## Non traditional Algorithms

1. Simulated Annealing
2. Genetic Algorithm
3. Particle Swarm
4. Ant Colony

1. They are based on natural phenomena

2. They are all "stochastic" methods. There is a good combination of random search and direction. Thus there is both exploration and exploitation.

3. Suitable for non-convex problems also; i.e., they can find the global optima out of many local optima.

4. The algorithms shown here are for unconstrained problems. Penalty functions need to be used if there are constraints.

## Genetic Algorithms (GA)

It is a Heuristic algorithm which is useful for multi-modal problems (many local optima).

Good combination of "random exploration" combined with "direction".

Imitates natural selection through "survival of the fittest".

All creatures existing today are the result of "survival of the fittest" by trial and error over millions of years.

Consider a species of animal living in nature:

1. Weak, unfit or disease prone animals perish.

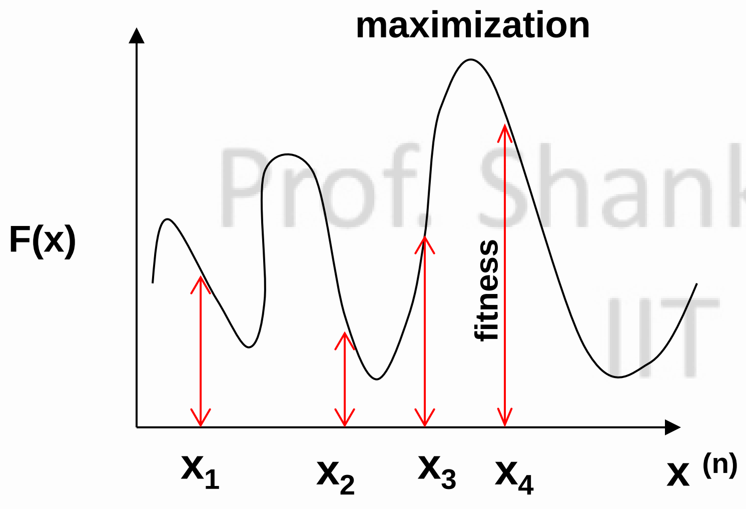
2. They are unable to find a mate.

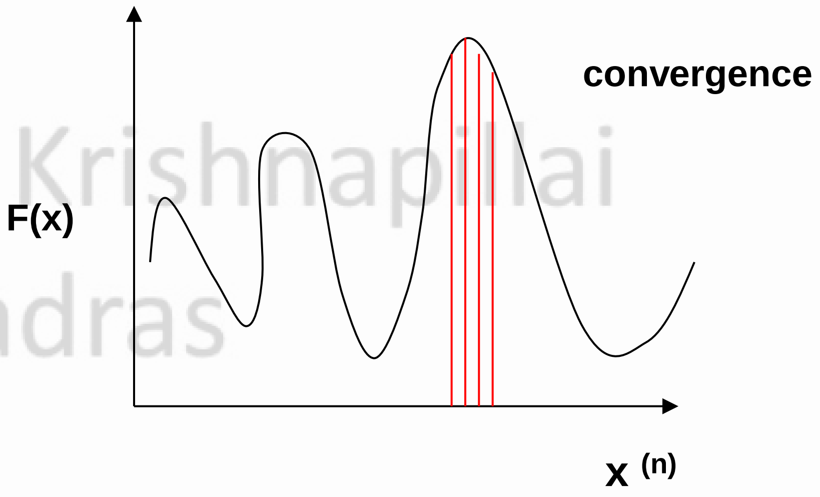
3. Only animals in good fitness can find a mate and produce offspring (i.e pass on their genes to next generation) .
   This is simulated by the ranking and cross-over operations in GA.

5. The genes of the offspring may not be an exact mixture of parents.
   Small random changes called "mutations" may take place in the offspring's genes.

Some mutations are beneficial, as they may give some new property such as resistance to a disease, more strength etc.

GA also simulates mutations in the form of randomly making small changes in the solutions, hopefully improving the solution.





How does GA simulate natural selection?

GA starts with a randomly picked population $x^{(1)}, x^{(2)} \ldots\ldots x^{(n)}$ in the domain.

It ranks them according to fitness (i.e obj. function values) - worst members are rejected.

Crossover (mating) and mutation are carried out between fittest candidates.

The resulting offspring form the next iteration, and give marginally better fitness than the previous generation.

Over many iterations (generations) all the values of x converge to the global maxima.

## GA

$$\begin{tikzcd}
 & \text{GA} \arrow[dl] \arrow[dr] & \\
\text{Binary GA} & & \text{Real coded GA}
\end{tikzcd}$$

**Binary GA**

**All numbers are expressed as binary strings (string of bits).**

**Real coded GA** ✓ **For exam**

**Numbers are used in floating point format. Simpler to program.**

**Note: The GA presented here is for maximizing a function**

## Binary GA

**Illustrated with a simple non-convex example.**

$$maximize: \quad x.\sin(10\pi x)+1$$

$$-1 \le x \le 2$$

Illustration of Binary GA operations.

1. Range of design variable = 3 (i.e -1 to 2)

2. Decide on how many divisions we need in this range?

Suppose we want decimal precision of $10^{-6}$;

It means we must have $\dfrac{3}{10^{-6}} = 3\times10^{6}$ steps along the x axis.

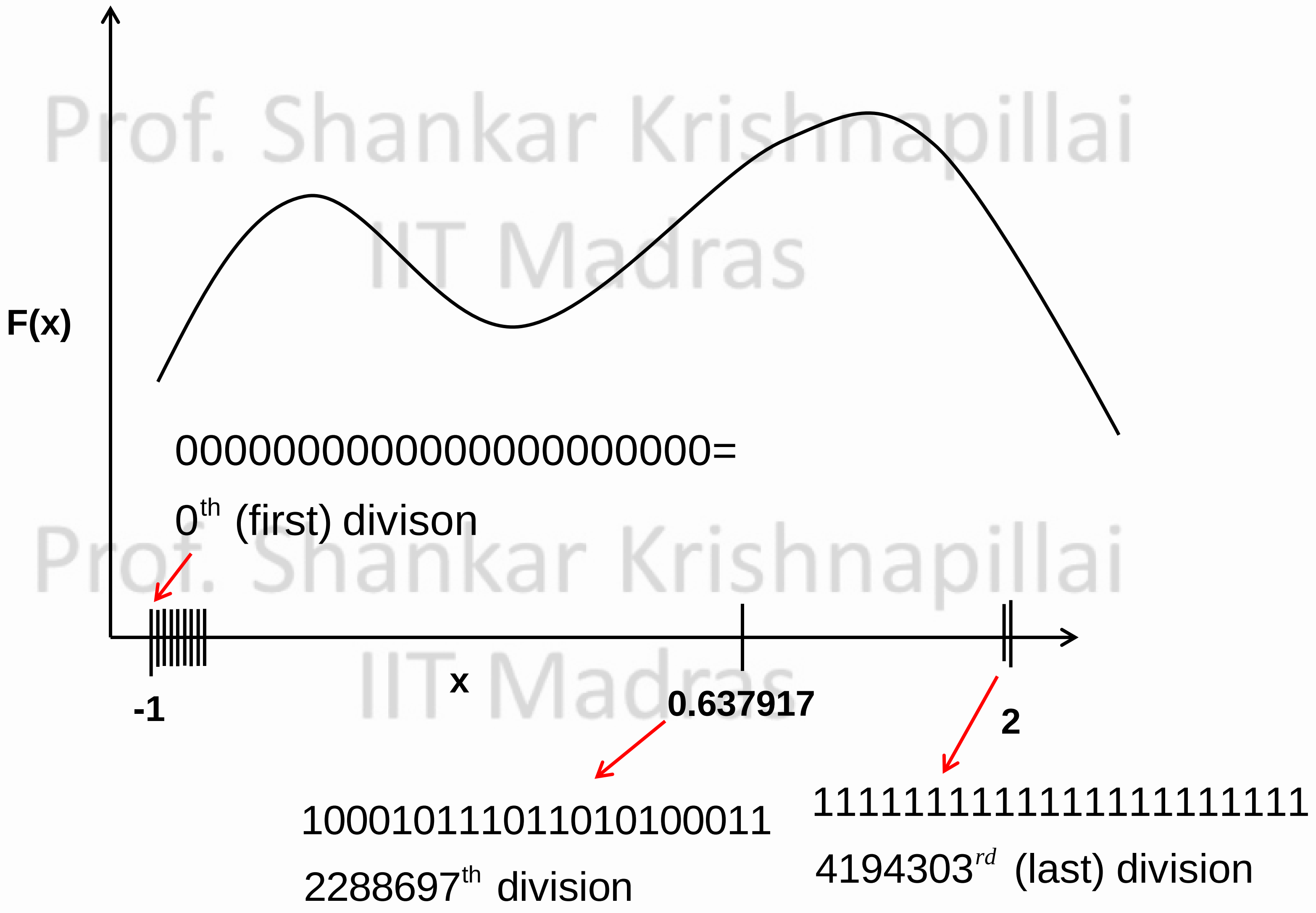
How many bits do we need to represent $3 \times 10^6$ integer divisions?

22 bit representation gives $2^{22}$= 4194304 integers.

21 bit representation gives $2^{21}$= 2097152 integers.

Hence we need 22 bits.

Now, the divison numbers (integers) from -1 to 2 has to be represented by 22 bits.


F(x)
0000000000000000000000=
0th (first) divison
x
-1
0.637917
2
1000101110110101000011
2288697th division
1111111111111111111111
4194303rd (last) division

**Now GA creates a "population" of 10 *random samples* in the range -1 to 2.**

| | Coordinate (chromosome) | F(x) - fitness |
|---|---|---|
| x1 | 1000101110110101000111 (0.637197) | 1.586345 |
| x2 | 0000001110000000010000 (-0.958973) | 0.078878 |
| x3 | 1110000000111111000101 (1.627888) | 2.250650 |
| …x10 | .. | .. |

**x3 has the best fitness**

## Basic Cross-over operation:

**Two parents are selected for cross-over or mating; say x2 and x3.**

**x2** 00000 | 01110000000010000

**x3** 11100 | 00000111111000101

**swapped**

**5th bit randomly chosen for cross-over.**

**A random number '*r*' can be generated between 1 to 22 to decide the cut point.**

$$\text{2 offsprings} \begin{cases} \textbf{x2'} & 00000\ 00000111111000101 = -0.998113 \\ \textbf{x3'} & 11100\ 01110000000010000 = 1.666028 \end{cases}$$

It is seen that x3'= 1.666028 has higher fitness than both parents. Generally this is the case. But x2' is worse and not an improvement. It will be eliminated from the pool.

## Mutation Operation:

**A member in the new population can be mutated by changing 1 bit in their strings. Hopefully the fitness would improve.**

| | |
|---|---|
| **x3'** | 1110 $\boxed{0}$ 01110000000010000= 1.666028 |
| Mutated_**x3'** | 1110 $\boxed{1}$ 01110000000010000 =1.721638 |

## Continuous or Floating Point GA
## (also called Real coded GA)

**Here the numbers are represented only by decimal (Floating point).**

|  | Coordinate (chromosome) | F(x) = fitness |
|---|---|---|
| x1 | 0.637197 | 1.586345 |
| x2 | -0.958973 | 0.078878 |
| x3 | 1.627888 | 2.250650 |
| …x10 | 1.963257 .. | -0.819386 |

**Let us take 2 parents, say x1 and x3.**
**Crossover between x1 and x3 can be conducted by interpolation.**

**Offspring 1 = $\beta x_1 + (1-\beta)x_3$.** **($\beta$ is a random number between 0 and 1) (say $\beta$=0.4)**

**=1.2316116**

**Offspring 2 = $(1-\beta)x_1 + \beta x_3$= 1.0334734**

## Mutation in Floating Point GA

**Out of a population of 10, decide how many members (*n*) to mutate.**

**Probability of mutation is n/10. In some GA subroutines we have to specify this probability of mutation.**

**Implementation:**

**Generate a random integer between 1 and 10, pick that member. Replace it by a new member randomly picked within the variable range.**

For both Binary and Continuous GA,

First generation:

These are made of the first 10 ‘*x*’ values. A portion (say half) are rejected. Remainder are crossed over and mutated to give offsprings.

Second Generation:
Offsprings of first generation (say 8) + a few parents (say 2), give the second generation of 10.

This second generation will undergo crossover and mutation to give the 3rd generation.

The generations will continue until all members in the population converge to the Global Optima.

At this stage the algorithm can be stopped.

## Some schemes for Parent Selection, Crossover, Mutation.

**Parent Selection:**

**The Roulette Wheel Method is the most popular method.**
**Here, the probability of a parent being selected for crossover is proportional to its fitness.**

**Suppose population N=10 and n=5 is the pool of fittest members (i.,e parents).**

**In this method, first the fitnesses of parents have to be normalized to a positive scale to avoid negative probabilities.**

For each member $'i'$ in the pool of 5;

Find : $p_i = \dfrac{f(x_i)}{\sum_{i=1}^{5} f(x_i)}$ $(probability\ of\ a\ member\ 'i'\ being\ selected\ for\ Crossover)$

$$q_i = \sum_{1}^{i} p_i \qquad (cumulative\ probability\ of\ 'i')$$

**Illustration: for a typical single variable problem, maximize F(x).**

| x (population) | F(x) |
|---|---|
| x1= 1.0 | -10 |
| x2 =0.6 | 3 |
| x3= 2 | 5 |
| x4=5 | 14 |
| x5=4 | 12 |
| x6=8 | 13 |
| x7=3.5 | 7 |
| x8=9 | 11 |
| x9=3 | 6 |
| x10=10 | 4 |

**Normalized F(x)**

| F(x)+10 |
|---|
| 0 |
| 13 |
| 15 |
| 24 |
| 22 |
| 23 |
| 17 |
| 21 |
| 16 |
| 14 |

**Ranked**

| x | F(x)+10 |
|---|---|
| x4 | 24 |
| x6 | 23 |
| x5 | 22 |
| x9 | 21 |
| x7 | 17 |

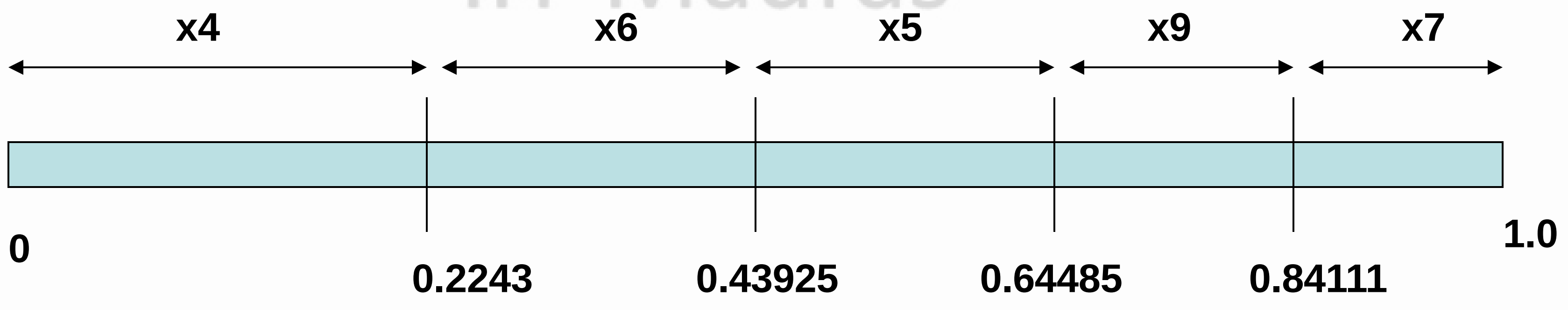
**Crossover Pool of 50%**
**OR**
**Crossover Rate of 50%**

**Worst 50% rejected**

## Best Parents Ranked

| x | Norm. Fitness | Prob.(*p*) | Cum. Prob.(*q*= ***Σp***) |
|---|---|---|---|
| x4 | 24 | 0.22430=**24/107 etc.** | 0.2243 |
| x6 | 23 | 0.21495 | 0.43925 |
| x5 | 22 | 0.2056 | 0.64485 |
| x9 | 21 | 0.19626 | 0.84111 |
| x7 | 17 | 0.15888 | 1.00000 |

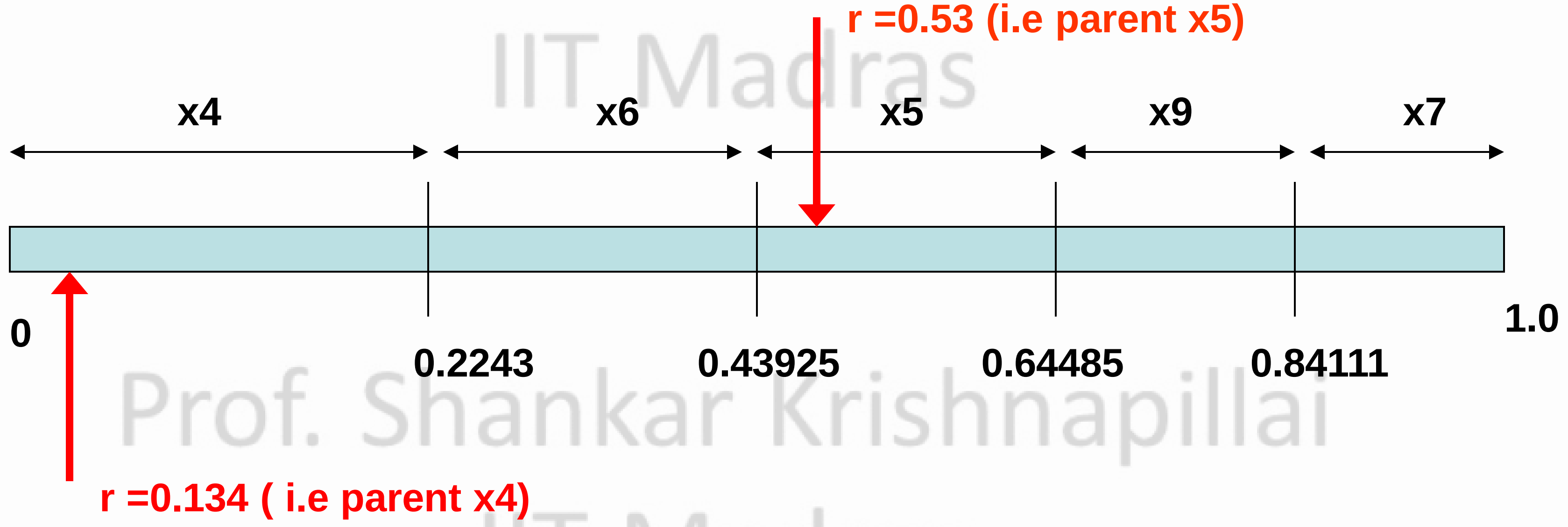
**Σ Fitness= 107**



Cumulative probability scale

Suppose we want to select 2 parents for crossover:

1st , Select one parent by generating a random number r between 0 and 1. Say r =0.53



2nd, Select another parent using another random number, say r = 0.134

Now the two parents are x4 and x5; and they give 2 offsprings x4’ and x5’.

We now have to specify how many "cross-over" operations we want per generation i.e., *N.cross*.

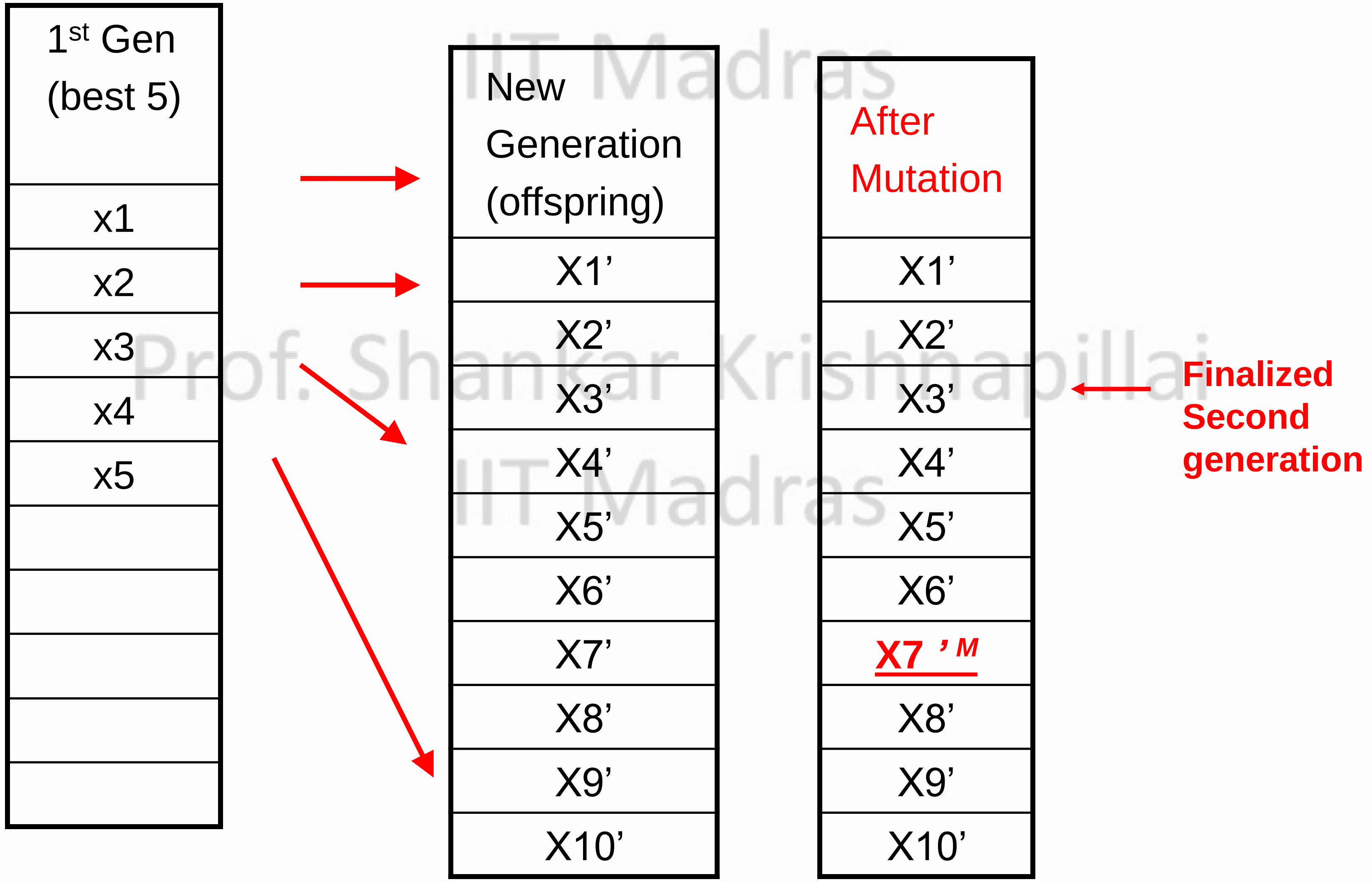
(No. of offspring = *N.cross* x 2, must not exceed the population size *Npop*)

In this example, we can have maximum of 5 cross-over operations per generation. This will give 10 offspring which will completely fill the next generation.

OR

We can have 4 cross-over operations.
This will give 8 offspring which will go into the next generation.
The remaining 2 have to be picked up from the 5 parents (preferably chosen by roulette wheel).

## Mutation also has to be done before Finalizing the 2nd generation (at least one member is randomly chosen and replaced by a new number)



## Elitism Vs Diversity: *These are opposing forces in GA*

**Diversity means encouraging the selection of diverse members across the entire variable range. Random operations like Roulette wheel, Mutation encourages diversity. It prevents GA from being stuck in a local optima.**

**Elitism means showing preference to a top ranked member in the population. In some cases it can lead to quick convergence, in other cases it spoils diversity.**

## Crossover schemes for Floating Point GA

**Arithmetic crossover (Linear Interpolation):**
**Say the two parents are x1 and x2.**

**Offspring 1 = β x1 + (1- β)x2.** **(β is a random number between 0 and 1)**
**Offspring 2 = (1-β) x1 + β x2.**

**Heuristic Crossover (Extrapolation & Interpolation):**

**Offspring 1 = β (x1-x2) + x1.**
**Offspring 2 = x2 + β (x2-x1).**

**If β is a positive random number between 0 and 1:**
**(eg. x1=2; x2=10; β =0.2. Offsprings are 0.4 and 11.6 )**
**The offsprings are extrapolated.**

**If β is a negative random number between 0 and -1, it will interpolate.**

**We have to obtain both interpolated and extrapolated offsprings (4 nos) and randomly pick two out of them.**

## Selecting Typical GA parameters

**Population size**: In the order of 50-500 (depend on difficulty of Obj. function)

(Larger population means more efficient sampling of search space, but more computational effort)

**Crossover Rate**: This is the percentage of parents selected for
**(or Crossover Pool)** cross-over pool. Usually 40-60% of *Npop*.

(Parents will be selected from this pool using the Roulette Wheel)

**Number of cross-overs**: User has to specify how many cross-over operations to conduct from the parent pool.

Maximum: Npop / 2.

**Mutation rate:** 0.01 to 0.001 for Binary GA.
upto 10% for Continous GA.

## GA for multi-variable problems i.e., f(x,y)

(*Binary GA becomes difficult for multiple variables, as bit string becomes large*)

## Continuous (Real coded) GA for f(x,y)

**Floating point accuracy is directly set to required value ($10^{-8}$ say ) in Matlab.**

**A population (x1,y1), (x2,y2)… is created and ranked according to f(x,y).**

**If we want to cross-over members (x1,y1) and (x2,y2):**

**x1 and x2 are crossed over. They give offspring x1’ and x1’.**

**y1 and y2 are crossed over to get y1’ and y2’ (use different random numbers for x and y).**

**Thus new offspring (x1’,y1’) and (x2’,y2’) are got.**

| Pop. No | Population | F(x,y) ranked |
|---|---|---|
| 1 | $(x_1,y_1)$ | 120 |
| 2 | $(x_2,y_2)$ | 75 |
| .. | | .. |
| 10 | $(x_{10},y_{10})$ | 10 |

## Some Applications in GA
## (real coded GA)

**To solve a GA problem, the following parameters must be supplied:**

**1. Floating Point Resolution (eps command) & Variable bounds.**

**2. Population size**

**3. No. of Mutations in the population**

**4. Crossover rate (40-60% of *Npop*) *i.e* pool for crossover**

**5. Number of crossovers**

**6. Type of crossover (Arithmetic, Heuristic etc)**

**7. No. of iterations to run the GA.**

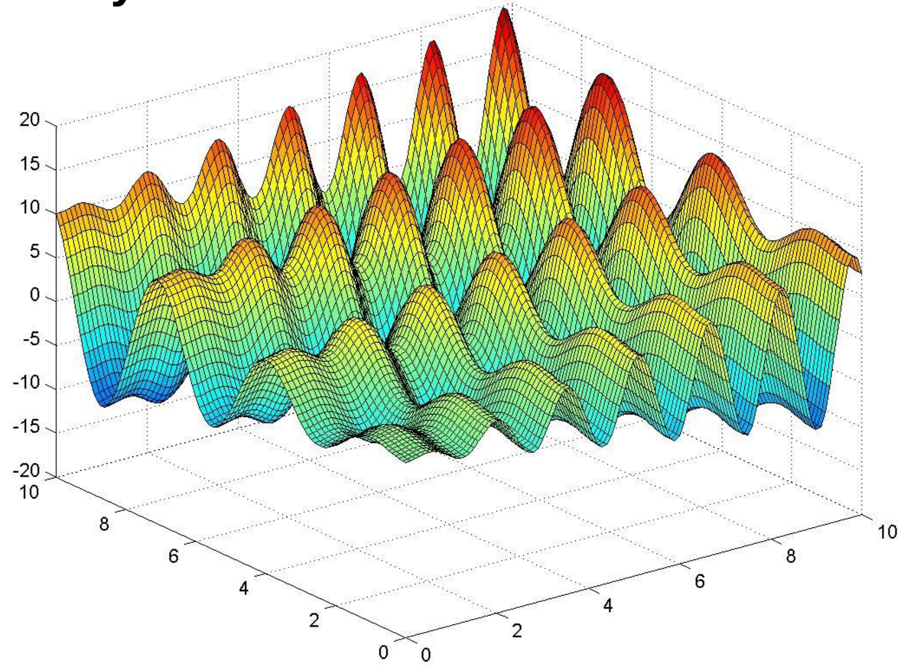
**GAOT - A free ware Matlab GA code was used for these examples.**
**It uses Real Coded GA.**

## Some sample problems solved by GAOT

$Maximize \quad x\sin(4x) + 1.1y\sin(2y)$

$within\ variable\ bounds: \quad 0 \le (x, y) \le 10.$

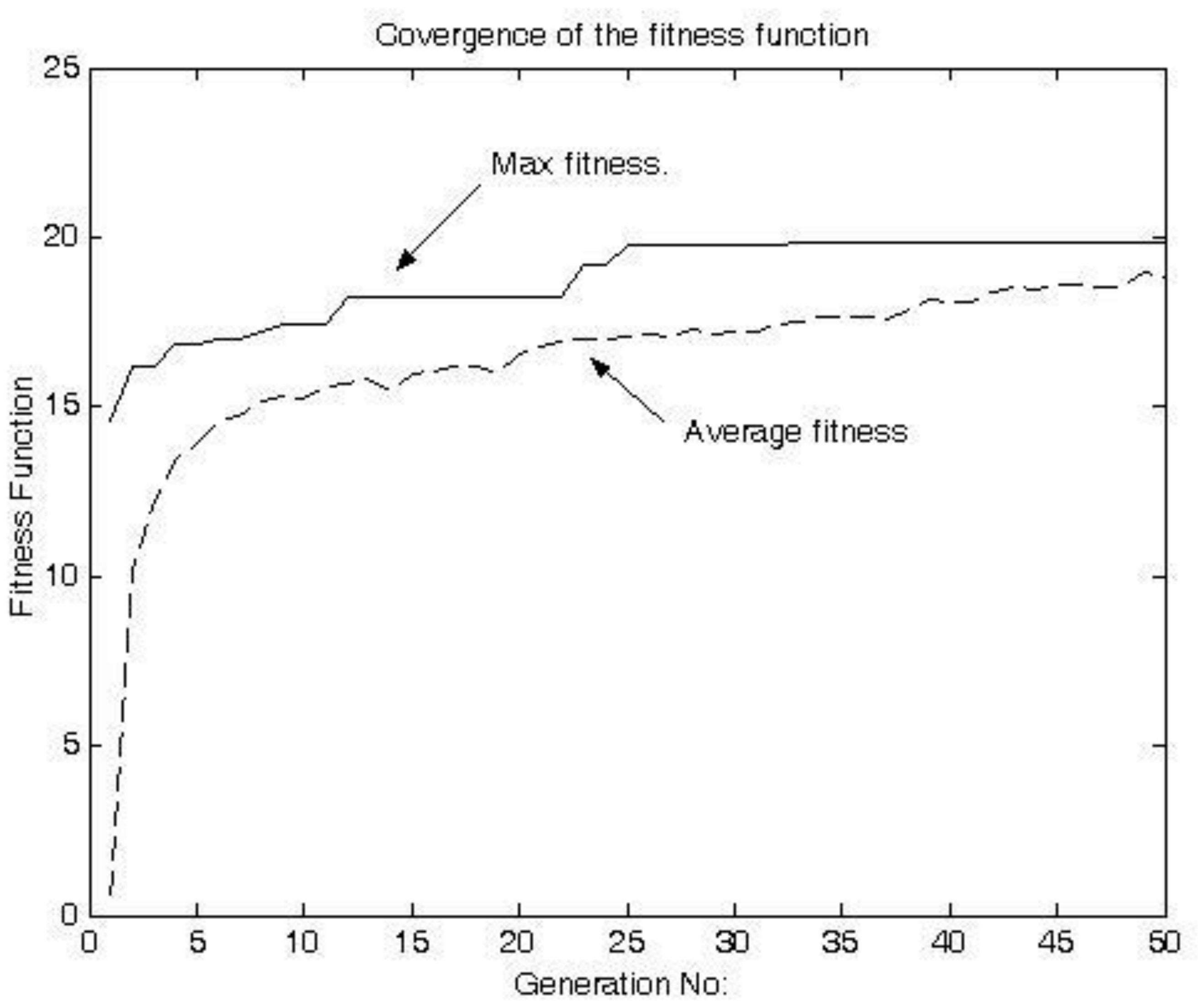
$Resolution: 10^{-8}.$



1. **Variable bounds of x and y are set to 0-10.**
2. **Population size=100.**
3. **Crossover rate of 40% of *Npop* (i.e parent pool for crossover)**
4. **Number of crossovers=50**
5. **Create a subroutine which returns "x sin(4x)+1.1ysin(2y)" .**
6. **Set 100 iterations to termination.**

**Solution is (x,y)= (9.82 , 10) *correct***

**and f(x,y)=20 (maxima)**



**Average fitness of the population must improve over the generations.**

**The member with maximum fitness in the final population is the solution.**

## Multi-Objective Genetic Algorithms

1. Multi-Objective GA's are sometimes preferred to Analytical methods.

2. They are general purpose methods – applicable to linear as well as non-linear problems.

3. Unlike the 'method of constraints' or 'weighted sum' method, there is no need for the user to know the internals of the problem.

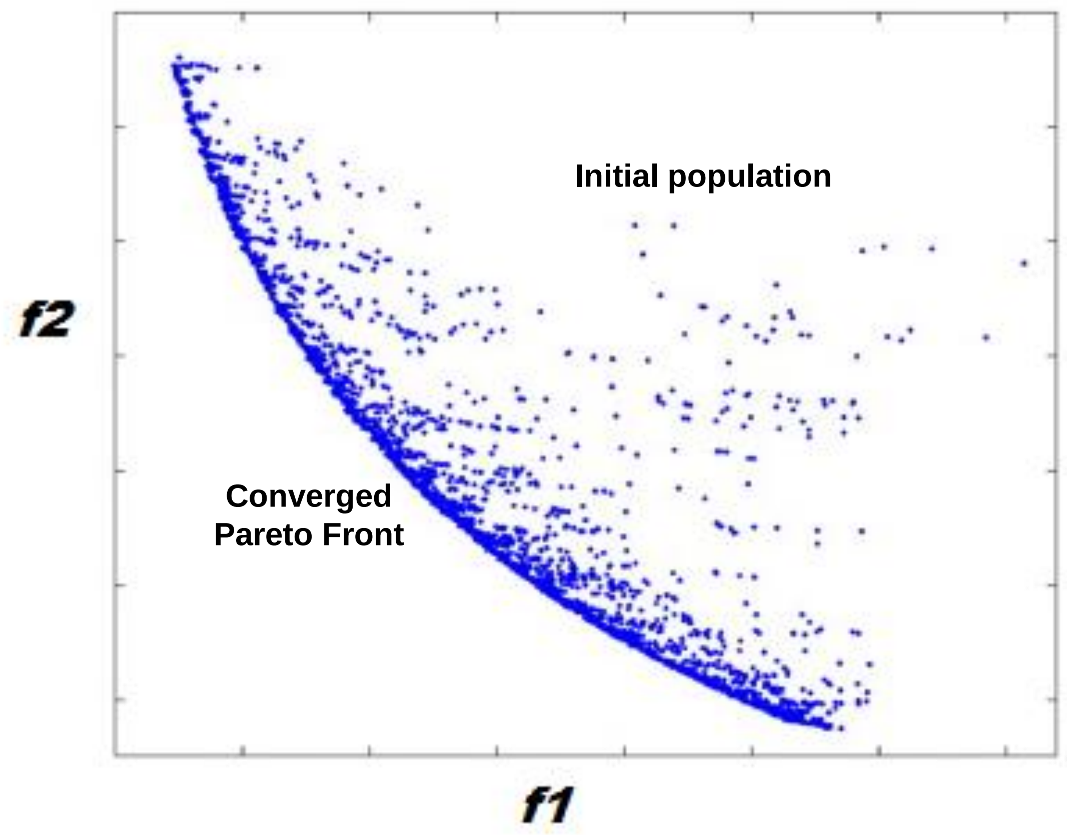
A sample of the existing GA's used for multi-objective optimization is presented here.

For detailed algorithms, Refer:

"Multi-Objective optimization and Evolutionary Algorithms"
*by Kalyanmoy Deb* – Wiley Ed.

Pr

Pr



**Typical convergence to Pareto Front using Genetic Algorithms**

## Vector Evaluated GA (VEGA)

It is one of the earliest Multi-Objective GA's.

It is a straightforward extension of single Objective GA to Multi-objective GA.

A random population of N is created in the Design space (i.e (x,y) points).

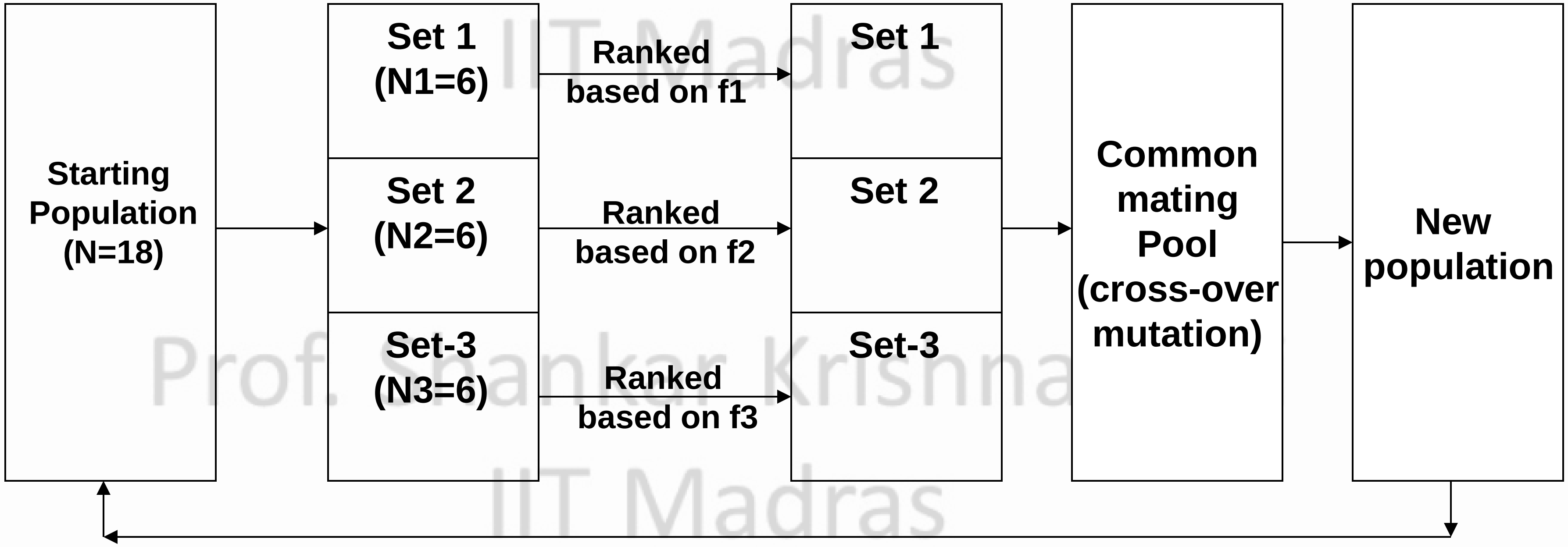
If there are M objective functions, the population is divided into M equal groups randomly.
In each group the members are ranked according to the fitness of that Objective function only.

The best (fittest) members in each group are put into a common 'mating pool'.

From this mating pool the crossover and mutation are done to create a final population of N.
This is the next generation.

Suppose 3 Objective functions $f_1$, $f_2$ ,$f_3$ , and total population is N=18.

Then we have 3 sets each made of 6 members, randomly assigned.



Members in Set-1 are ranked according to fitness of f1 (similarly for Set-2 and Set-3, w.r to f2 and f3).

All members in the common mating pool are considered equally fit.

**We have to specify the population contribution to the mating pool, say 60% of each set (i.e 4 out of 6 will be selected to mating pool).**

| *Set-1* | |
|---|---|
| *(x,y)* | *f1* |
| (x1,y1) | 100 |
| (x7,y7) | 75 |
| (x3,y3) | 50 |
| (x10,y10) | 40 |
| (x14,y14) | 37 |
| (x9,y9) | 23 |

**Ranked based on f1**

**4 best members are selected from Set-1 for the common pool .**

**(x1,y1)**
**(x7,y7)**
**(x3,y3)**
**(x10,y10)**

**The common mating pool will have total 12 members (i.e., 4 from each set).**

**All these members are considered equally fit, and crossover and mutation is done by randomly picking a parent with uniform probability.**

**The results of the crossovers give the next generation of 18 members.**

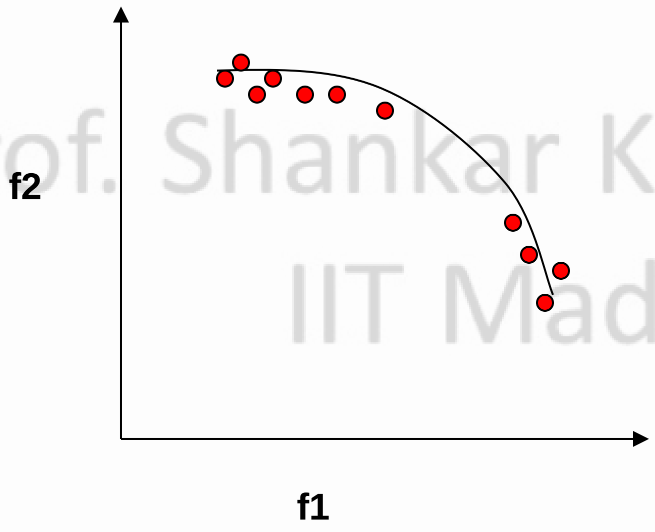
**These 18 members are again grouped into three sets, and next iteration starts.**

Comments on VEGA:

1. Vega is simple and easy to program.

2. It works reasonably well and gives the Pareto Front.

3. Vega emphasises solutions which are good for each individual Objective function. Thus the Pareto Front points obtained by Vega contains more points to the extremes than in the intermediate region.



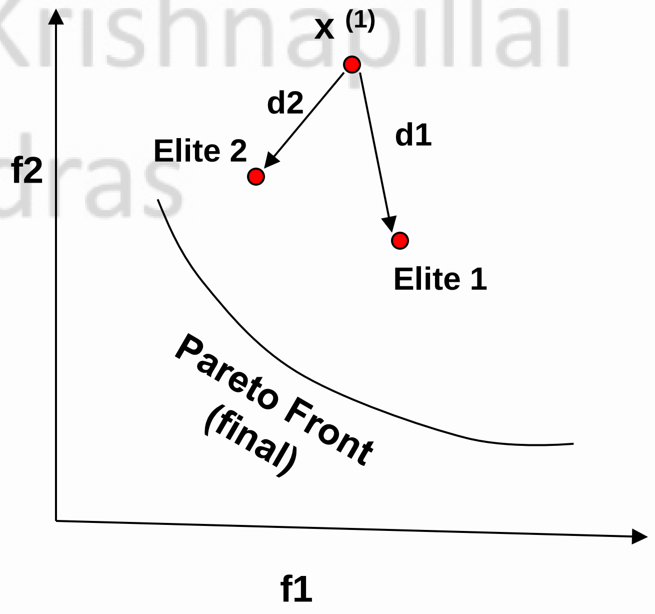
# Distance based pareto front GA -DPGA (Elitist)

In this method, the 'fitness' of each population member is represented by a single number which is a measure of its minimum 'distance' from the ELITE members of the population.

This distance is simply defined as the RMS of the difference of objective funtion values between that member and each 'elite' member.

For each member calculate the distance from all elite members (say $d_1$,$d_2$)

Fitness of $x^{(1)}$ is $\min(d_1, d_2)$

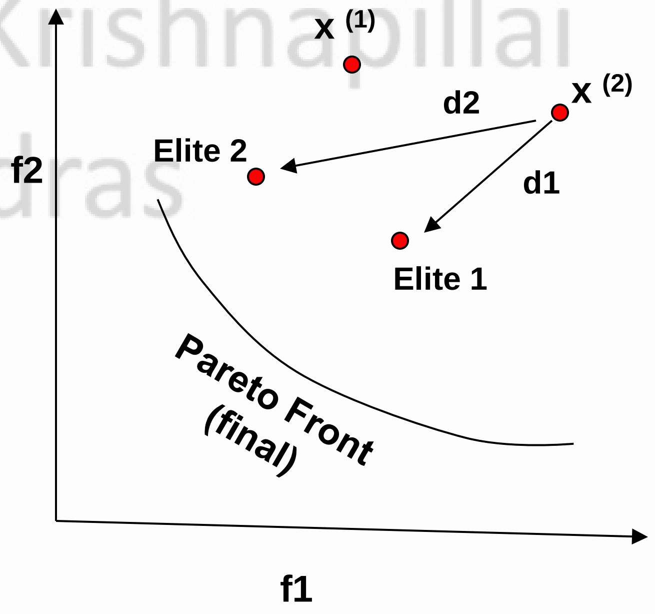


## Distance based pareto front GA -DPGA (Elitist)

**Similarly we can take another population member $x^{(2)}$ and find its minimum distance from all the elite members.**

**$x^{(2)}$ is superior to $x^{(1)}$ if it has a smaller minimum distance.**

**This will affect the ranking and chances of selection between $x^{(2)}$ and $x^{(1)}$.**



Advantages of DPGA:

It correctly finds the Pareto front and the curve is evenly spread out.

Disadvantages:

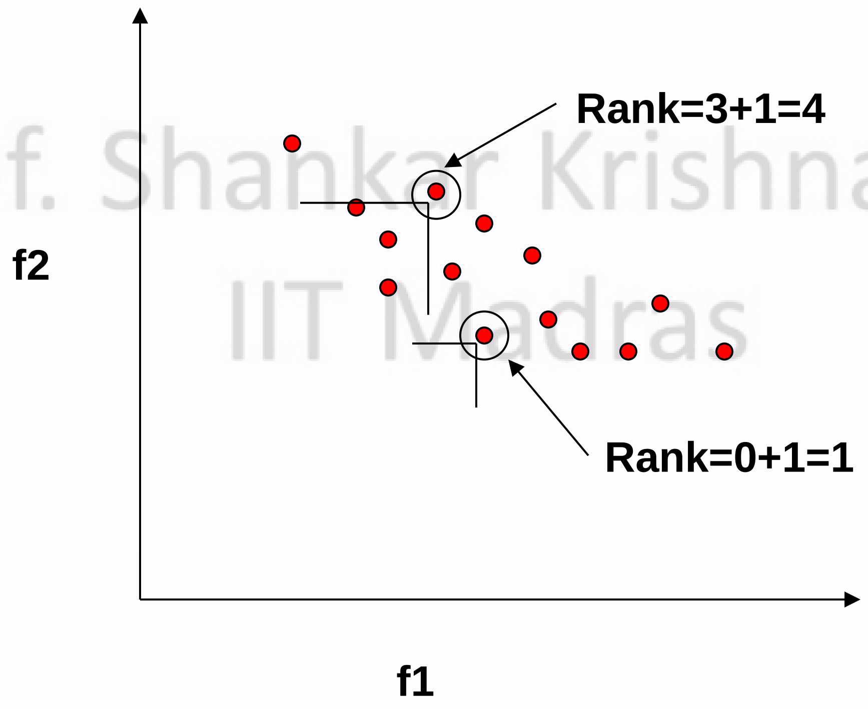
Depending on the initial elite members, the convergence may be fast or slow.

The computational complexity increases with generations
(i.e. comparing minimum distance from many elite points)

## Non-Dominated Sorting GA - NSGA 2 (Elitist)

It uses a simple but effective Pareto ranking concept to find the 'fitness' of a population member.

The entire population is sorted into ranks using the equation
Rank of member $i$ = No. of dominating members +1

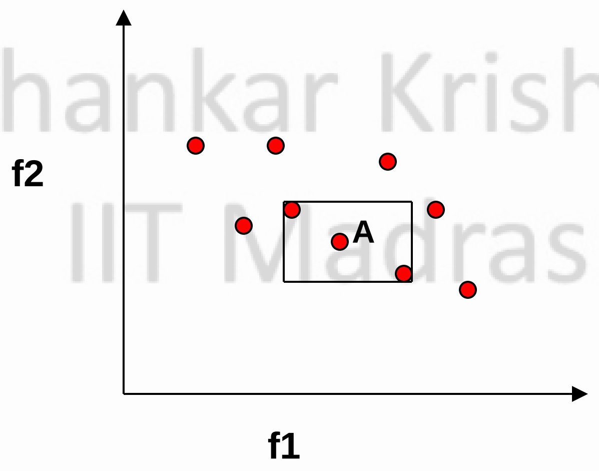


All members in the same rank have the same fitness.

Lower the rank, higher the fitness:   Rank 1 has best fitness...etc.,

Crowding Distance:

An important feature of NSGA-2 is the crowding distance operator, which encourages 'diversity' among  the population. i.e NSGA-2 discourages members which are crowded close together.



**Various measures can be used to evaluate the crowding distance of a point A.**

**The semi-perimeter of the rectangle contaning the nearest points to A is a good crowding estimate. Larger the semi-perimeter, less crowding.**

## Crossover Operations in NSGA

Tournament selection:

NSGA-2 uses a "*tournament selection*" scheme to find the fittest parents for crossover.

(*Tournament Selection is used when we have to satisfy multiple criteria*.)

How to Choose a parent:

Two members $i$ and $j$ are randomly selected (with equal probability) from the pool of parents awaiting crossover.

If $i$ has a better Pareto rank than $j$, it is preferred.
If their ranks are equal, then the member with better crowding distance is preferred.

Another parent is selected in the same manner and crossover is conducted.

## NSGA-2 for Constrained problems:

Instead of using Penalty functions, the concept of *Constraint Domination* is used in Tournament Selection.

Out of two potential parents $i$ and $j$ selected in one Tournament round, $i$ is said to constraint dominate $j$ if:

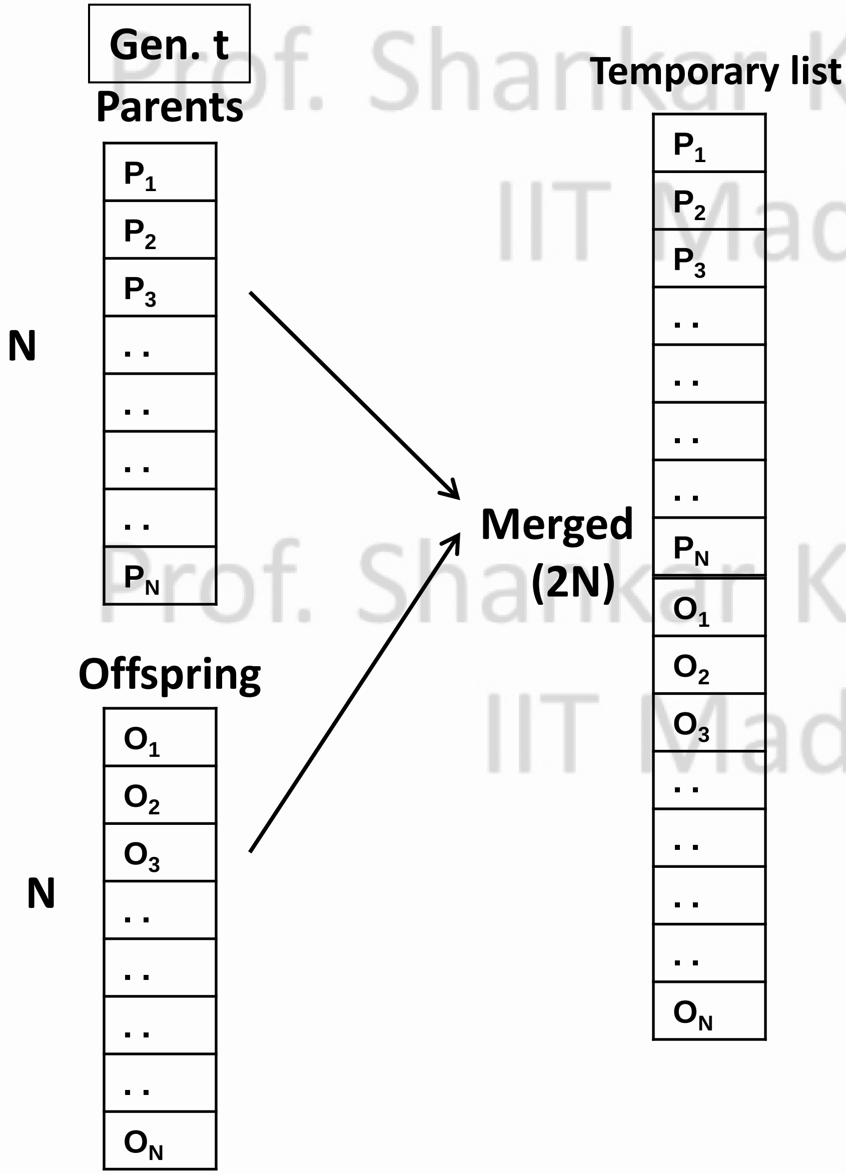
1. $i$ is feasible and $j$ is not.

2. $i$ and $j$ are both infeasible, $i$ has the smaller constraint violation.

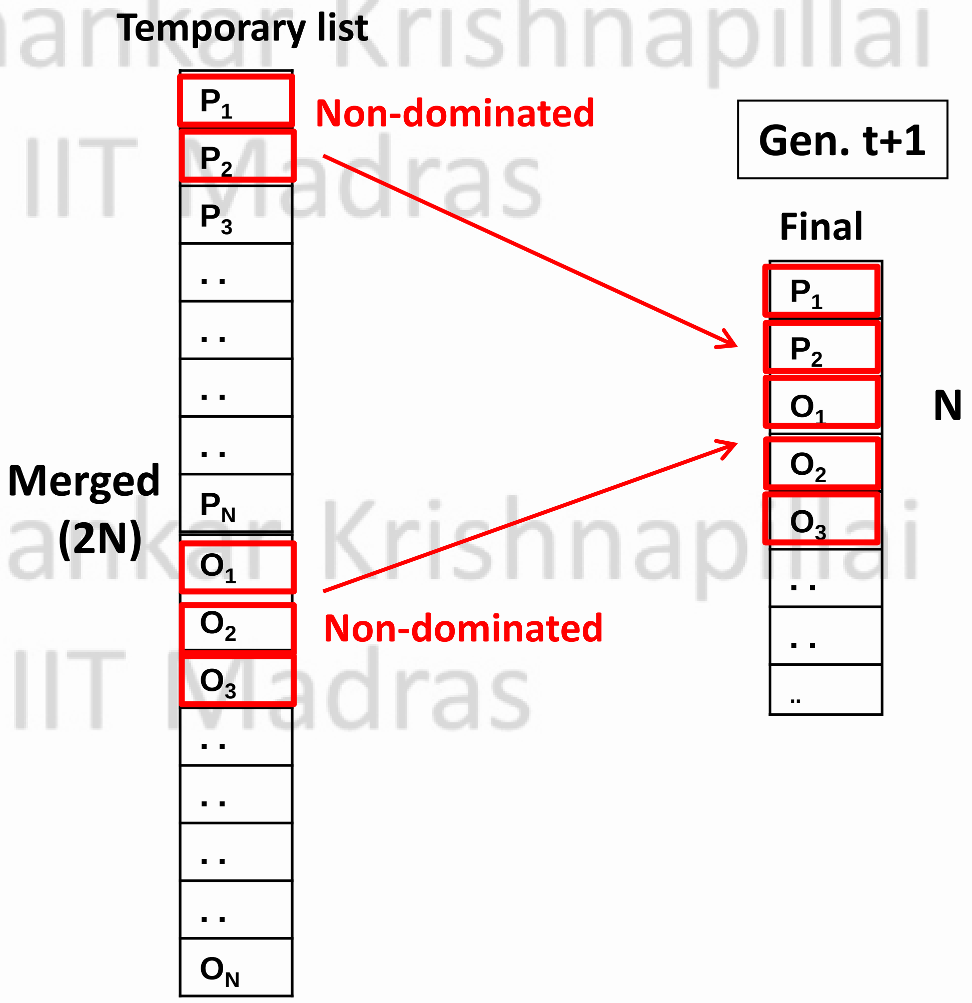
3. $i$ and $j$ are both feasible, $i$ has the superior Pareto Rank.

4. $i$ and $j$ have same Pareto rank, $i$ has better crowding distance.

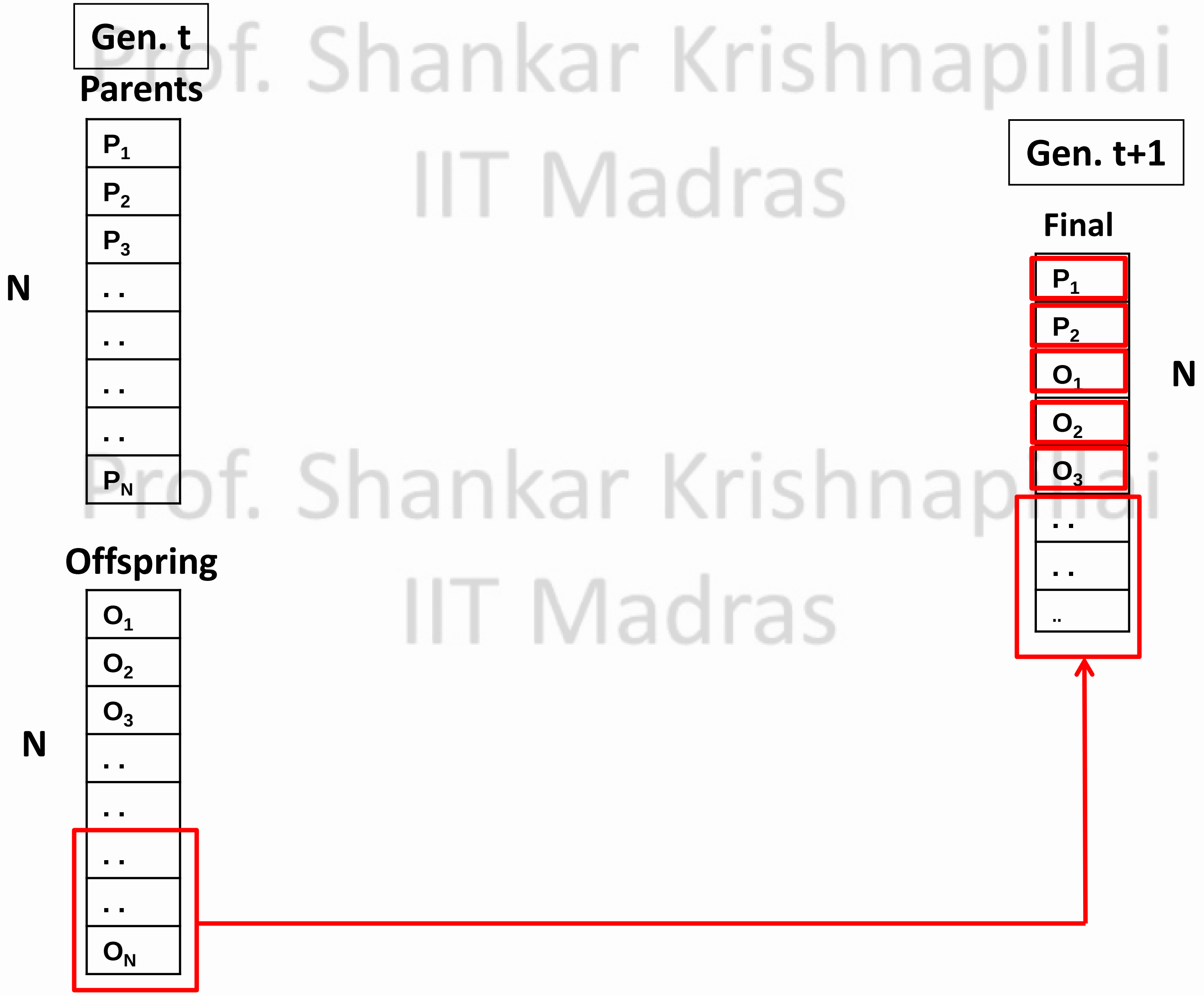
## *NSGA maintains two population groups – parents and offspring*
## *It merges them in a single list*



## Non-dominated members of the combined group are inserted into the next generation

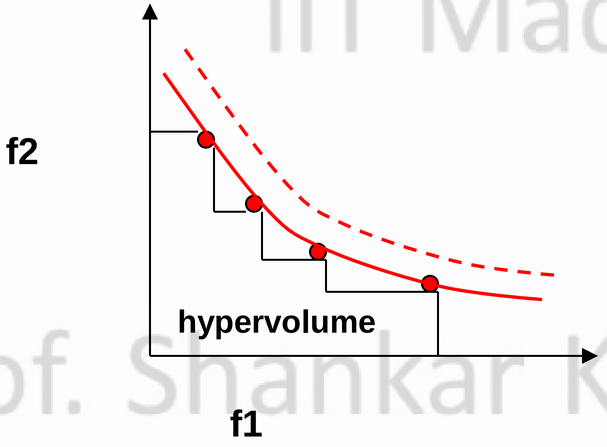


## *Remaining members are taken from offspring*

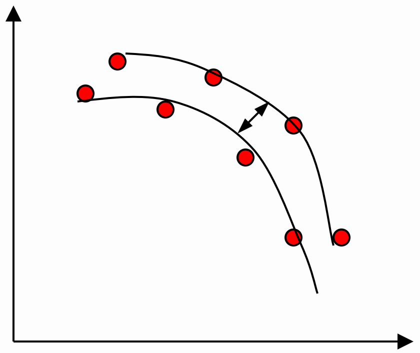


Some estimates of convergence of Multi-Objective GA.

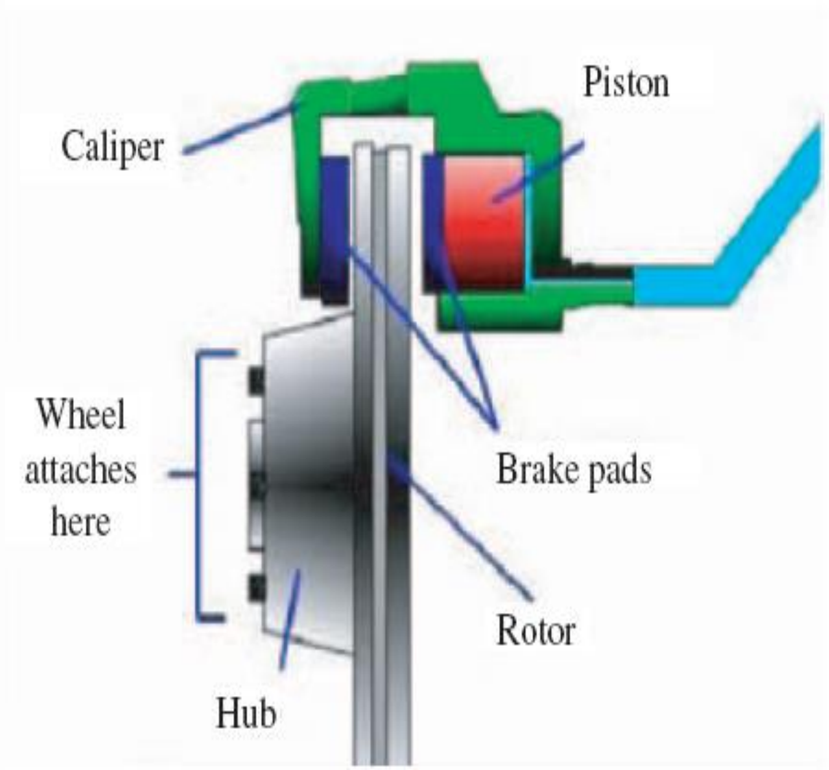
1. Estimate the 'hypervolume' of the points of the Pareto front (1st rank points) in every generation. It is an 'attainment measure' of the Pareto front. The convergence of hypervolume to a steady value is an indication of attaining the Pareto front.

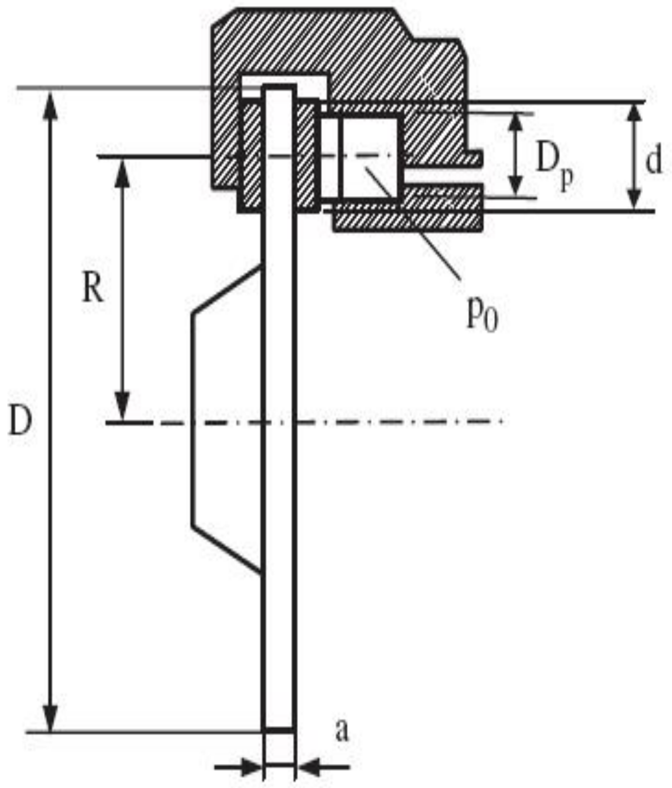


2. We can find the average distance between the Pareto fronts of succesive generations. When the change in average distance becomes lesser than a tolerance value, we can assume that convergence is reached.



## Case study: Multi-Objective Optimization of a Disc brake





**DESIGN VARIABLES**

- **D: outside disc diameter (mm)**
- **R: radius of center line of pad lining (mm)**
- **d: diameter of pad lining (mm)**
- **$D_p$: diameter of piston (mm)**
- **a: thickness of disc (mm)**
- **$p_o$: oil pressure ($N/mm^2$)**

– Objective function -1

(MANUFACTURING COST)

$$C = c_1 \frac{\pi D^2}{4} a + 2c_2 \frac{\pi D^2}{4} + 2c_3 \frac{\pi d^2}{4} h$$

- **c1 = unit cost of disc material**
- **c2 = unit cost of disc surface milling**
- **c3 = unit cost of brake pad lining material**
- **h = thickness of brake pad lining**

***Smaller the brake drum, lesser the manufacturing cost.***

## Objective function -2

STOPPING TIME of the vehicle (T in sec)

(K.E Energy of the car) / 4 = Braking Torque x $\omega_{Avg}$ x T

(Calculated for a 800kg vehicle, moving at 40 kmph, with 4 brakes)

***Smaller the brake drum, larger the stopping distance***

**Stopping time depends on all the design variables:**

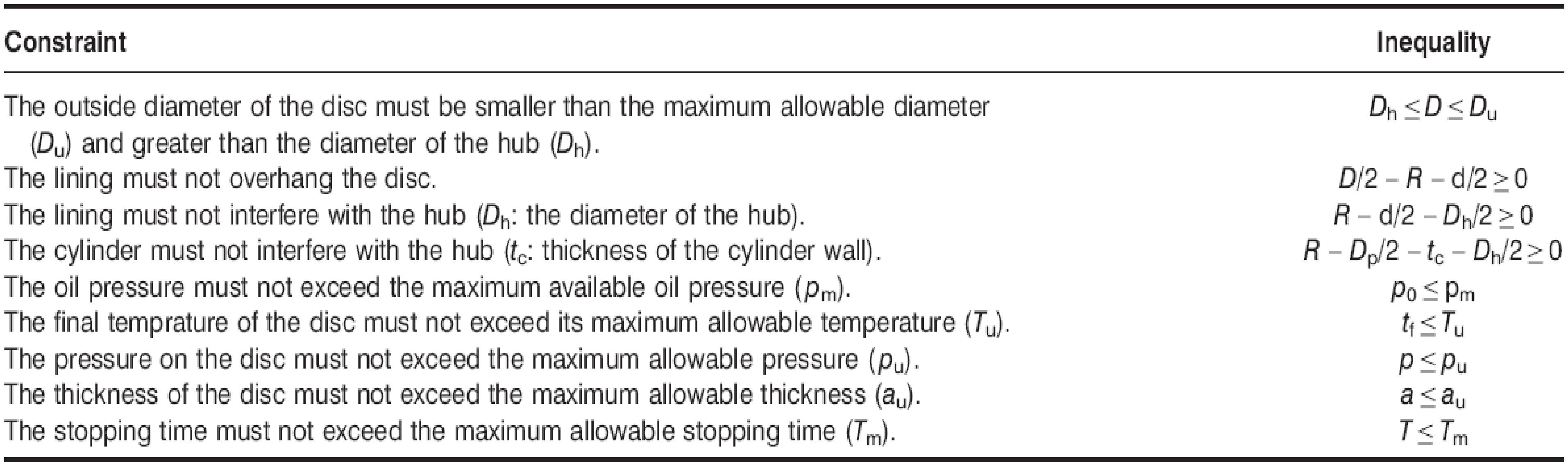
Force exerted by the pad on the disc
(depends on oil pressure ($p_o$) and piston dia)

Initial rpm of the wheel.

Coefficient of friction.

Effective radius of the Friction pad from the brake axis.

## Various design constraints and maximum limits are imposed :



| Constraint | Inequality |
|---|---|
| The outside diameter of the disc must be smaller than the maximum allowable diameter ($D_u$) and greater than the diameter of the hub ($D_h$). | $D_h \leq D \leq D_u$ |
| The lining must not overhang the disc. | $D/2 - R - d/2 \geq 0$ |
| The lining must not interfere with the hub ($D_h$: the diameter of the hub). | $R - d/2 - D_h/2 \geq 0$ |
| The cylinder must not interfere with the hub ($t_c$: thickness of the cylinder wall). | $R - D_p/2 - t_c - D_h/2 \geq 0$ |
| The oil pressure must not exceed the maximum available oil pressure ($p_m$). | $p_0 \leq p_m$ |
| The final temprature of the disc must not exceed its maximum allowable temperature ($T_u$). | $t_f \leq T_u$ |
| The pressure on the disc must not exceed the maximum allowable pressure ($p_u$). | $p \leq p_u$ |
| The thickness of the disc must not exceed the maximum allowable thickness ($a_u$). | $a \leq a_u$ |
| The stopping time must not exceed the maximum allowable stopping time ($T_m$). | $T \leq T_m$ |

**Maximum limits and variable bounds (8):**

| | | |
|---|---|---|
| **$D_u$** | **:** | **Maximum disc diameter (305mm)** |
| **$D_h$** | **:** | **Hub diameter (76 mm)** |
| **$T_u$** | **:** | **Max allowable disc temperature (260$^o$C)** |
| **$P_u$** | **:** | **Max allowable disc pressure (10 N/mm2)** |
| **$P_m$** | **:** | **Max available oil pressure (7 N/mm2)** |
| **$T_c$** | **:** | **Thickness of the cylinder wall (6.5mm)** |
| **$a_u$** | **:** | **Maximum thickness of the disc (25 mm)** |

They are expressed as Mathematical Inequalities and added as Penalty functions to the two Objective functions.

**For Example:**

**Brake pad must not overhang the disc** $\longrightarrow g_2(x) = \frac{D}{2} - R - \frac{d}{2} \geq 0$

**Brake pad must not touch the hub** $\longrightarrow g_3(x) = R - \frac{d}{2} - \frac{D_h}{2} \geq 0$

**$T_{max}$=260$^0$c, $T_{amb}$=25$^0$c**
**c=502 J/kg/$^0$C for steel**
**ρ=7850 kg/m$^3$** $\longrightarrow g_6(x) = T_{\max} - \left( \frac{E}{\pi c \rho D^2 a} + T_{\text{amb}} \right) \geq 0$

9 Penalty functions are added to each Objective Function:

$$f_{1,2} + r \sum_{k=1}^{K} G_k \left[ g_k(x) \right]^2 \qquad (r \text{ is a multiplier evaluated iteratively as } 10^5)$$

$$(G_k \text{ is a heavyside operator}: \; G_k = 0, \; \text{if } \; g_k(x) \geq 0$$
$$G_k = 1, \; \text{if } \; g_k(x) < 0 \;)$$

**Various attempts to find the Pareto Optimal front using Distance based Pareto front GA (DPGA).**

**The algorithm (Matlab programme) is tested using various standard Bench mark functions.**

**Min f1(x) = x1**
**Min f2(x) = (1+x2)/ x1**

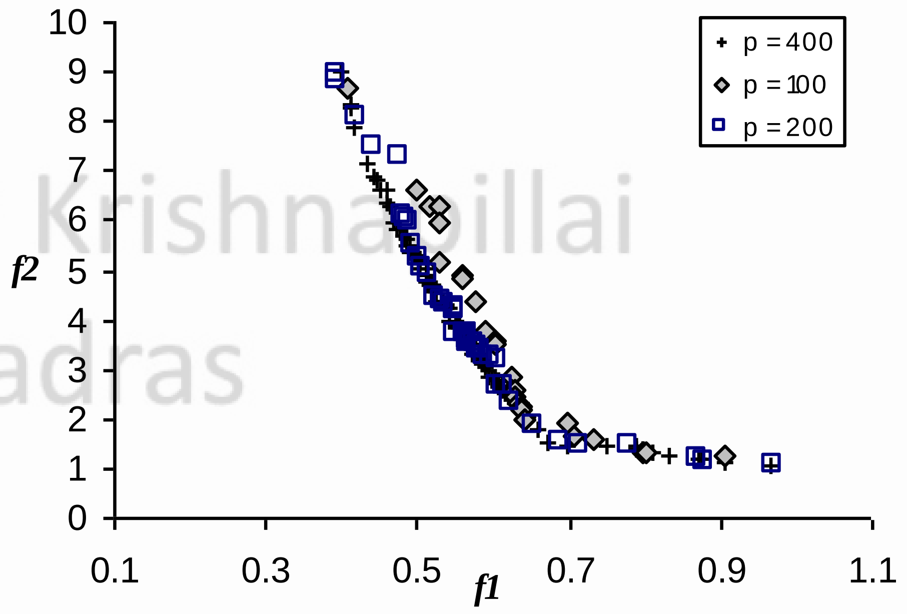
**Subject to,**

**g1(x)= x2+9x1>=6**

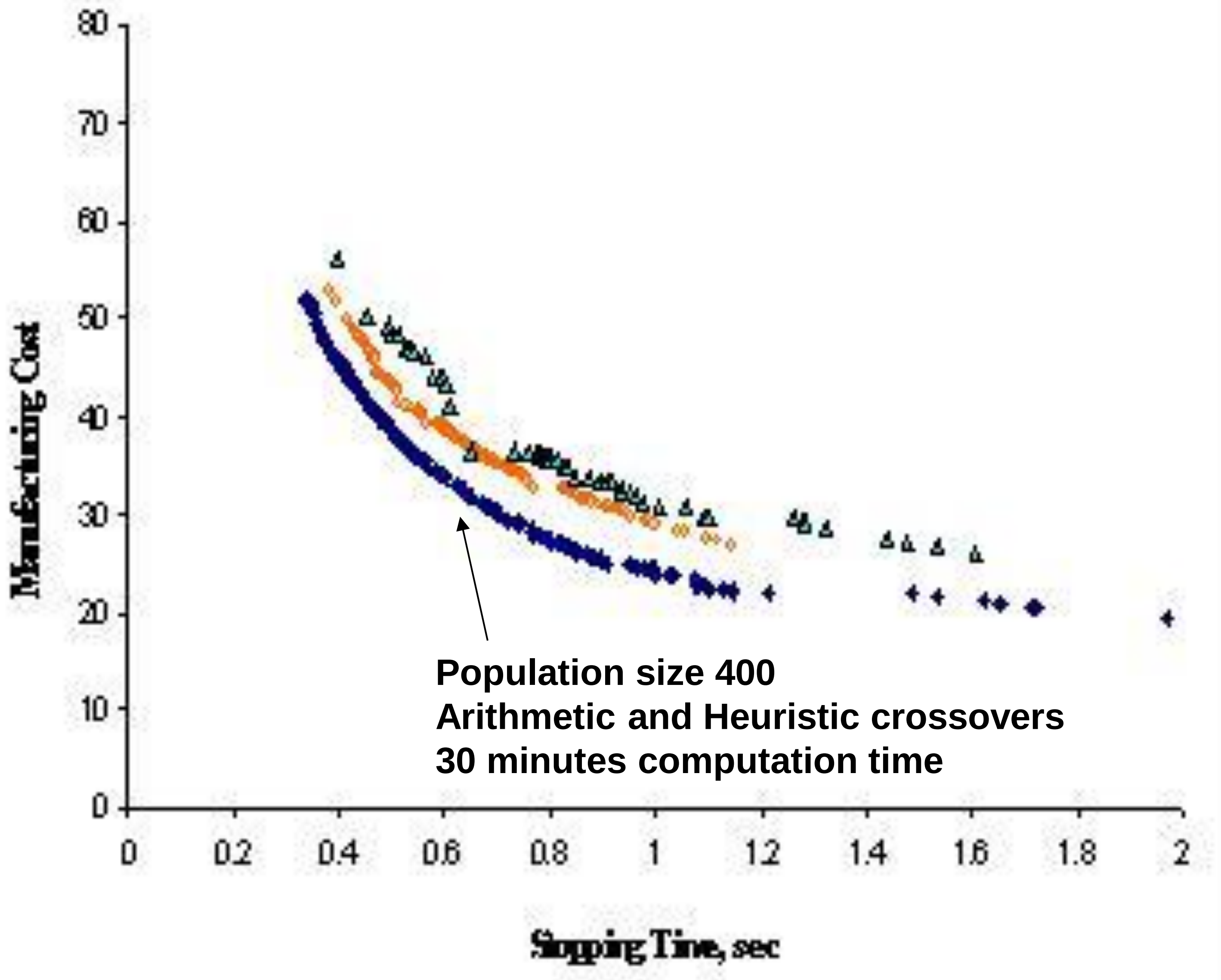
**g2(x)= -x2+9x1>=1**

**0.1<= x1 <= 1;**

**0 <= x2 <= 5;**



## After experimenting on Benchmark problems, it is applied on Disc Brake problem.



**Population size 400**
**Arithmetic and Heuristic crossovers**
**30 minutes computation time**

## Interpreting the Pareto Optimal Front (trade-off solutions)

| D (mm) | R (mm) | d (mm) | $D_p$ (mm) | a (mm) | $p_o$ (N/mm$^2$) | $f_1$ (sec) | $f_2$ (Rs/100) |
|---|---|---|---|---|---|---|---|
| 304.80 | 110.98 | 82.46 | 87.44 | 20.46 | 5.20 | 0.397 | 56.061 |
| 296.60 | 98.63 | 80.40 | 82.34 | 17.08 | 5.80 | 0.454 | 50.248 |
| 292.30 | 98.87 | 78.22 | 79.10 | 16.93 | 5.68 | 0.500 | 48.636 |
| 283.72 | 98.57 | 73.08 | 75.12 | 17.72 | 5.57 | 0.566 | 46.327 |
| 273.79 | 92.19 | 79.67 | 75.76 | 17.55 | 5.52 | 0.605 | 43.362 |
| 259.11 | 83.31 | 84.70 | 77.82 | 13.19 | 5.46 | 0.648 | 36.386 |
| 250.42 | 83.69 | 82.58 | 73.01 | 17.10 | 5.47 | 0.730 | 36.369 |
| 252.83 | 87.95 | 74.46 | 68.76 | 16.17 | 5.46 | 0.776 | 36.156 |
| 248.30 | 84.24 | 75.75 | 70.29 | 16.95 | 5.32 | 0.799 | 35.434 |
| 250.12 | 91.90 | 64.51 | 61.28 | 13.99 | 5.97 | 0.850 | 33.775 |
| 247.15 | 91.41 | 59.66 | 62.58 | 14.95 | 5.37 | 0.909 | 33.416 |
| 240.74 | 84.50 | 64.56 | 63.90 | 15.67 | 5.33 | 0.954 | 32.296 |
| 236.28 | 82.76 | 62.35 | 62.59 | 15.47 | 5.37 | 1.008 | 30.976 |
| 228.22 | 87.62 | 51.53 | 51.29 | 17.01 | 5.91 | 1.279 | 29.429 |
| 215.55 | 84.79 | 45.24 | 44.94 | 17.09 | 6.33 | 1.604 | 26.206 |

END