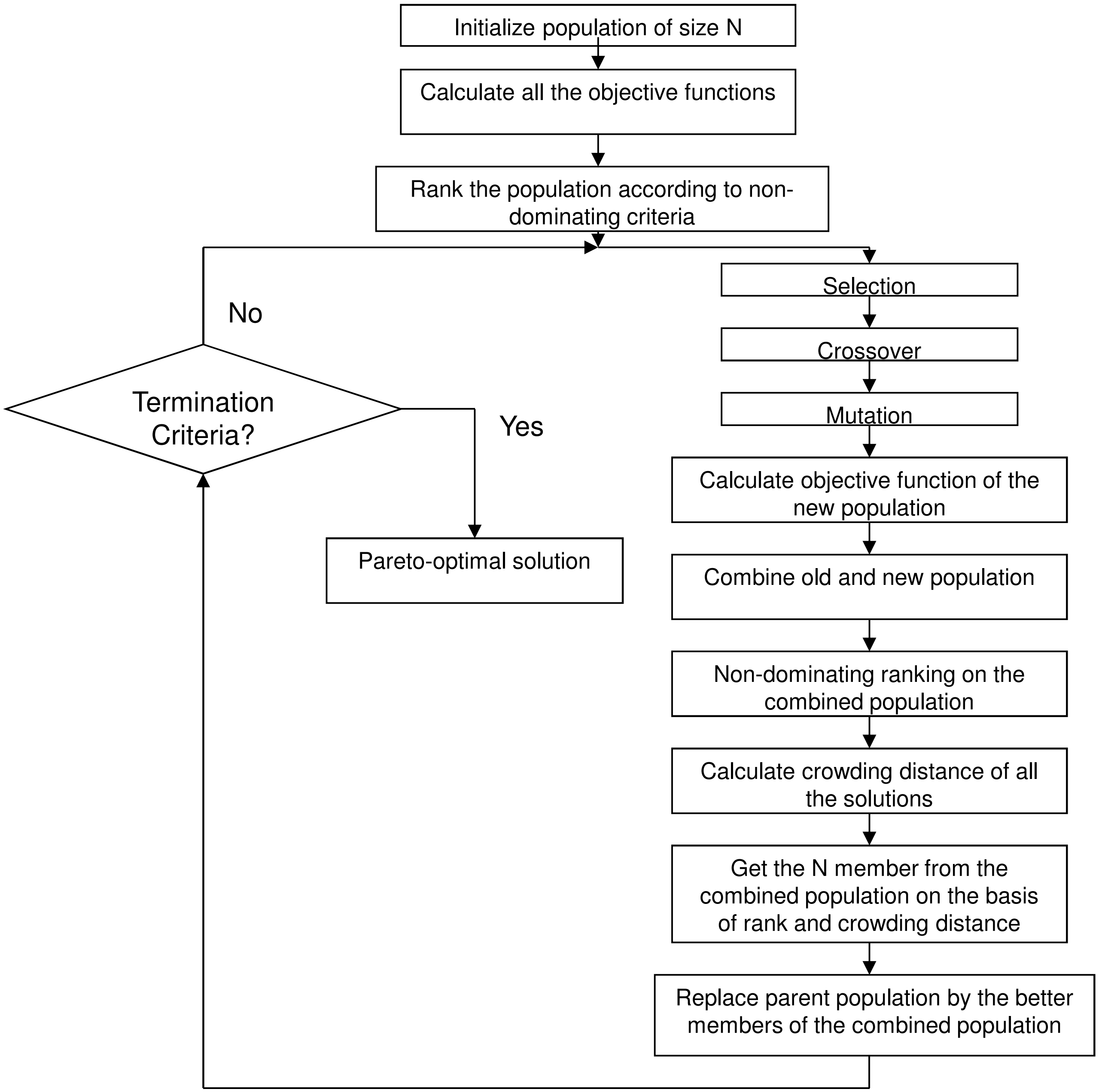
- A solution i wins a tournament with another solution j,
  - If the solution i has better rank than j, i.e. $r_i < r_j$
  - If they have the same rank, but i has a better crowding distance than j, i.e. $r_i = r_j$ and $d_i > d_j$.

Replacement scheme of NSGA II



Non dominated sorting

Rejected based on crowding distance



Initialize population of size N

Calculate all the objective functions

Rank the population according to non-dominating criteria

Selection

Crossover

Mutation

Calculate objective function of the new population

Combine old and new population

Non-dominating ranking on the combined population

Calculate crowding distance of all the solutions

Get the N member from the combined population on the basis of rank and crowding distance

Replace parent population by the better members of the combined population

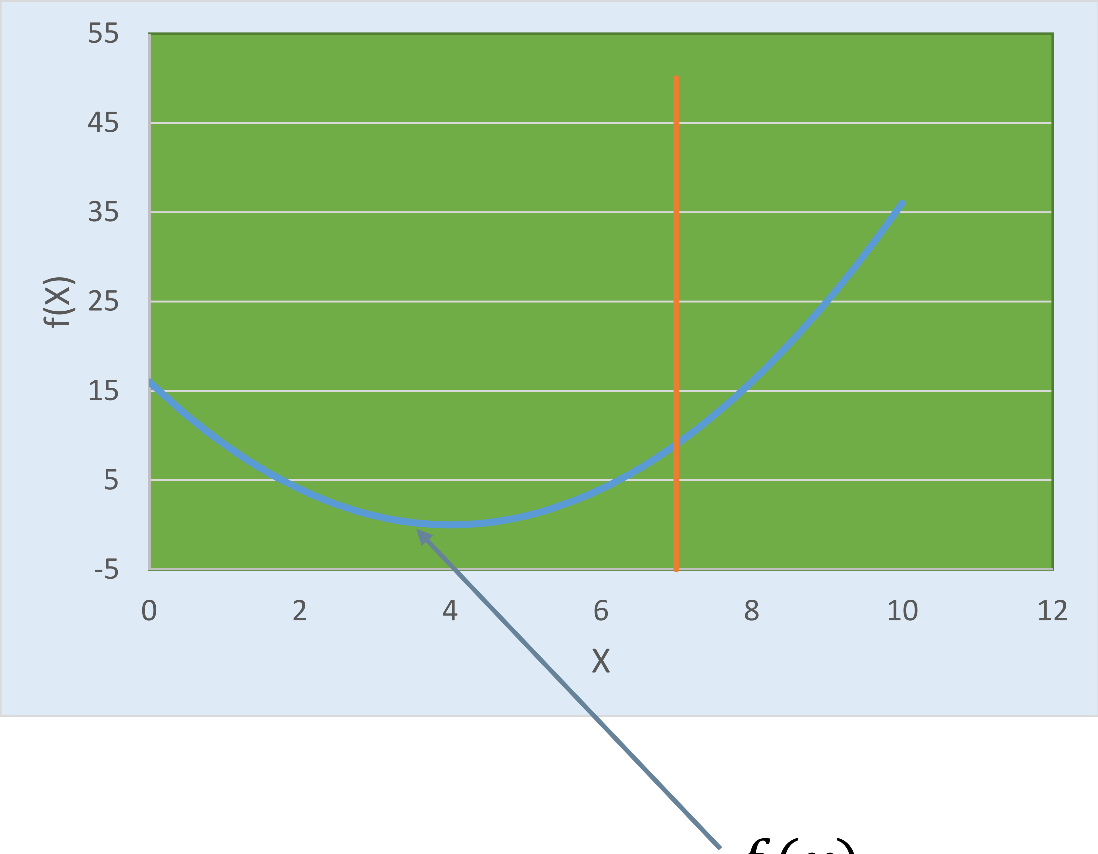
Termination Criteria?

No

Yes

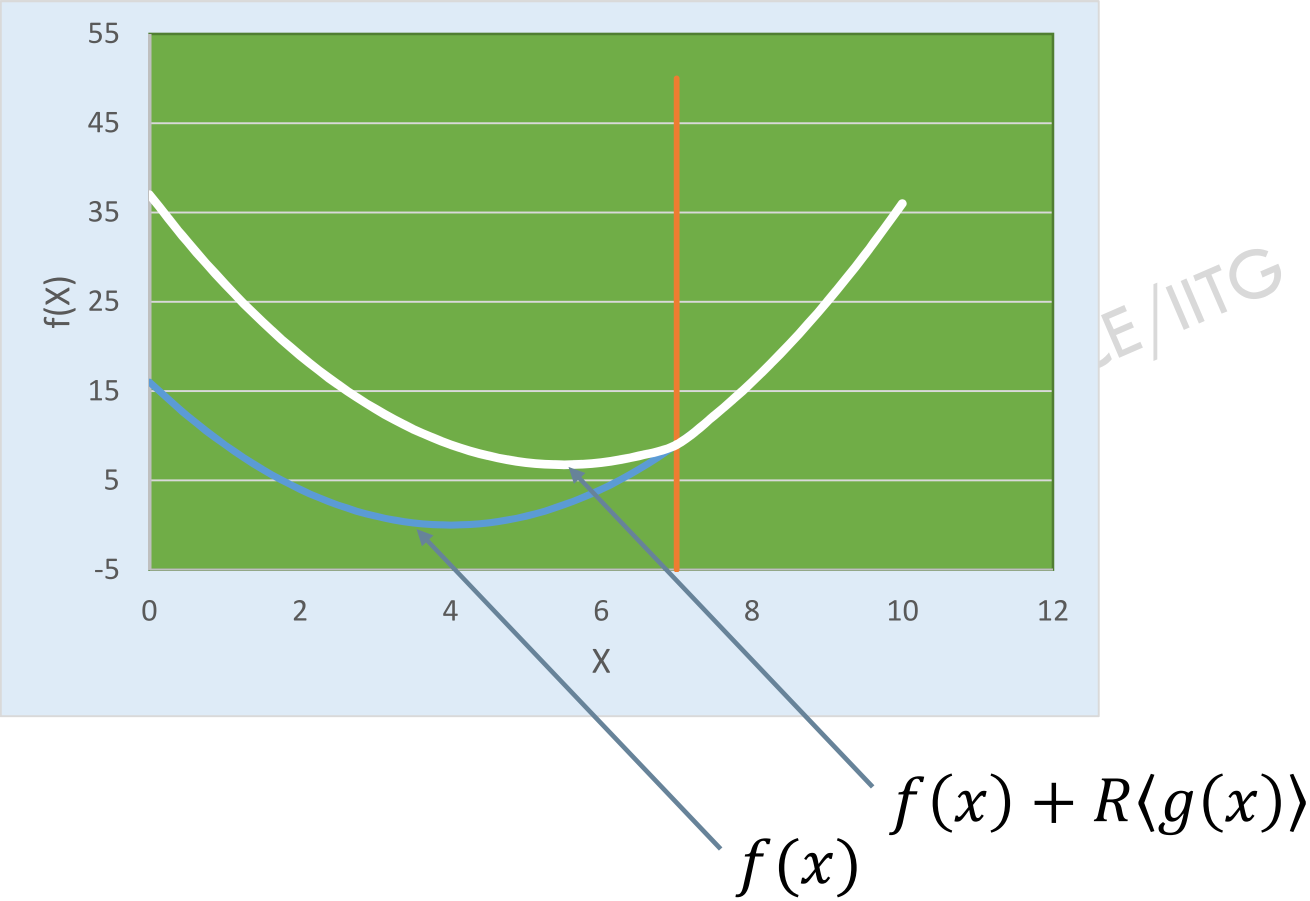
Pareto-optimal solution

Constraints handling in GA



$f(x)$

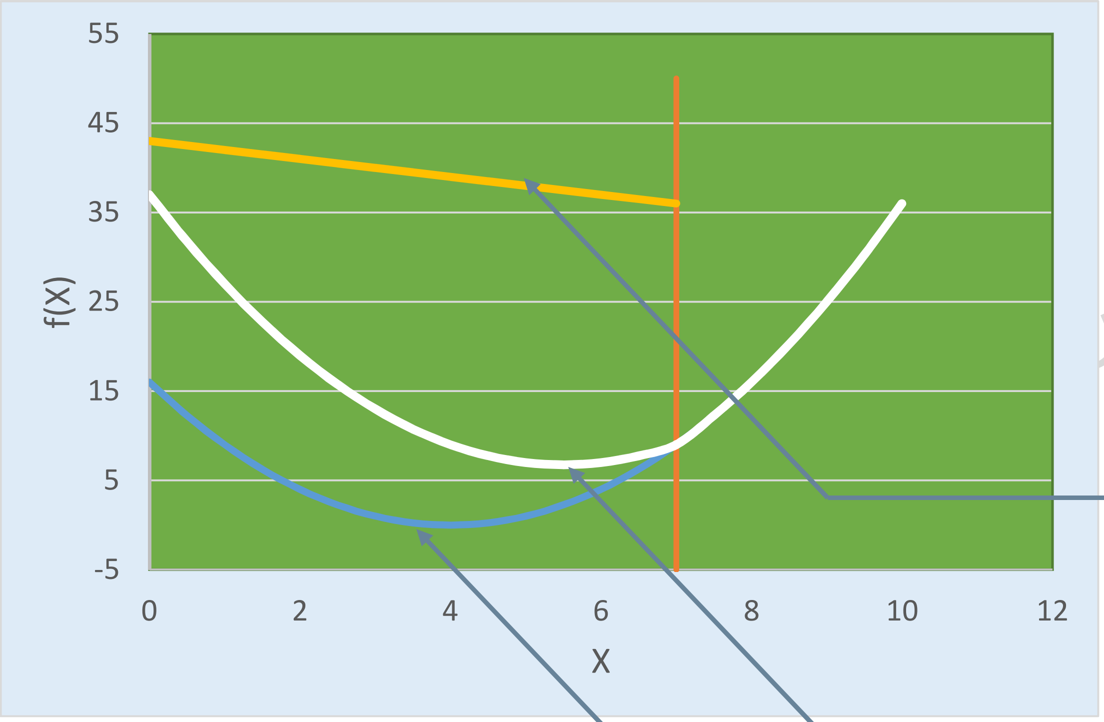
Constraints handling in GA



$f(x) + R\langle g(x)\rangle$

$f(x)$

Constraints handling in GA



$f_{max} + \langle g(x) \rangle$

$f(x) + R\langle g(x) \rangle$

$f(x)$

Constrain handling in GA

Minimize $f(X)$

Subject to

$$g_j(x) \leq 0 \qquad j = 1,2,3,\dots,J$$

$$h_k(x) = 0 \qquad k = 1,2,3,\dots,K$$

Deb's approach

$$F = f(X) \qquad \text{If } X \text{ is feasible}$$

$$= f_{max} + \sum_{j=1}^{J} \langle g_j(x) \rangle + \sum_{k=1}^{K} |h_k(x)| \qquad \text{Otherwise}$$

THANKS