Genetic Algorithm

Dr. Rajesh Kumar

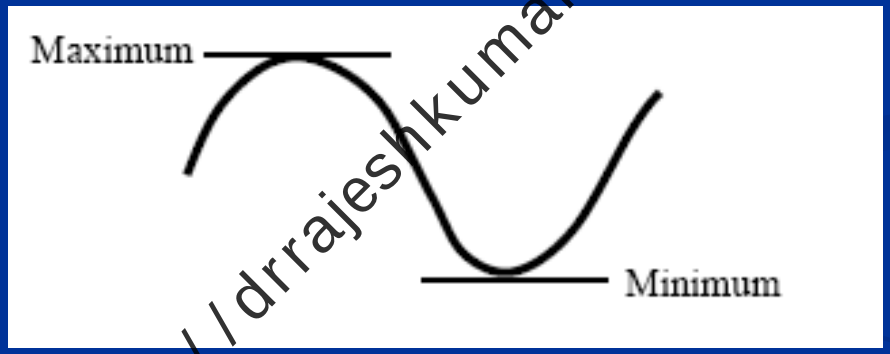
PhD, PDF (NUS, Singapore)
SMIEEE, FIETE, MIE (I),LMCSI, SMIACSIT, LMISTE, MIAENG

Associate Professor, Department of Electrical Engineering
Adjunct Associate Professor, Centre of Energy

Malaviya National Institute of Technology, Jaipur, India, 302017
Tel: (91) 9549654481
rkumar@ieee.org, rkumar.ee@gmail.com
http://drrajeshkumar.wordpress.com/

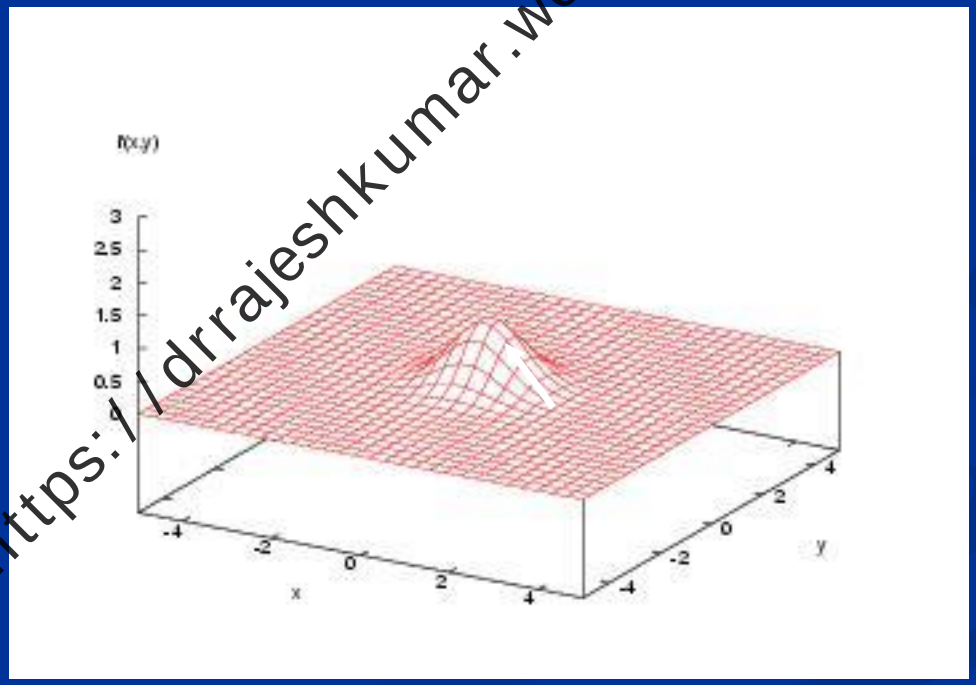
Calculus

- Maximum and minimum of a smooth function is reached at a stationary point where its gradient vanishes



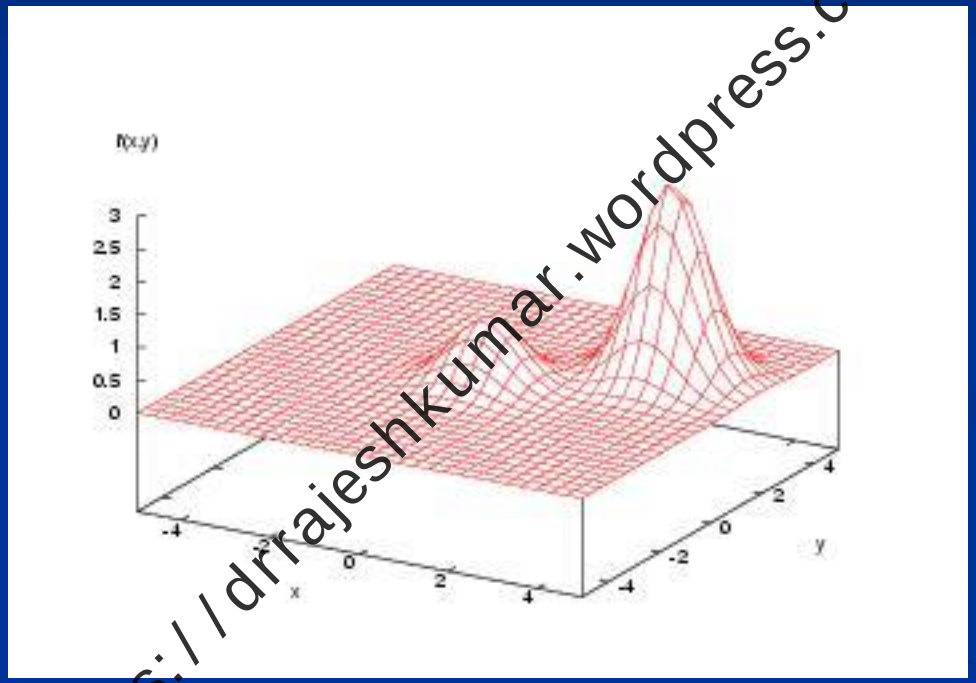
Hill-climbing

Calculus based approach



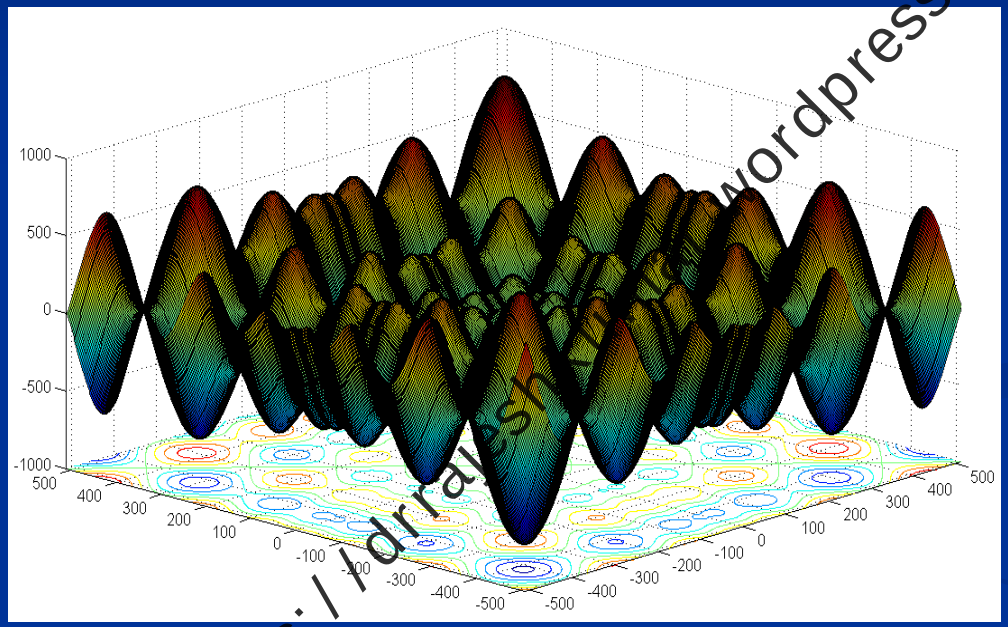
But…

?

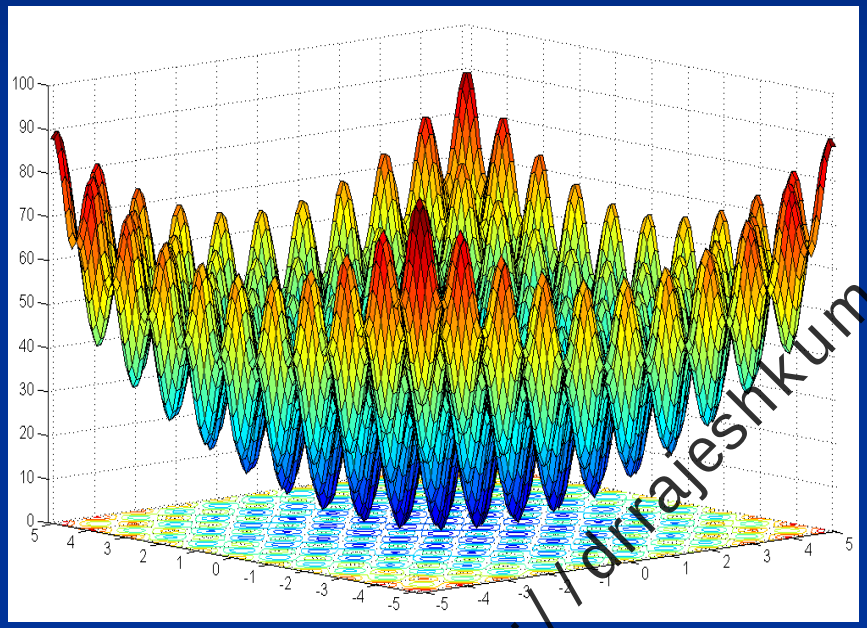


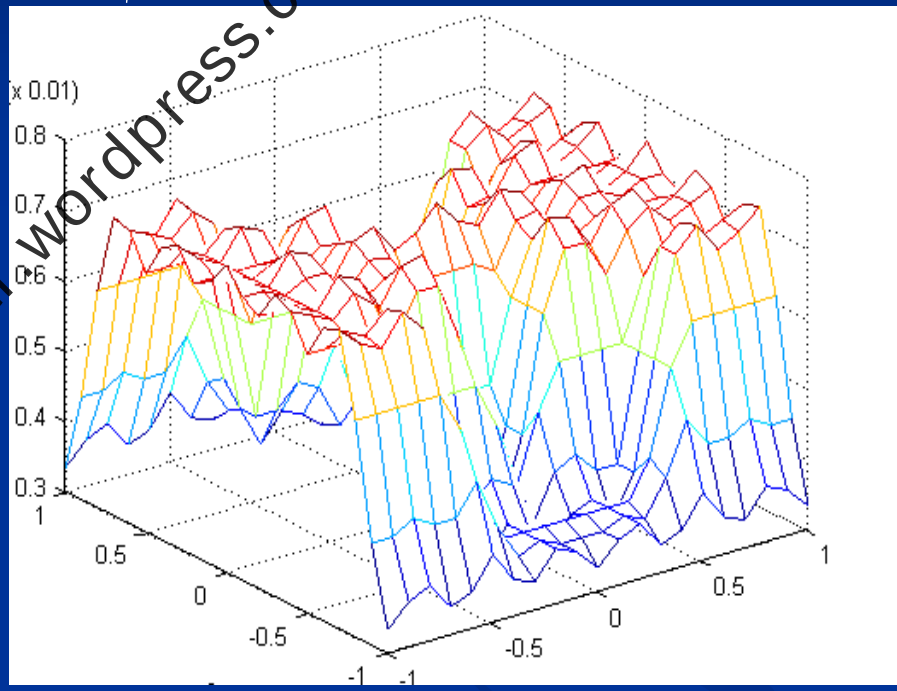
?

But…

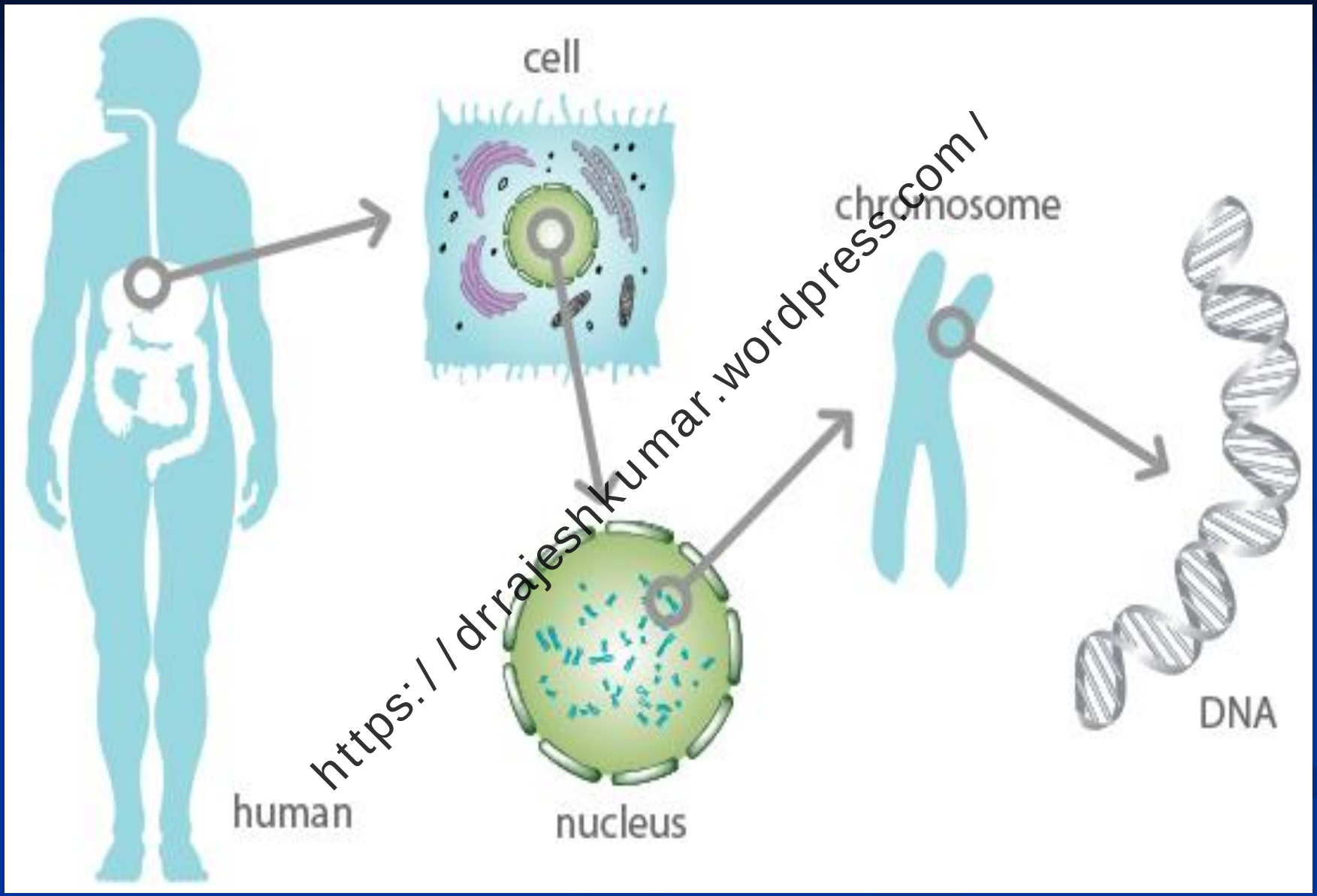


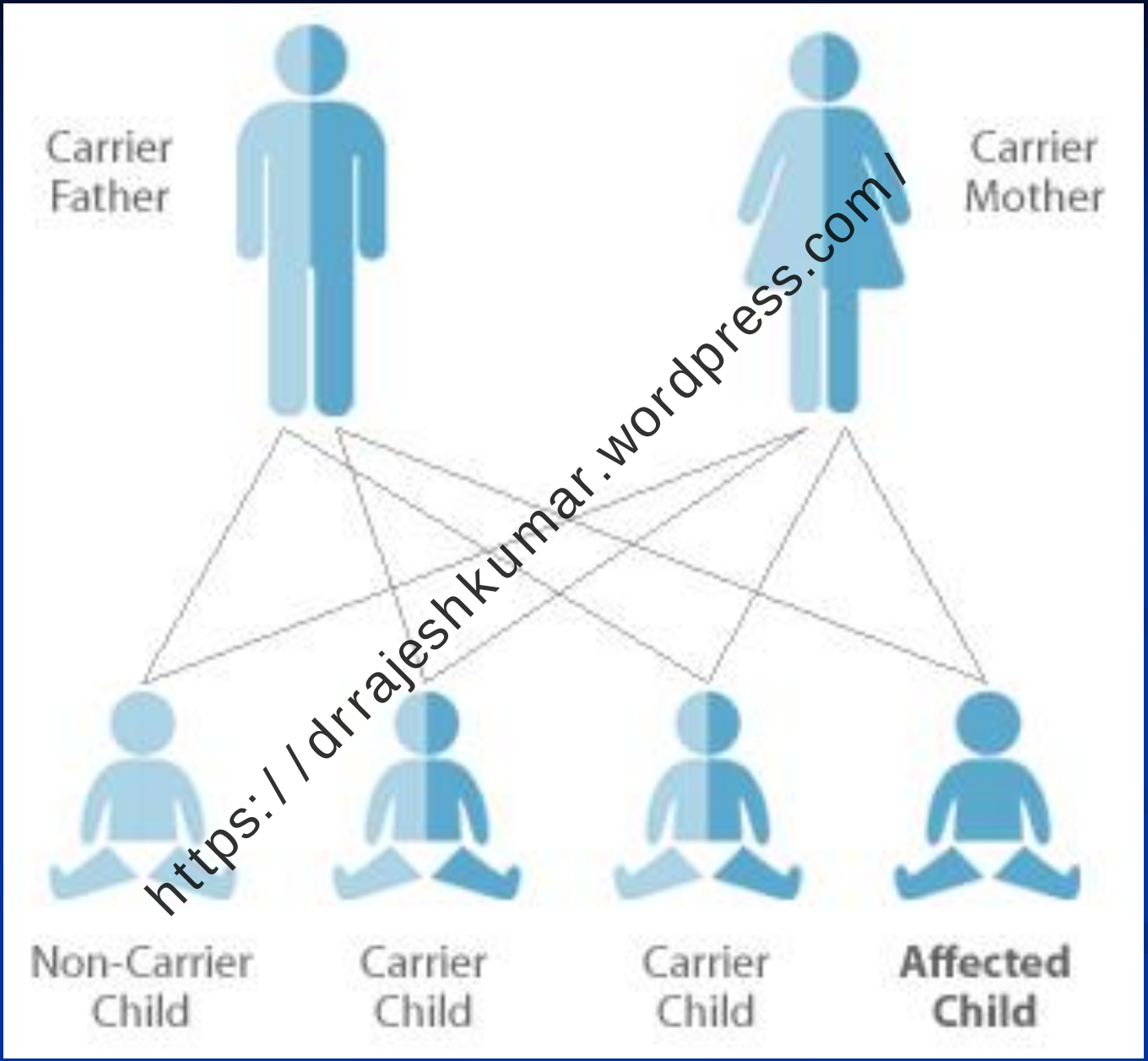
More realistically?





- Noisy?
- Discontinuous?

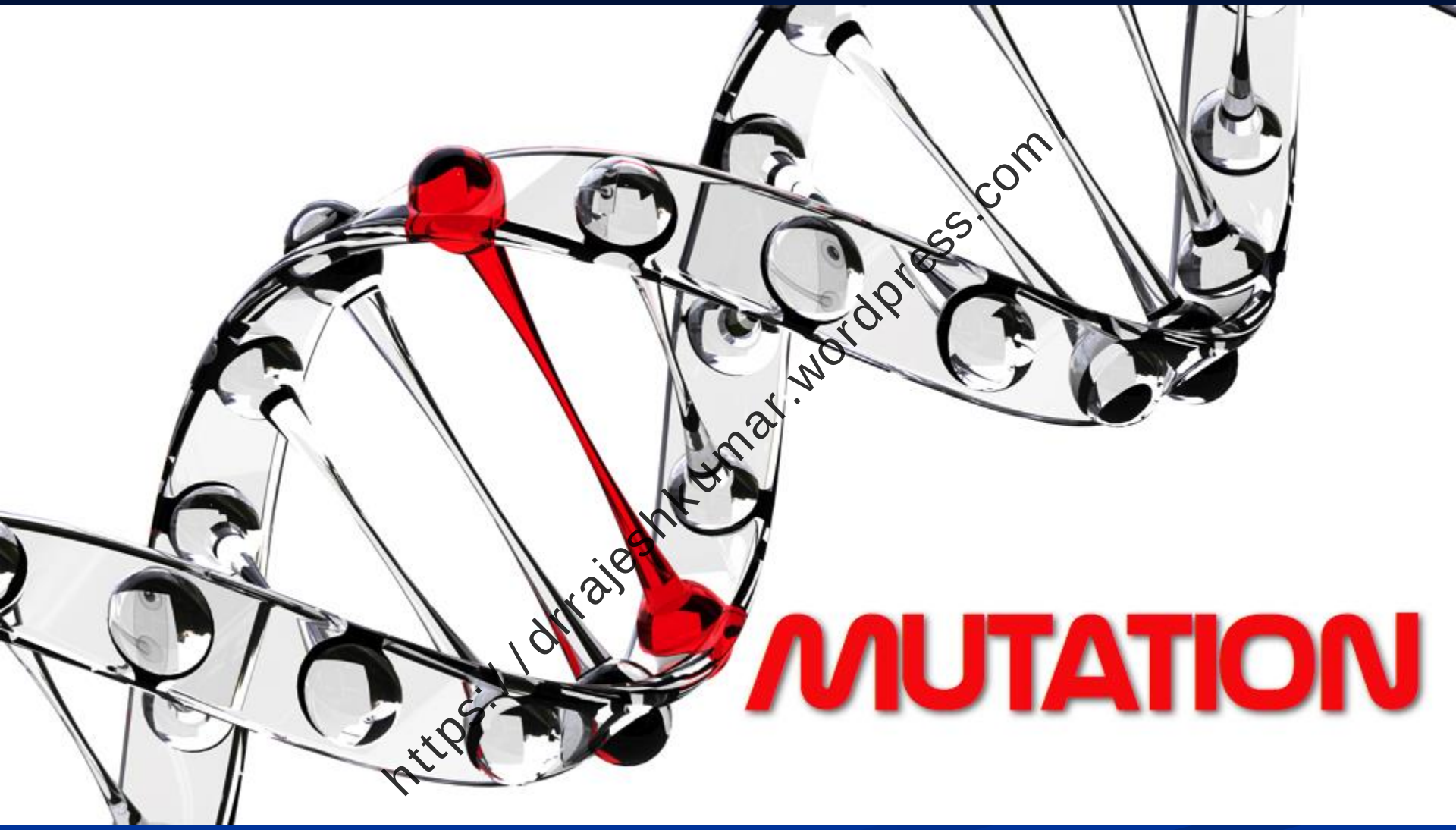

cell
chromosome
DNA
human
nucleus
https://drrajeshkumar.wordpress.com/




https://drrajeshkumar.wordpress.com/

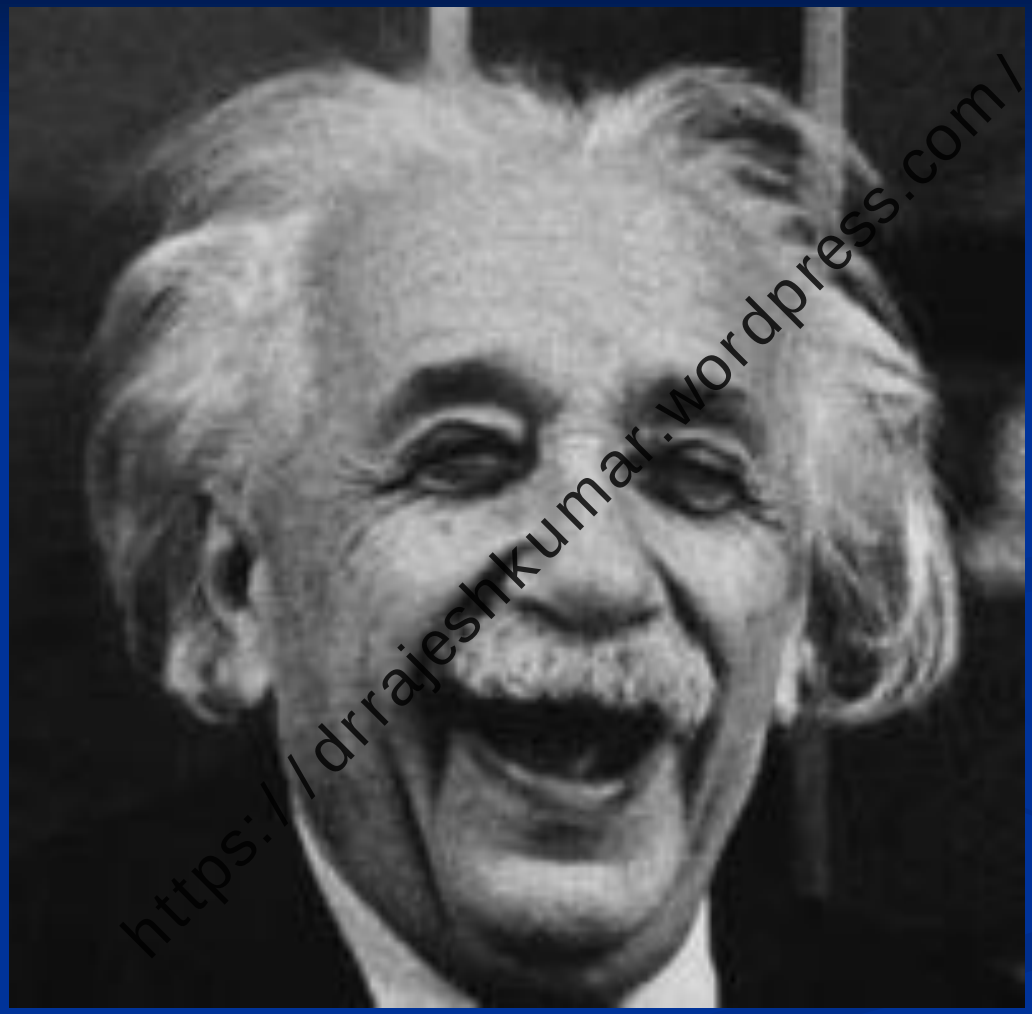

https://drrajeshkumar.wordpress.com/
FunNFun.in

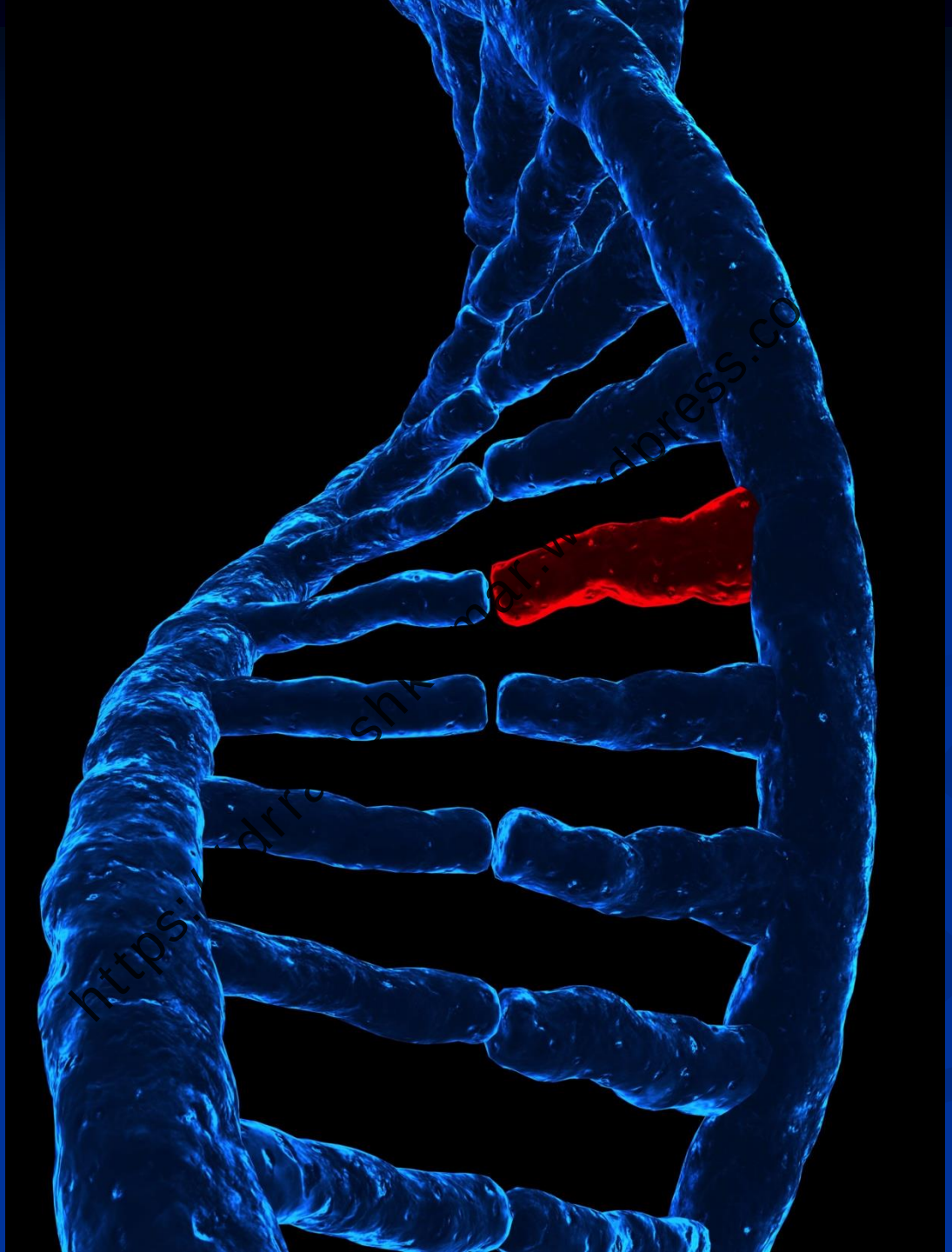

https://drrajeshkumar.wordpress.com/

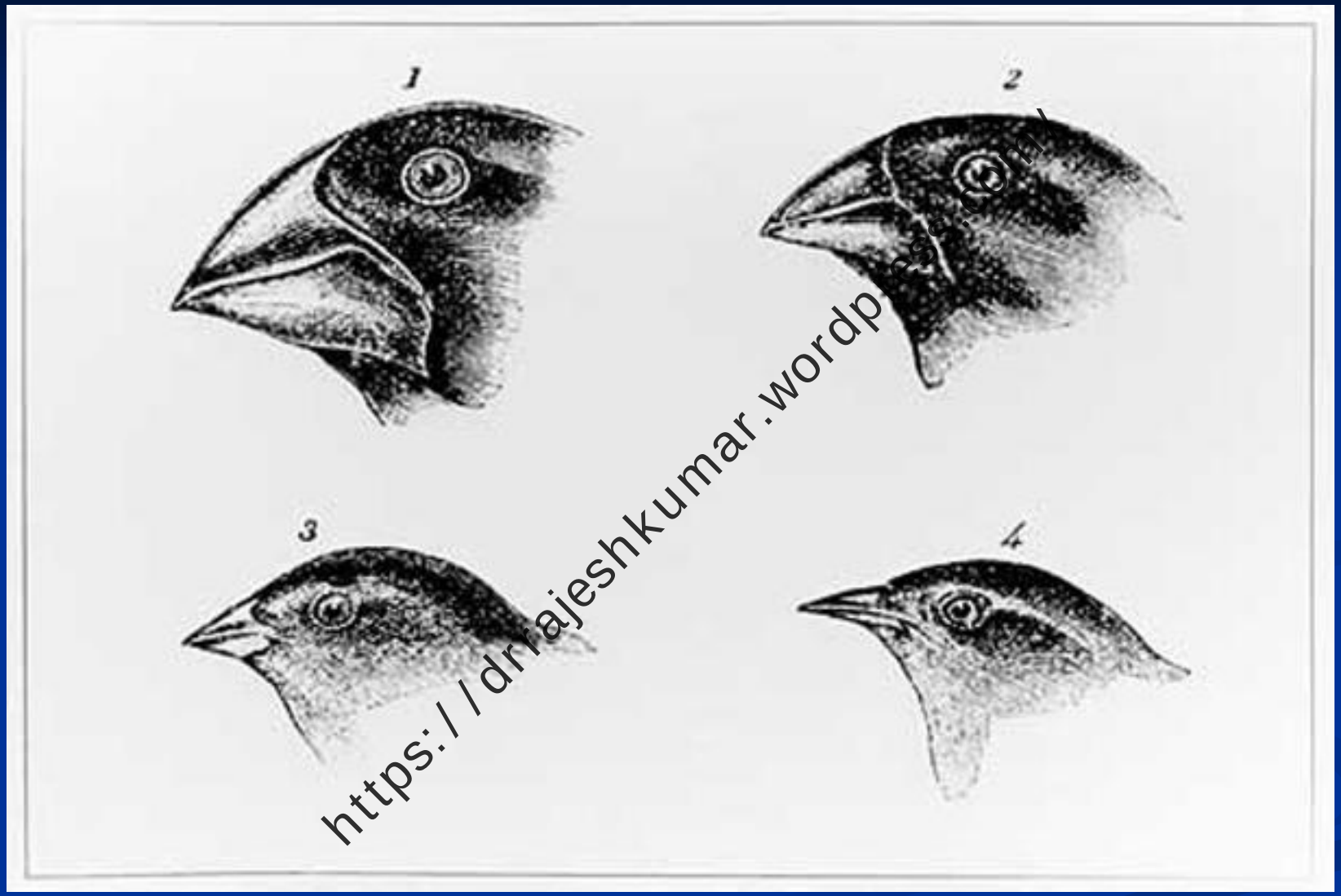



https://drrajeshkumar.wordpress.com/


https://drrajeshkumar.wordpress.com/







It can live up to 70 years

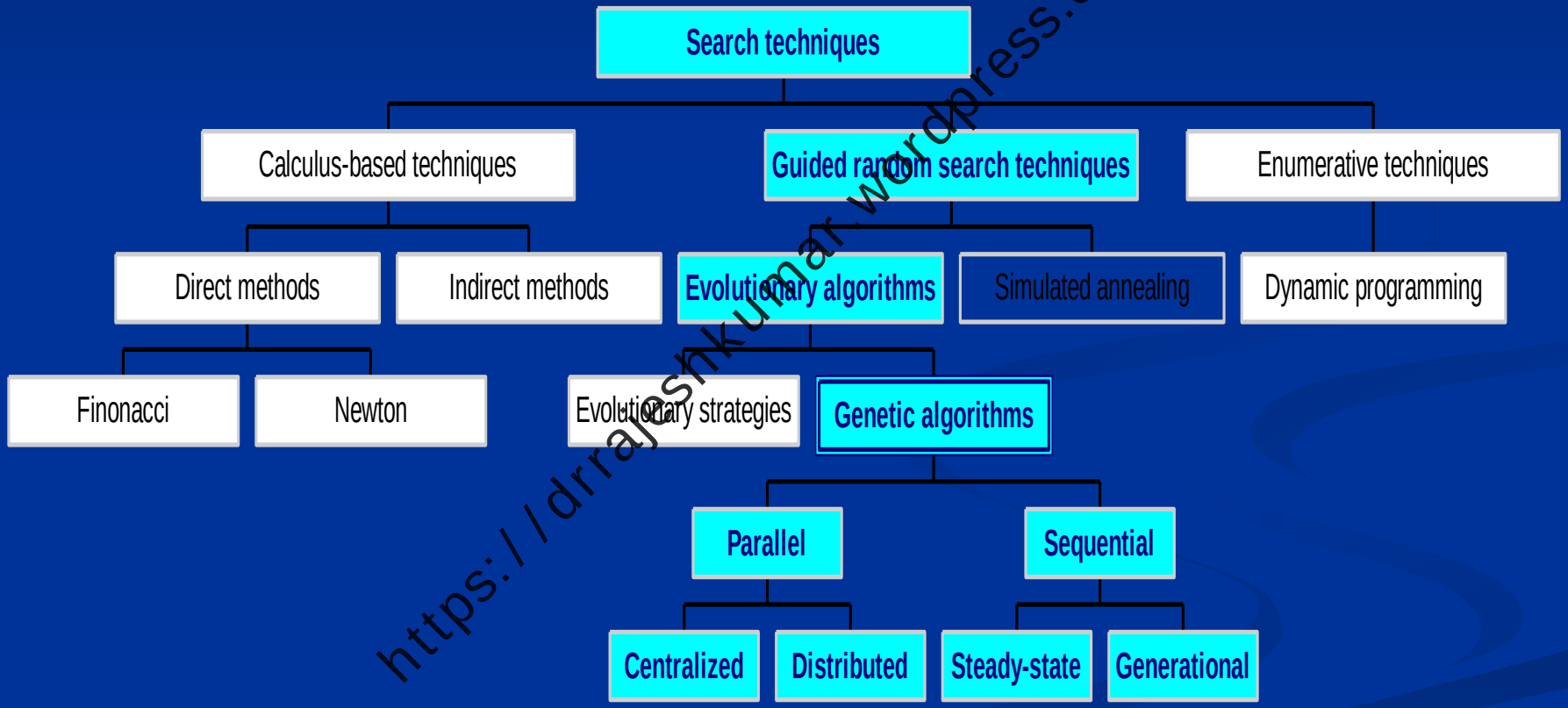
But to reach this age, the eagle must make a hard decision

Scientists have tried to mimic the nature throughout the history.

The nontraditional optimization algorithms are

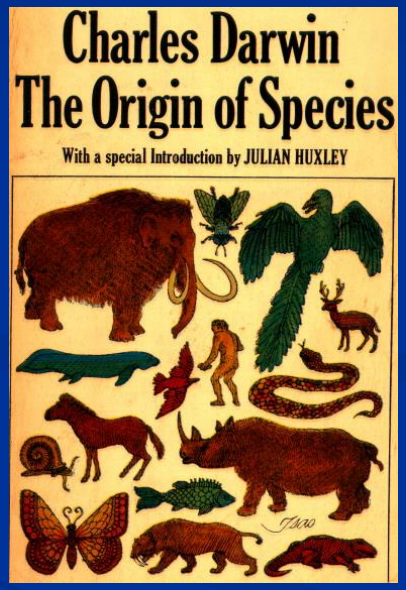
- Genetic Algorithms
- Neural Networks
- Ant Algorithms
- Simulated Annealing

Classes of Search Techniques



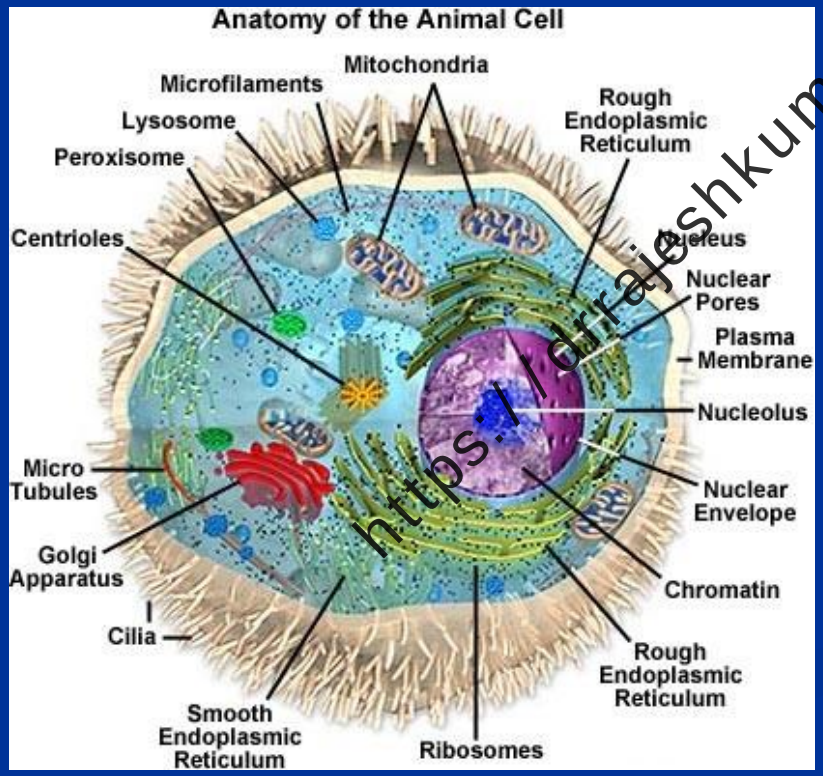
General Introduction to GA's

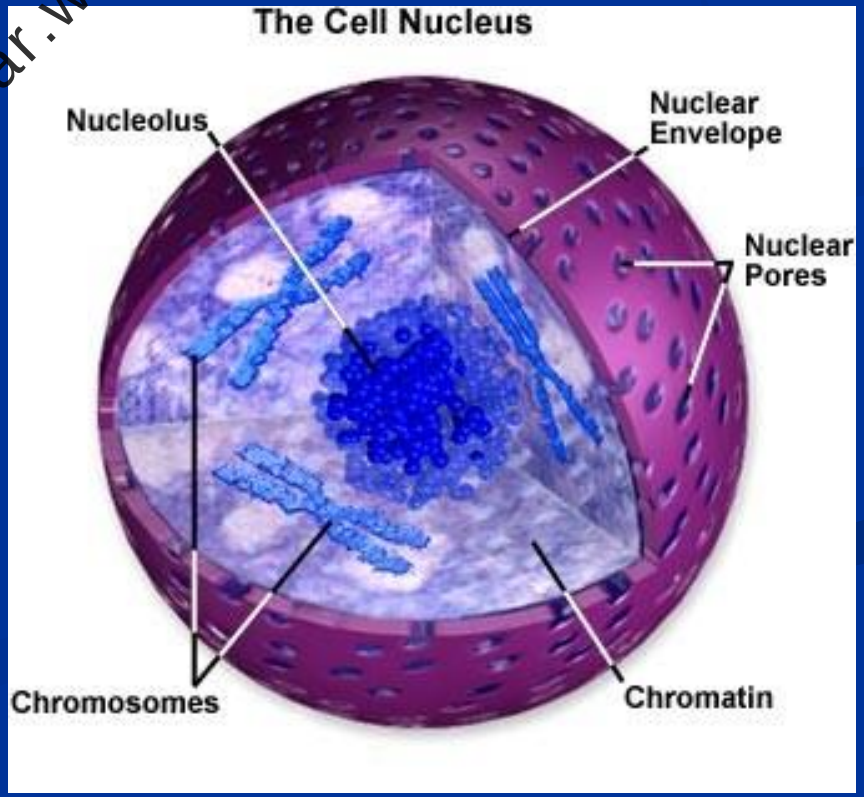
- Genetic algorithms (GA's) are a technique to solve problems which need optimization
- GA's are a subclass of Evolutionary Computing
- GA's are based on Darwin's theory of evolution
- History of GA's
  - Evolutionary computing evolved in the 1960's.
  - GA's were created by John Holland in the mid-70's.



Biological Background (1) – The cell

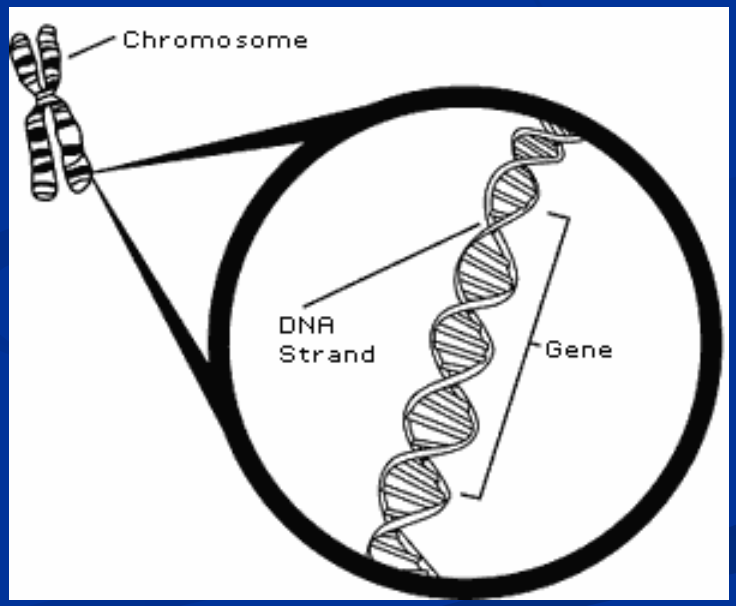
- Every animal cell is a complex of many small "factories" working together
- The center of this all is the cell nucleus
- The nucleus contains the genetic information





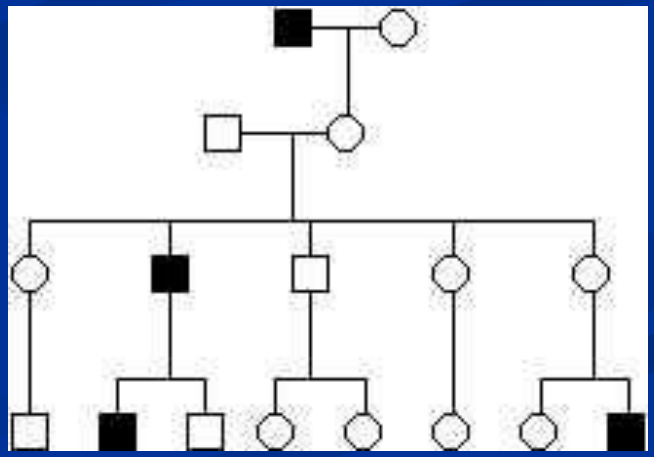
Biological Background (2) – Chromosomes

- Genetic information is stored in the chromosomes
- Each chromosome is build of DNA
- Chromosomes in humans form pairs
- There are 23 pairs
- The chromosome is divided in parts: genes
- Genes code for properties
- The posibilities of the genes for one property is called: allele
- Every gene has an uniqae position the chromosome: locus



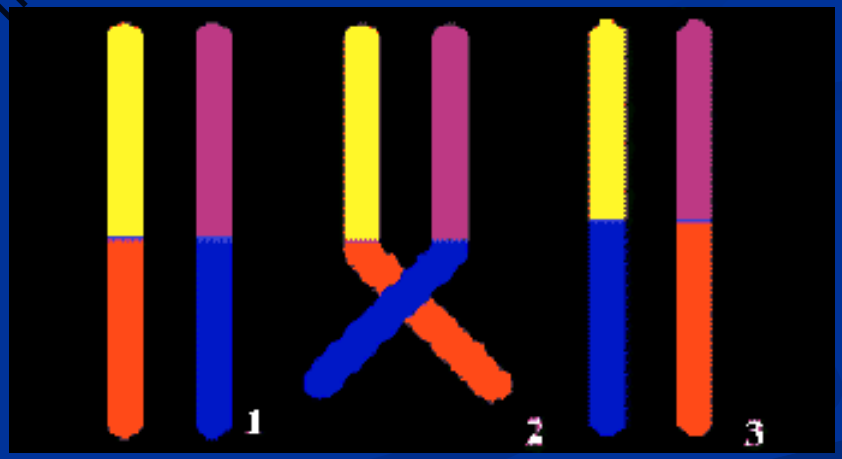
Biological Background (3) – Genetics

- The entire combination of genes is called genotype
- A genotype develops to a phenotype
- Alleles can be either dominant or recessive
- Dominant alleles will always express from the genotype to the fenotype
- Recessive alleles can survive in the population for many generations, without being expressed.



Biological Background (4) – Reproduction

- During reproduction "errors" occur
- Due to these "errors" genetic variation exists
- Most important "errors" are:
  - Recombination (cross-over)
  - Mutation



Biological Background (5) – Natural selection

- The origin of species: "Preservation of favourable variations and rejection of unfavourable variations."

- There are more individuals born than can survive, so there is a continuous struggle for life.

- Individuals with an advantage have a greater chance for survive: survival of the fittest.

Biological Background (6) – Natural selection

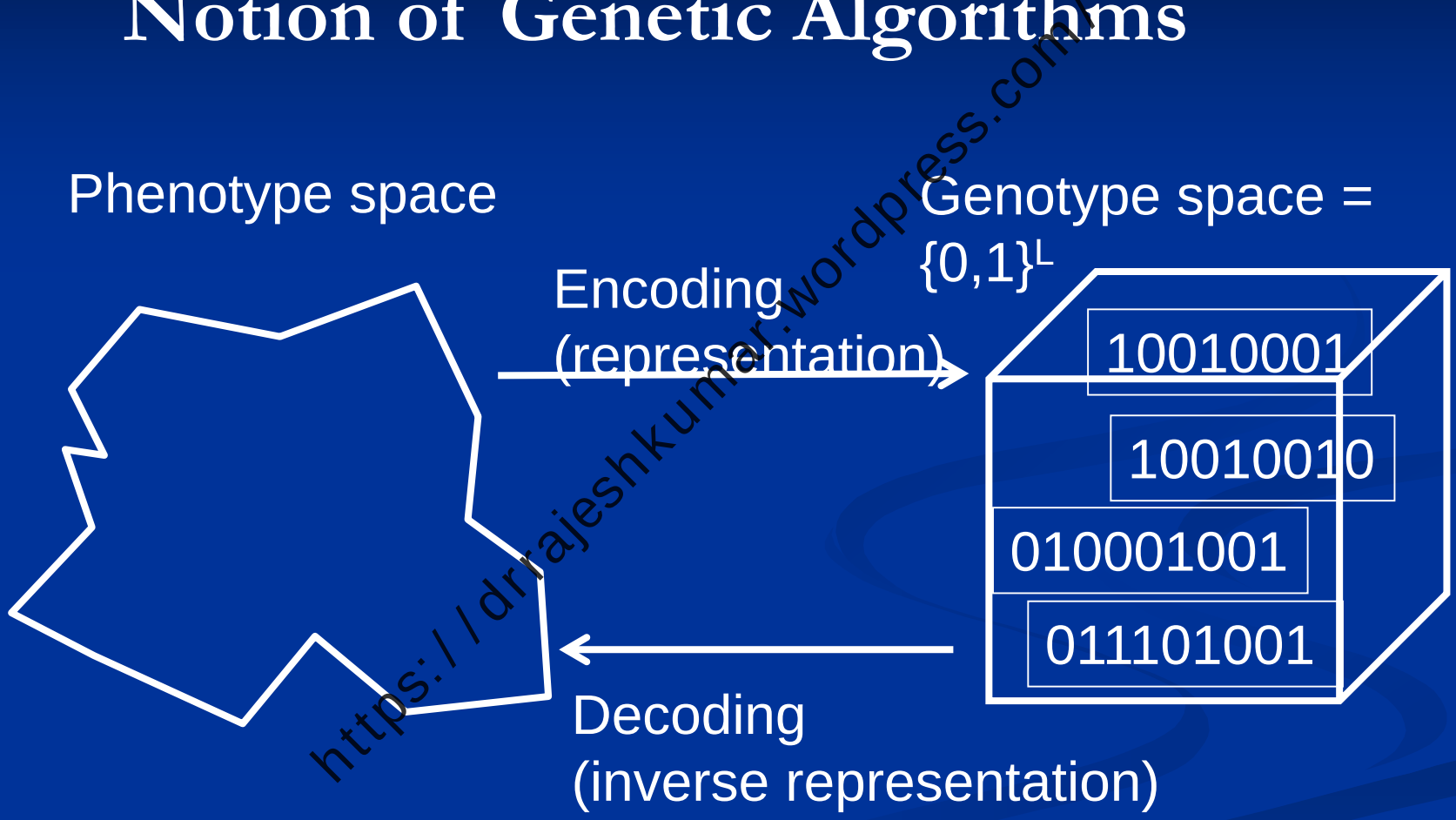
- Important aspects in natural selection are:
  - adaptation to the environment
  - isolation of populations in different groups which cannot mutually mate

- If small changes in the genotypes of individuals are expressed easily, especially in small populations, we speak of genetic drift

- Mathematical expresses as fitness: success in life

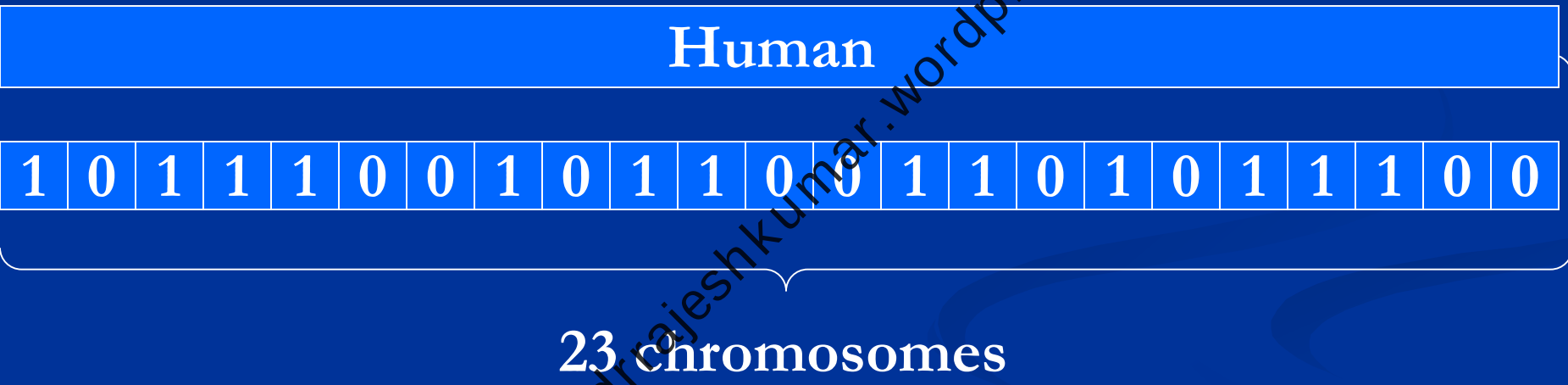
Genetic Algorithms

Notion of Genetic Algorithms

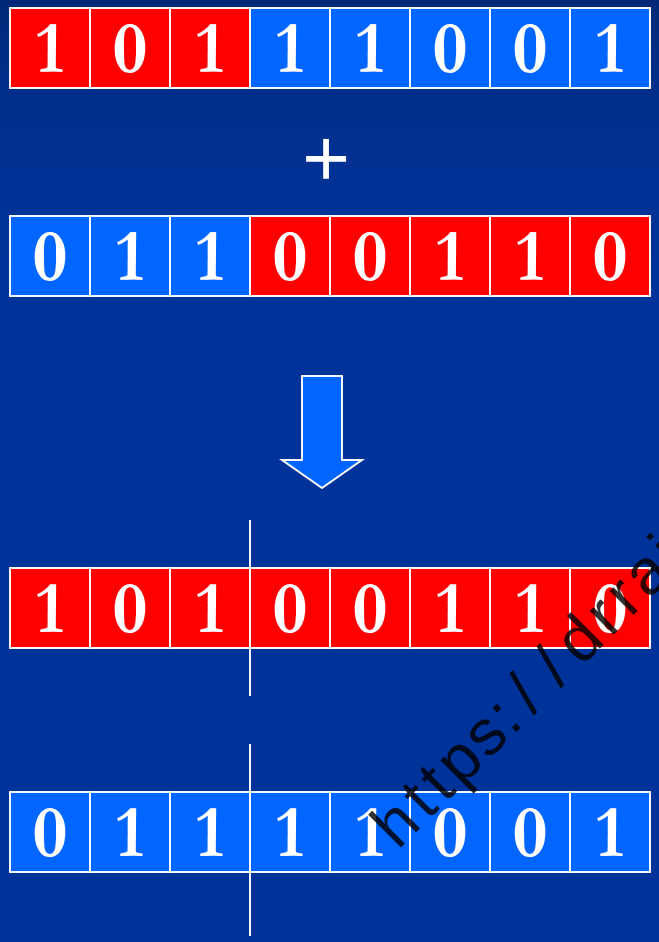


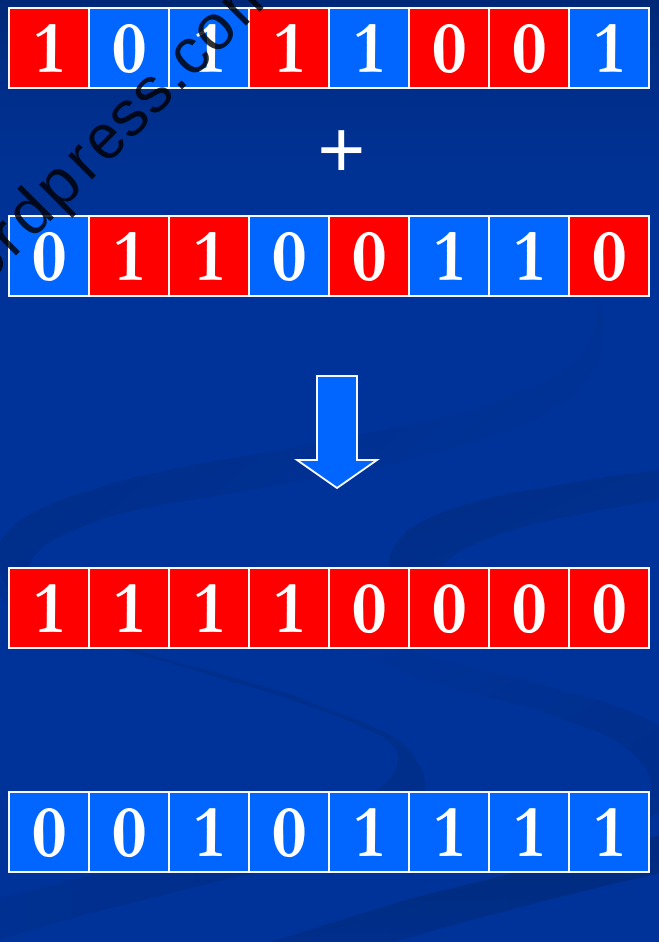
Genetic Algorithms

Notion of Genetic Algorithms



A fetus is formed by a Male(sperm) and female(egg).

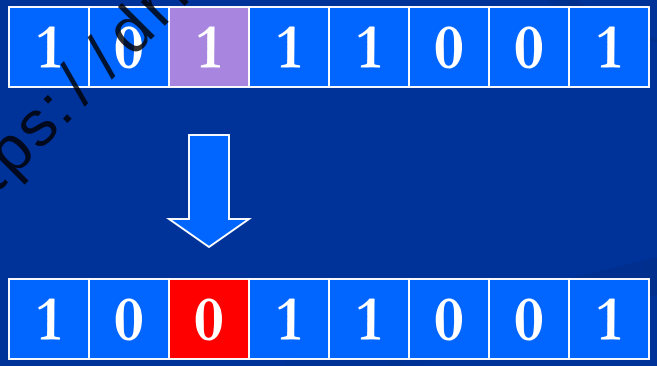




- The evolutionary process can be expedited by improving the variety of the gene pool.

  It is done via mutation.

  Mutation Process



Simple Genetic Algorithm

```
{
    initialize population;
    evaluate population;
    while Termination CriteriaNotSatisfied
    {
        select parents for reproduction;
        perform recombination and mutation;
        evaluate population;
    }
}
```

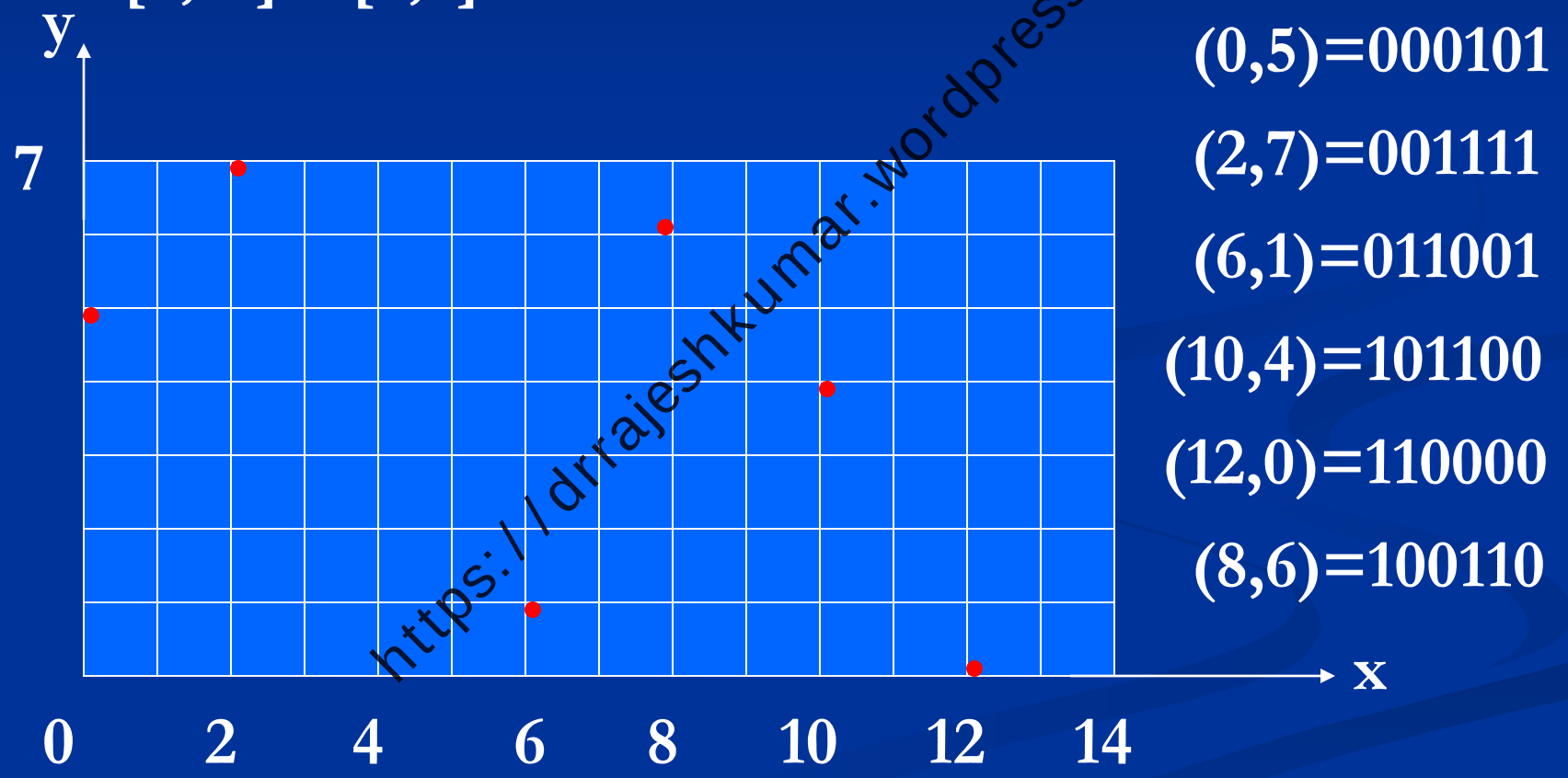
- Genetic algorithms are usually applied for maximization problems.

  To minimize f(x) (f(x)>0)using GAs, consider maximization of

$$\frac{1}{1+f(x)}$$

Example

Maximize $y^{1.3}+10e^{(-xy)}+\sin(x-y)+3$ in $R=[0,14] \times [0,7]$.



(0,5)=000101

(2,7)=001111

(6,1)=011001

(10,4)=101100

(12,0)=110000

(8,6)=100110

String	(x,y)	f(x,y)
000101	(0,5)	21.02
001111	(2,7)	15.46
011001	(6,1)	4.11
101100	(10,4)	9.17
110000	(12,0)	13.21
100110	(8,6)	13.31

Fitness:

$$\frac{21.02}{21.02+15.46+4.11+9.17+13.21+13.31} = 0.276$$

String	(x,y)	f(x,y)	Prob.
000101	(0,5)	21.02	0.276
001111	(2,7)	15.46	
011001	(6,1)	4.11	
101100	(10,4)	9.17	
110000	(12,0)	13.21	
100110	(8,6)	13.31	

Fitness:

$$\frac{21.02}{21.02+15.46+4.11+9.17+13.21+13.31} = 0.276$$

String	(x,y)	f(x,y)	Prob.	Cum.Prob
000101	(0,5)	21.02	0.276	0.276
001111	(2,7)	15.46	0.203	0.479
011001	(6,1)	4.11	0.054	0.533
101100	(10,4)	9.17	0.120	0.653
110000	(12,0)	13.21	0.173	0.826
100110	(8,6)	13.31	0.174	1.000

String	(x,y)	f(x,y)	Prob.	Cum.Prob	String
000101	(0,5)	21.02	0.276	0.276	
001111	(2,7)	15.46	0.203	0.479	
011001	(6,1)	4.11	0.054	0.533	
101100	(10,4)	9.17	0.120	0.653	
110000	(12,0)	13.21	0.173	0.826	
100110	(8,6)	13.31	0.174	1.000	

String	(x,y)	f(x,y)	Prob.	Cum.Prob	String
000101	(0,5)	21.02	0.276	0.276	
001111	(2,7)	15.46	0.203	0.479	
011001	(6,1)	4.11	0.054	0.533	
101100	(10,4)	9.17	0.120	0.653	
110000	(12,0)	13.21	0.173	0.826	
100110	(8,6)	13.31	0.174	1.000	

String	(x,y)	f(x,y)	Prob.	Cum.Prob	String
000101	(0,5)	21.02	0.276	0.276	
001111	(2,7)	15.46	0.203	0.479	
011001	(6,1)	4.11	0.054	0.533	
101100	(10,4)	9.17	0.120	0.653	
110000	(12,0)	13.21	0.173	0.826	
100110	(8,6)	13.31	0.174	1.000	

String	(x,y)	f(x,y)	Prob.	Cum.Prob	String
000101	(0,5)	21.02	0.276	0.276	000101
001111	(2,7)	15.46	0.203	0.479	
011001	(6,1)	4.11	0.054	0.533	
101100	(10,4)	9.17	0.120	0.653	
110000	(12,0)	13.21	0.173	0.826	
100110	(8,6)	13.31	0.174	1.000	

String	(x,y)	f(x,y)	Prob.	Cum.Prob	String
000101	(0,5)	21.02	0.276	0.276	000101
001111	(2,7)	15.46	0.203	0.479	
011001	(6,1)	4.11	0.054	0.533	
101100	(10,4)	9.17	0.120	0.653	
110000	(12,0)	13.21	0.173	0.826	
100110	(8,6)	13.31	0.174	1.000	

String	(x,y)	f(x,y)	Prob.	Cum.Prob	String
000101	(0,5)	21.02	0.276	0.276	000101
001111	(2,7)	15.46	0.203	0.479	100110
011001	(6,1)	4.11	0.054	0.533	
101100	(10,4)	9.17	0.120	0.653	
110000	(12,0)	13.21	0.173	0.826	
100110	(8,6)	13.31	0.174	1.000	

String	(x,y)	f(x,y)	Prob.	Cum.Prob	String
000101	(0,5)	21.02	0.276	0.276	000101
001111	(2,7)	15.46	0.203	0.479	100110
011001	(6,1)	4.11	0.054	0.533	000101
101100	(10,4)	9.17	0.120	0.653	001111
110000	(12,0)	13.21	0.173	0.826	110000
100110	(8,6)	13.31	0.174	1.000	101100

String
000101
100110
000101
001111
110000
101100

Crossover

String	(x,y)	f(x,y)	Prob.	Cum.Prob	String
000101	(0,5)	21.02	0.276	0.276	000101
001111	(2,7)	15.46	0.203	0.479	100110
011001	(6,1)	4.11	0.054	0.533	000101
101100	(10,4)	9.17	0.120	0.653	001111
110000	(12,0)	13.21	0.173	0.826	110000
100110	(8,6)	13.31	0.174	1.000	101100

String
000101
100110
000101
001111
110000
101100

Crossover

String

String	(x,y)	f(x,y)	Prob.	Cum.Prob	String
000101	(0,5)	21.02	0.276	0.276	000101
001111	(2,7)	15.46	0.203	0.479	100110
011001	(6,1)	4.11	0.054	0.533	000101
101100	(10,4)	9.17	0.120	0.653	001111
110000	(12,0)	13.21	0.173	0.826	110000
100110	(8,6)	13.31	0.174	1.000	101100

String
000101
100110
000101
001111
110000
101100

Crossover

String

String	(x,y)	f(x,y)	Prob.	Cum.Prob	String
000101	(0,5)	21.02	0.276	0.276	000101
001111	(2,7)	15.46	0.203	0.479	100110
011001	(6,1)	4.11	0.054	0.533	000101
101100	(10,4)	9.17	0.120	0.653	001111
110000	(12,0)	13.21	0.173	0.826	110000
100110	(8,6)	13.31	0.174	1.000	101100

String
000101
100110
000101
001111
110000
101100

Crossover

String
000110

String	(x,y)	f(x,y)	Prob.	Cum.Prob	String
000101	(0,5)	21.02	0.276	0.276	000101
001111	(2,7)	15.46	0.203	0.479	100110
011001	(6,1)	4.11	0.054	0.533	000101
101100	(10,4)	9.17	0.120	0.653	001111
110000	(12,0)	13.21	0.173	0.826	110000
100110	(8,6)	13.31	0.174	1.000	101100

String
000101
100110
000101
001111
110000
101100

Crossover

String
000110
100101

String	(x,y)	f(x,y)	Prob.	Cum.Prob	String
000101	(0,5)	21.02	0.276	0.276	000101
001111	(2,7)	15.46	0.203	0.479	100110
011001	(6,1)	4.11	0.054	0.533	000101
101100	(10,4)	9.17	0.120	0.653	001111
110000	(12,0)	13.21	0.173	0.826	110000
100110	(8,6)	13.31	0.174	1.000	101100

String
000101
100110
000101
001111
110000
101100

Crossover

String
000110
100101

String	(x,y)	f(x,y)	Prob.	Cum.Prob	String
000101	(0,5)	21.02	0.276	0.276	000101
001111	(2,7)	15.46	0.203	0.479	100110
011001	(6,1)	4.11	0.054	0.533	000101
101100	(10,4)	9.17	0.120	0.653	001111
110000	(12,0)	13.21	0.173	0.826	110000
100110	(8,6)	13.31	0.174	1.000	101100

String
000101
100110
000101
001111
110000
101100

Crossover

String
000110
100101
000111
001101
111100
100000

Mutation

String
000110
100101
000111
001101
111100
100000

Changes if p>0.95

String	(x,y)	f(x,y)	Prob.	Cum.Prob	String
000101	(0,5)	21.02	0.276	0.276	000101
001111	(2,7)	15.46	0.203	0.479	100110
011001	(6,1)	4.11	0.054	0.533	000101
101100	(10,4)	9.17	0.120	0.653	001111
110000	(12,0)	13.21	0.173	0.826	110000
100110	(8,6)	13.31	0.174	1.000	101100

String
000101
100110
000101
001111
110000
101100

Crossover →

String
000110
100101
000111
001101
111100
100000

Mutation →

String
000110
100101
000111
001101
111100
100000

String	(x,y)	f(x,y)	Prob.	Cum.Prob	String
000101	(0,5)	21.02	0.276	0.276	000101
001111	(2,7)	15.46	0.203	0.479	100110
011001	(6,1)	4.11	0.054	0.533	000101
101100	(10,4)	9.17	0.120	0.653	001111
110000	(12,0)	13.21	0.173	0.826	110000
100110	(8,6)	13.31	0.174	1.000	101100

String
000101
100110
000101
001111
110000
101100

Crossover →

String
000110
100101
000111
001101
111100
100000

Mutation →

String
000110
001101
000111
001101
111100
110000

String	(x,y)	f(x,y)	Prob.	Cum.Prob	String
000101	(0,5)	21.02	0.276	0.276	000101
001111	(2,7)	15.46	0.203	0.479	100110
011001	(6,1)	4.11	0.054	0.533	000101
101100	(10,4)	9.17	0.120	0.653	001111
110000	(12,0)	13.21	0.173	0.826	110000
100110	(8,6)	13.31	0.174	1.000	101100

String
000101
100110
000101
001111
110000
101100

Crossover

String
000110
100101
000111
001101
111100
100000

Mutation

String
000110
001101
000111
001101
111100
110000

Go to iter. 2

String
000110
001101
000111
001101
111100
110000

String	(x,y)	f(x,y)
000110	(0,6)	23.17
001101	(2,5)	11.05
000111	(0,7)	25.43
001101	(2,5)	11.05
111100	(14,4)	9.24
110000	(12,0)	13.21

Continue

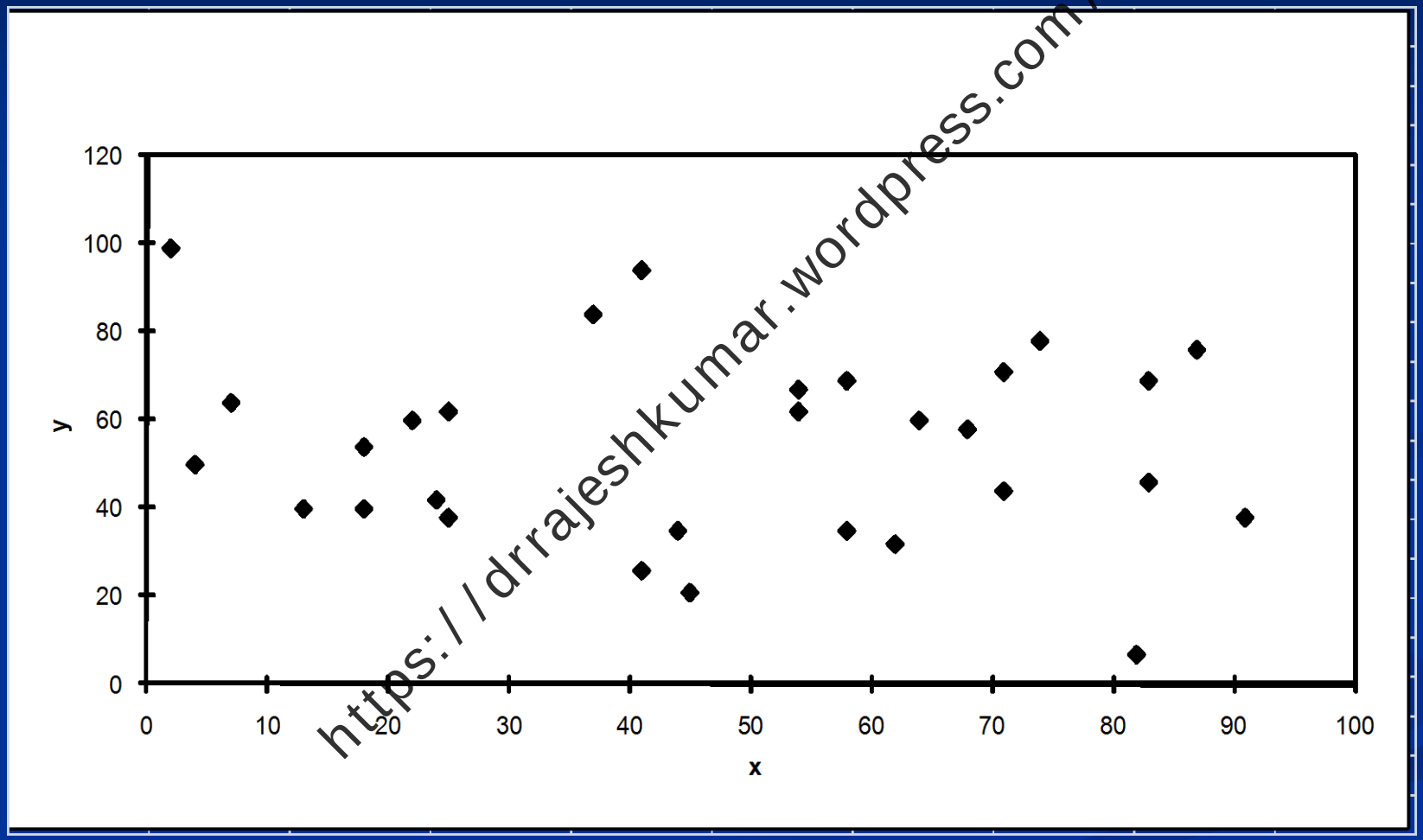
Previous Avg. = 12.71

New Avg. = 15.32

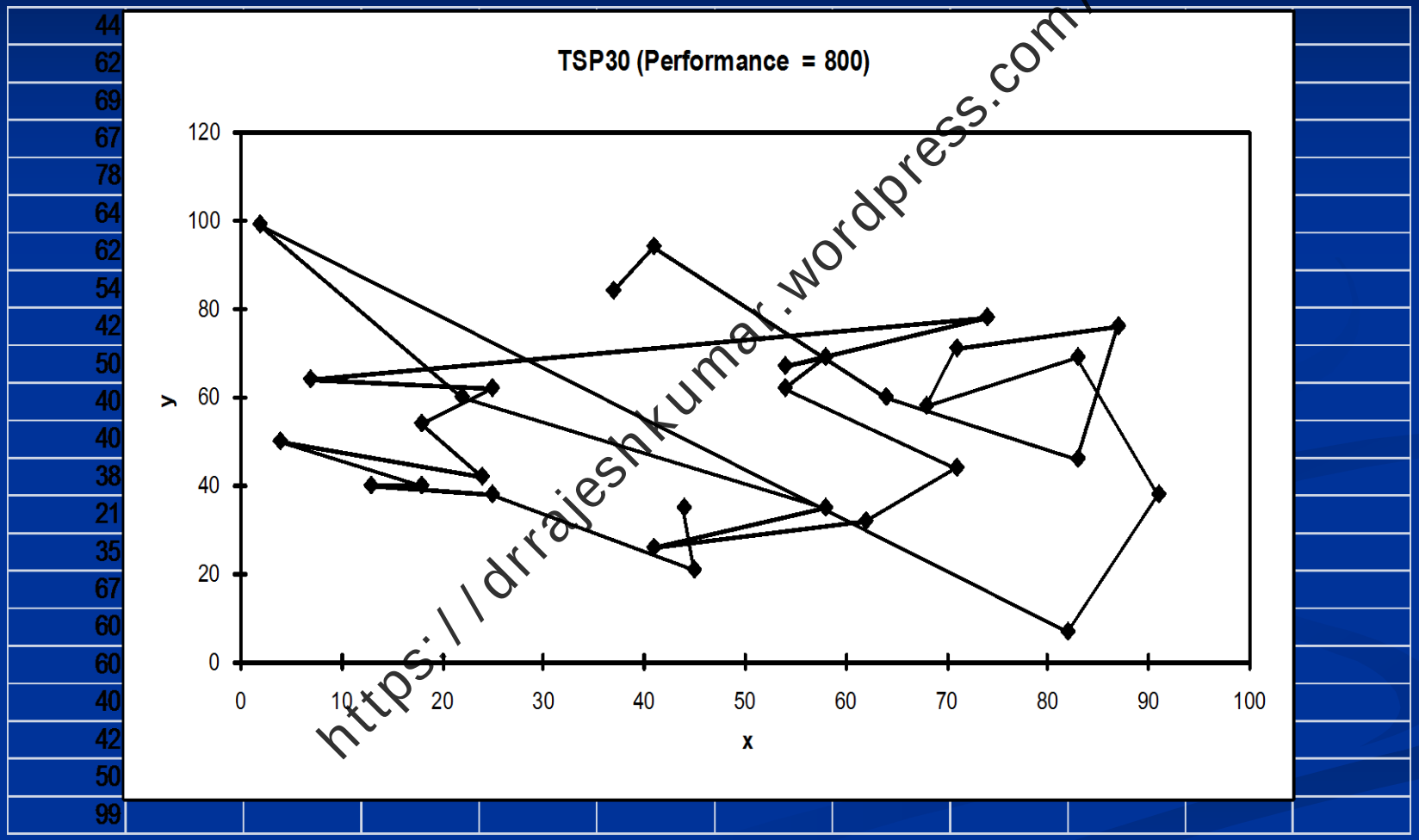
Previous Max. = 21.02

New Max. = 25.43

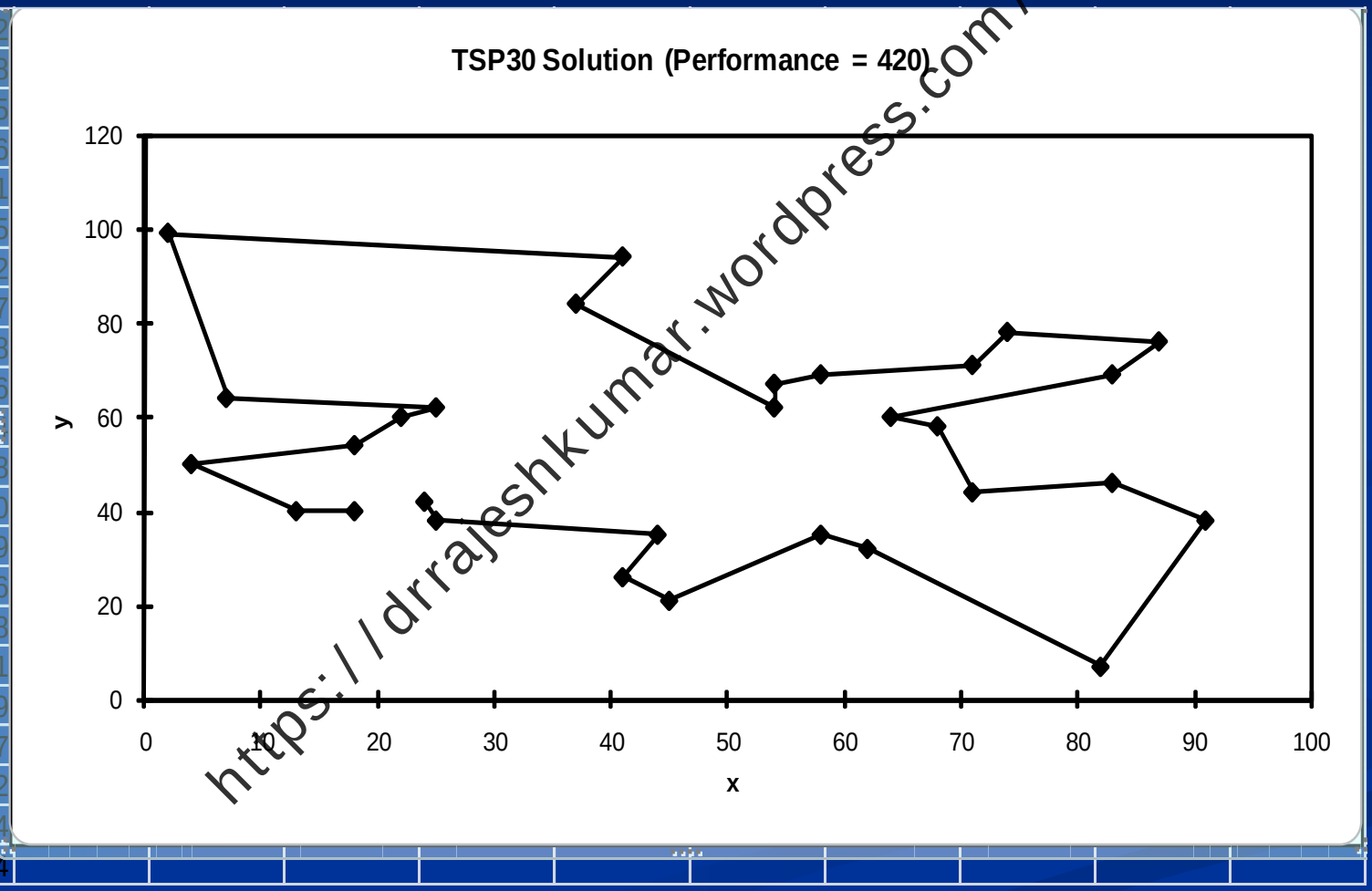
TSP Example: 30 Cities



Solution $_j$(Distance = 800)



Best Solution (Distance = 420)



Benefits of Genetic Algorithms

- Concept is easy to understand
- Modular, separate from application
- Supports multi-objective optimization
- Good for "noisy" environments
- Always an answer; answer gets better with time
- Inherently parallel; easily distributed

Benefits of Genetic Algorithms (cont.)

- Many ways to speed up and improve a GA-based application as knowledge about problem domain is gained
- Easy to exploit previous or alternate solutions
- Flexible building blocks for hybrid applications
- Substantial history and range of use

When to Use a GA

- Alternate solutions are too slow or overly complicated
- Need an exploratory tool to examine new approaches
- Problem is similar to one that has already been successfully solved by using a GA
- Want to hybridize with an existing solution
- Benefits of the GA technology meet key problem requirements

Some GA Application Types

Domain	Application Types
Control	gas pipeline, pole balancing, missile evasion, pursuit
Design	semiconductor layout, aircraft design, keyboard configuration, communication networks
Scheduling	manufacturing, facility scheduling, resource allocation
Robotics	trajectory planning
Machine Learning	designing neural networks, improving classification algorithms, classifier systems
Signal Processing	filter design
Game Playing	poker, checkers, prisoner's dilemma
Combinatorial Optimization	set covering, travelling salesman, routing, bin packing, graph colouring and partitioning

WRITING A FUNCTION

- Select ***New*** in the MATLAB File menu.
- Select ***M-File***. This opens a new M-file in the editor.
- In the M-file, enter the following two lines of code:

function z = my_fun(x)

*z = x(1)^2 - 2*x(1)*x(2) + 6*x(1) + x(2)^2 -6*x(2);*

- Save the M-file in a directory on the MATLAB path.

FUNCTION HANDLE

- A function handle is a MATLAB value that provides a means of calling a function indirectly.
- The syntax is

handle = @function name

- For example

h = @my_fun

USING THE TOOLBOX

There are two ways to use genetic algorithm with the toolbox :

- Calling the genetic algorithm function *ga* at the command line
- Using the genetic algorithm tool , a graphical interface

AT COMMAND LINE

The syntax is

[x fval] = ga(@fitnessfun , nvars , options)

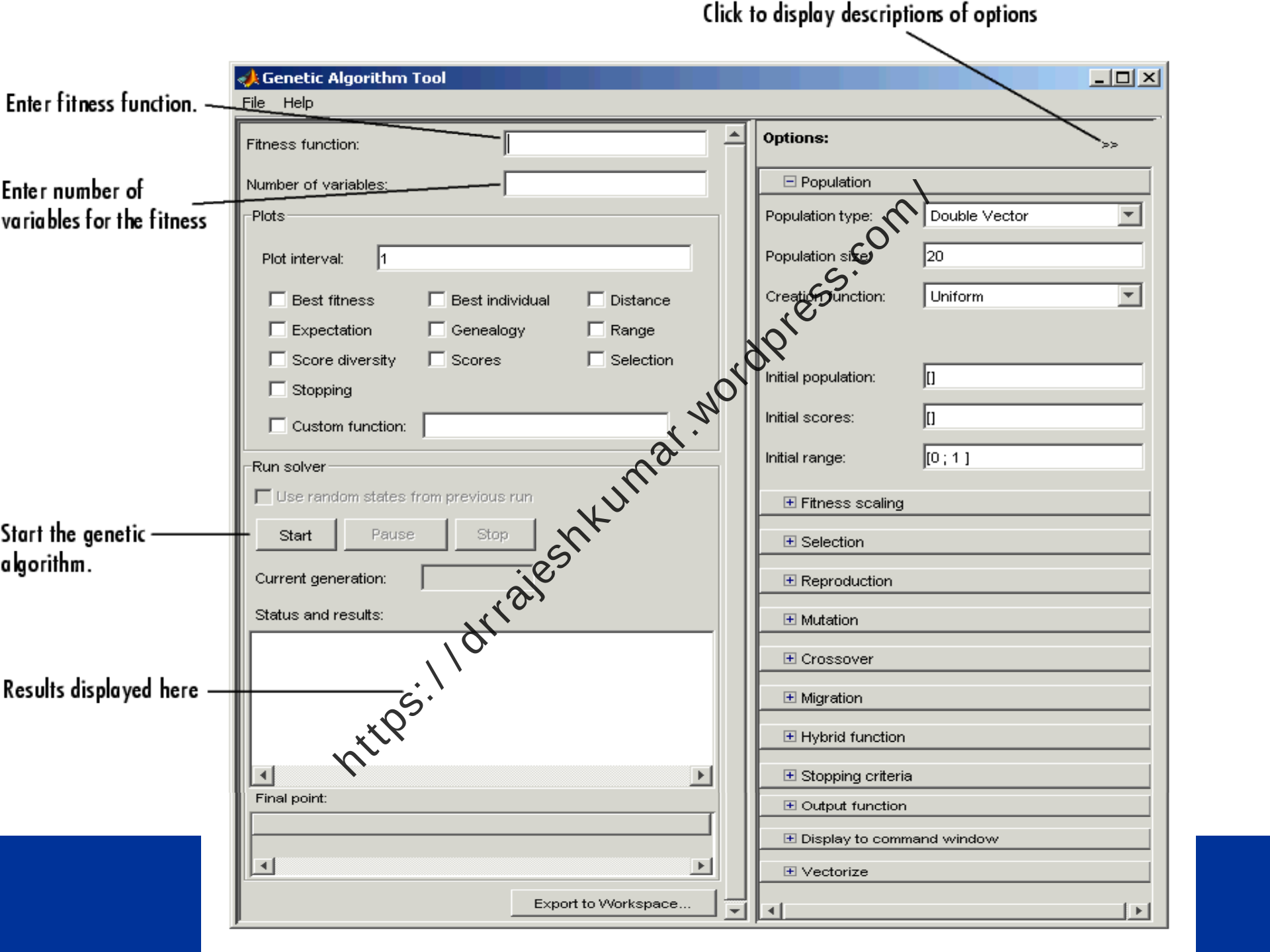
Where

- *@fitnessfun* is a function handle to the fitness function
- *nvars* is the number of independent variables for the fitness function
- *options* is a structure containing options for the genetic algorithm
- *fval* is the final value of the fitness function
- x is the point at which final value is attained

USING GENETIC ALGORITHM TOOL

- To open we have to write following syntax at the command line

  gatool



PAUSING AND STOPPING

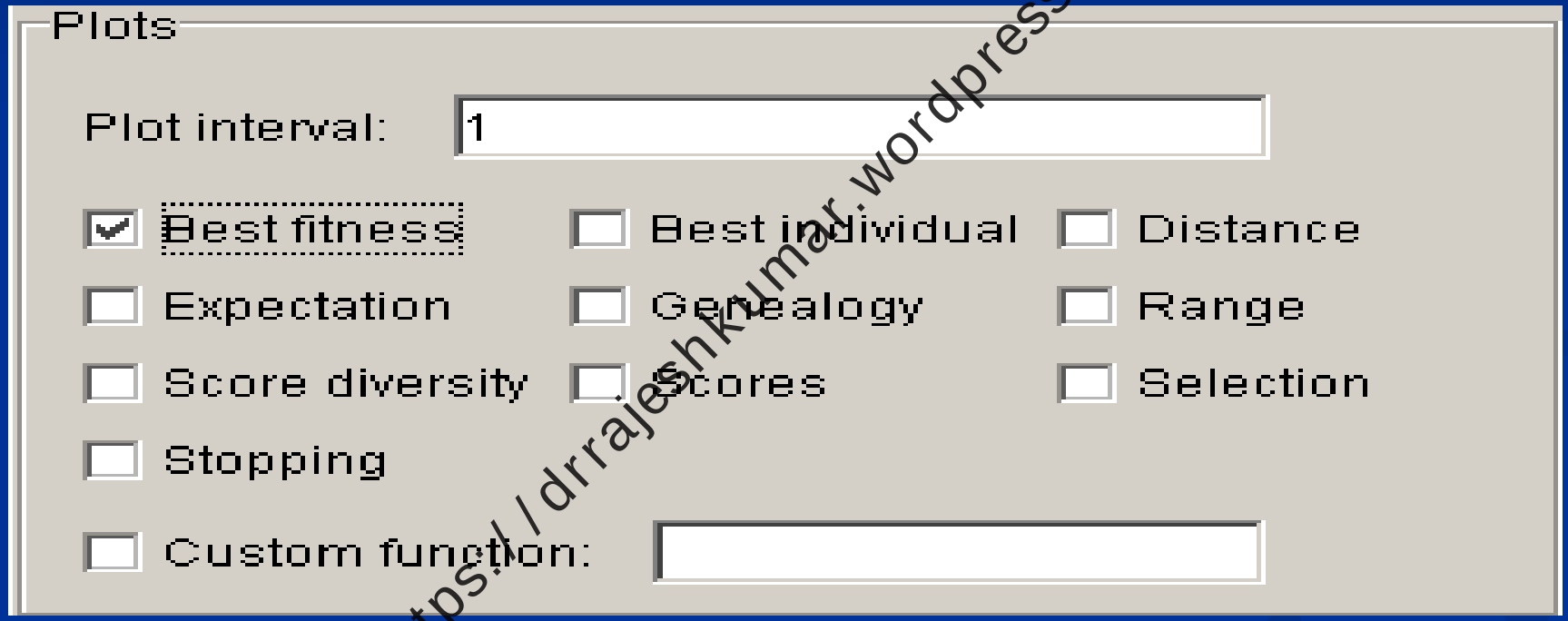
- Click *Pause* to temporarily suspend the algorithm. To resume the algorithm using the current population at the time you paused, click *Resume*.

- Click *Stop* to stop the algorithm. The Status and results pane displays the fitness function value of the best point in the current generation at the moment you clicked *Stop*.

SETTING STOPPING CRITERIA

- ***Generations*** -- The algorithm reaches the specified number of generations.
- ***Time*** -- The algorithm runs for the specified amount of time in seconds.
- ***Fitness limit*** -- The best fitness value in the current generation is less than or equal to the specified value.
- ***Stall generations*** -- The algorithm computes the specified number of generations with no improvement in the fitness function.
- ***Stall time limit*** -- The algorithm runs for the specified amount of time in seconds with no improvement in the fitness function.

DISPLAYING PLOTS



The Plots pane, shown in the following figure, enables you to display various plots of the results of the genetic algorithm.

RESULTS

- For the function we wanted to minimise i.e

  my_fun

  *z = x(1)^2 - 2*x(1)*x(2) + 6*x(1) + x(2)^2 - 6*x(2);*

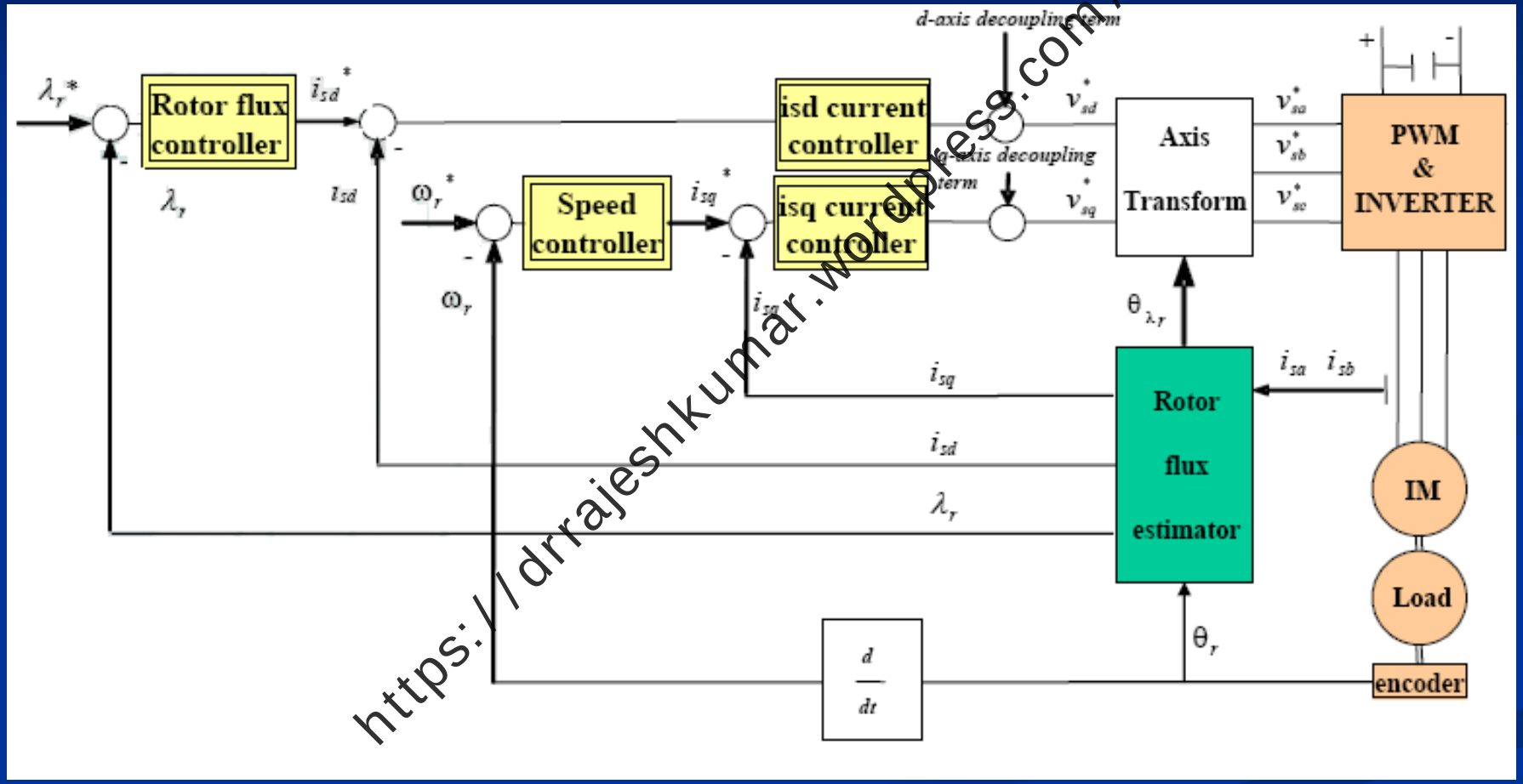
  The final result that will be obtained after using genetic algorithm will be

  x = [0.25836 3.26145]

Control of Induction Motor Drives

- Optimization of both the structure and the associated parameters of four cascaded controllers for an IM drive. The GA tests and compares controllers having different orders. The GA optimizes both the number and the location of controller's zeros and poles.
- While conventional design strategies firstly tune the current controller and then the speed controller (thus leading to a potentially sub-optimal couple of controllers), the GA simultaneously optimizes the two controllers, to obtain the best overall two cascaded- loops control systems
- We developed a system based on MATLAB/SIMULINK environment and dSPACE control boards to run the optimization on-line, directly on the hardware

Induction motor drive block diagram



Crossover, Mutation, and Selection

- **Crossover**

•Binary flags: multi-point crossover

•Real valued chromosomes: either heuristic or arithmetical crossover is applied with equal probability.

- **Mutation**

•Binary flags: binary mutation

•Real valued chromosomes: uniform mutation

- **Selection**

•Tournament selection

Main Details of the Evolutionary Algorithm

- The ability to work on heterogeneous solutions makes the considered GA similar to other evolutionary computation techniques, such as Genetic or Evolutionary Programming (GP or EP).
- A significant difference between EP and GAs lays in the rate of application of the mutation.
- In the preliminary investigation for optimal occurrence rate of operators, a much faster convergence was obtained with higher mutation rates.
- Due to our final choice of emphasizing the rate of **mutation (80%) with respect to crossover (20%), the algorithm can be** viewed as a hybrid GA-EP evolutionary algorithm.

Choice of the fitness function

- The objective function is a weighted sum of five performance indices that are directly measured on system' s response to the a sequence of speed steps of different amplitude.

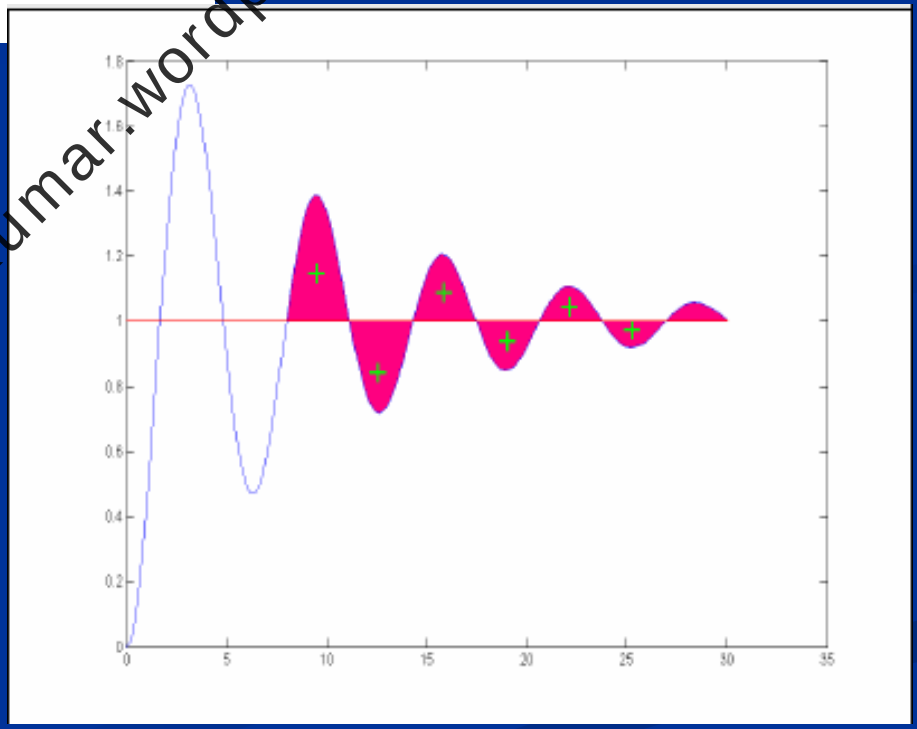
$$f = \sum_{j=1} \alpha_j \cdot f_j$$

Overview of the single indexes

- Steady state speed response

$$f_1 = \sum_{j=1}^{n} \left| \omega_r(j) - \omega_r^{ref}(j) \right| \cdot g(j)$$

This index measures the speed error along segments of the speed response settling around the steady state.

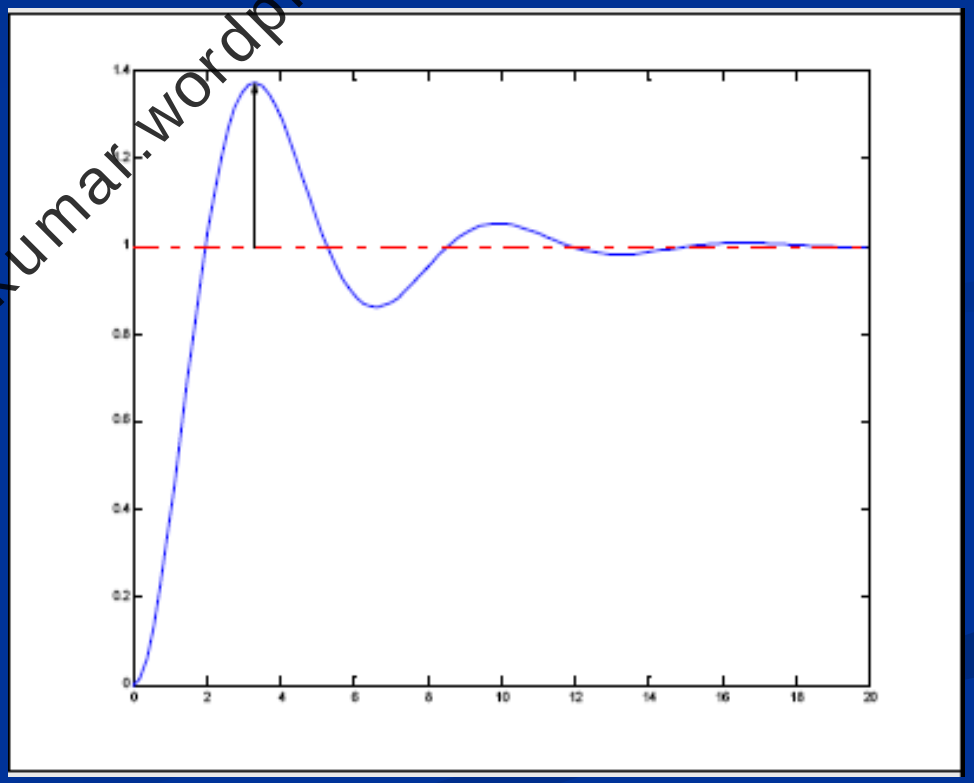


Overview of the single indexes

- Speed overshoot

$$f_2 = \frac{\omega_r^{max} - \omega_r^{steady}}{\omega_r^{steady}}$$

This index measures the speed overshoot when a speed step change is required

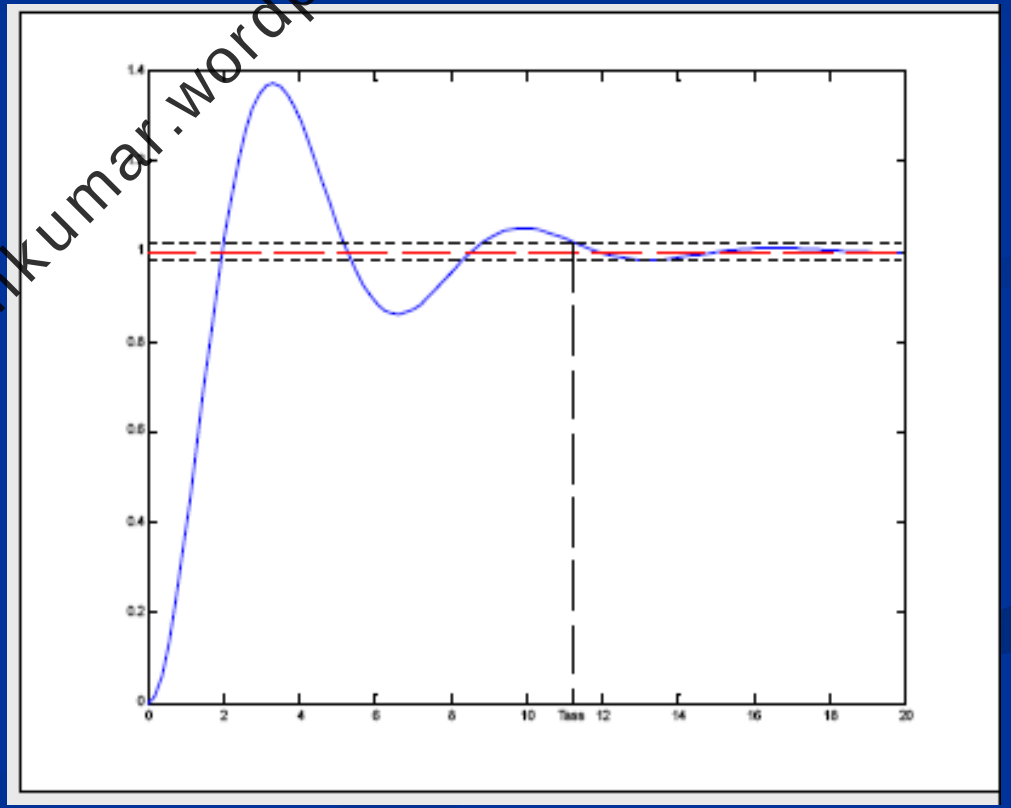


Overview of the single indexes

- Transient speed response duration

$$f_3 = \frac{n - n_t}{n}$$

This index measures the duration of transient conditions, that is the sum of all the settling times

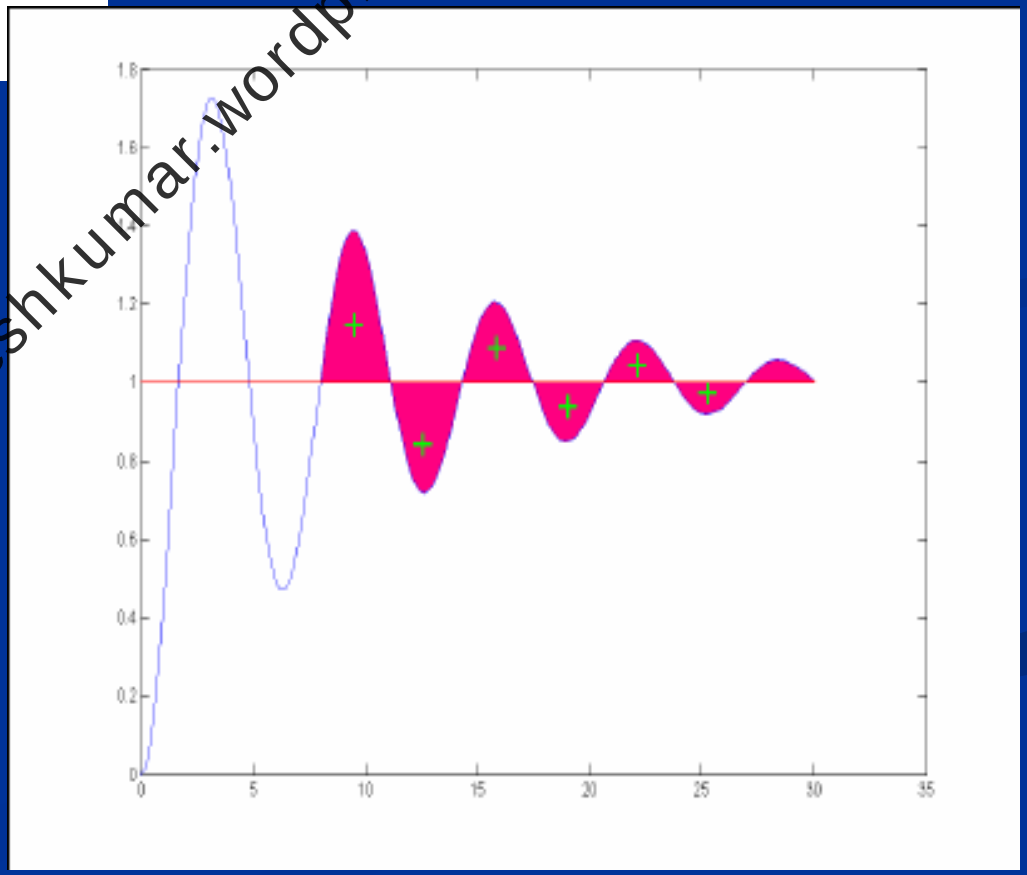


Overview of the single indexes

- Voltage reference oscillations for constant speed reference

$$f_4 = \sum_{j=1}^{n} \left| v_a^{ref}(j) - v_a^{mean}(j) \right| \cdot g(j)$$

This index accounts for undesired ripples and oscillations of the voltage that would increase losses and acoustic noise, contrasting with constant torque operations

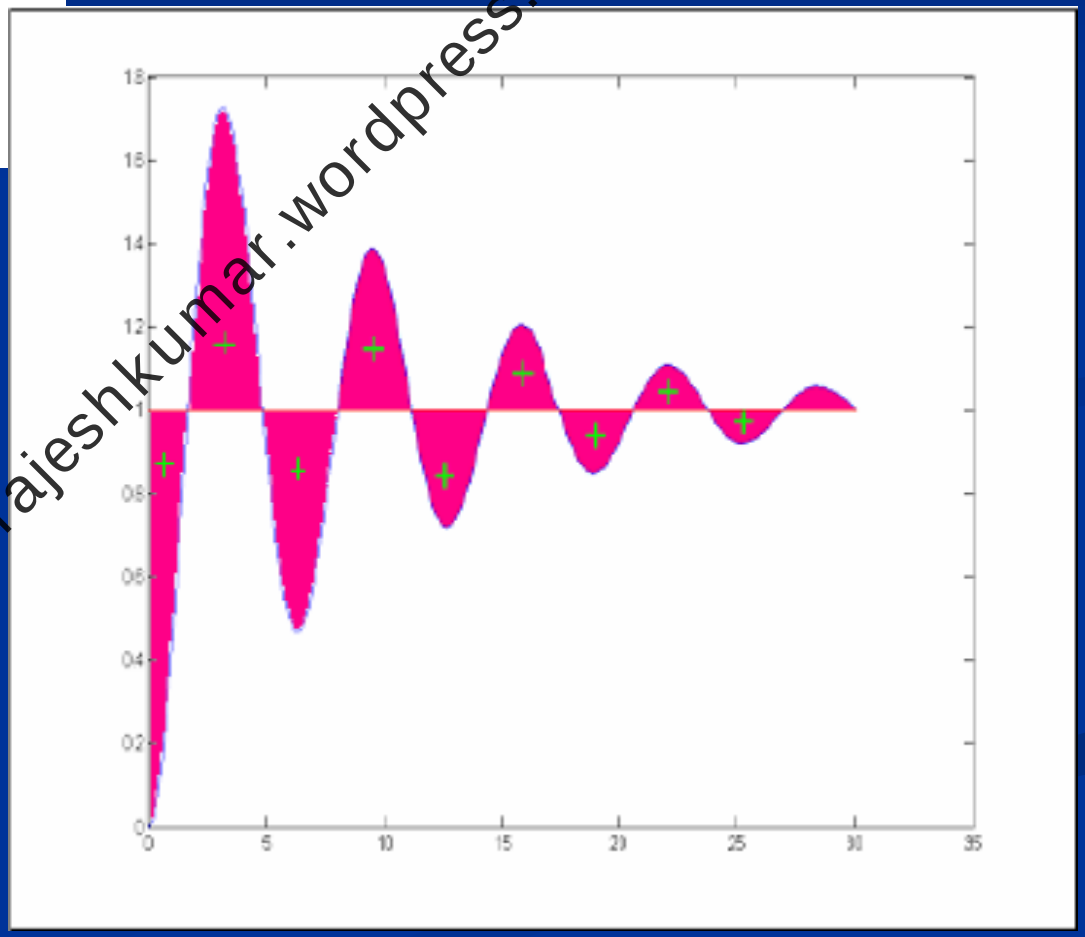


Overview of the single indexes

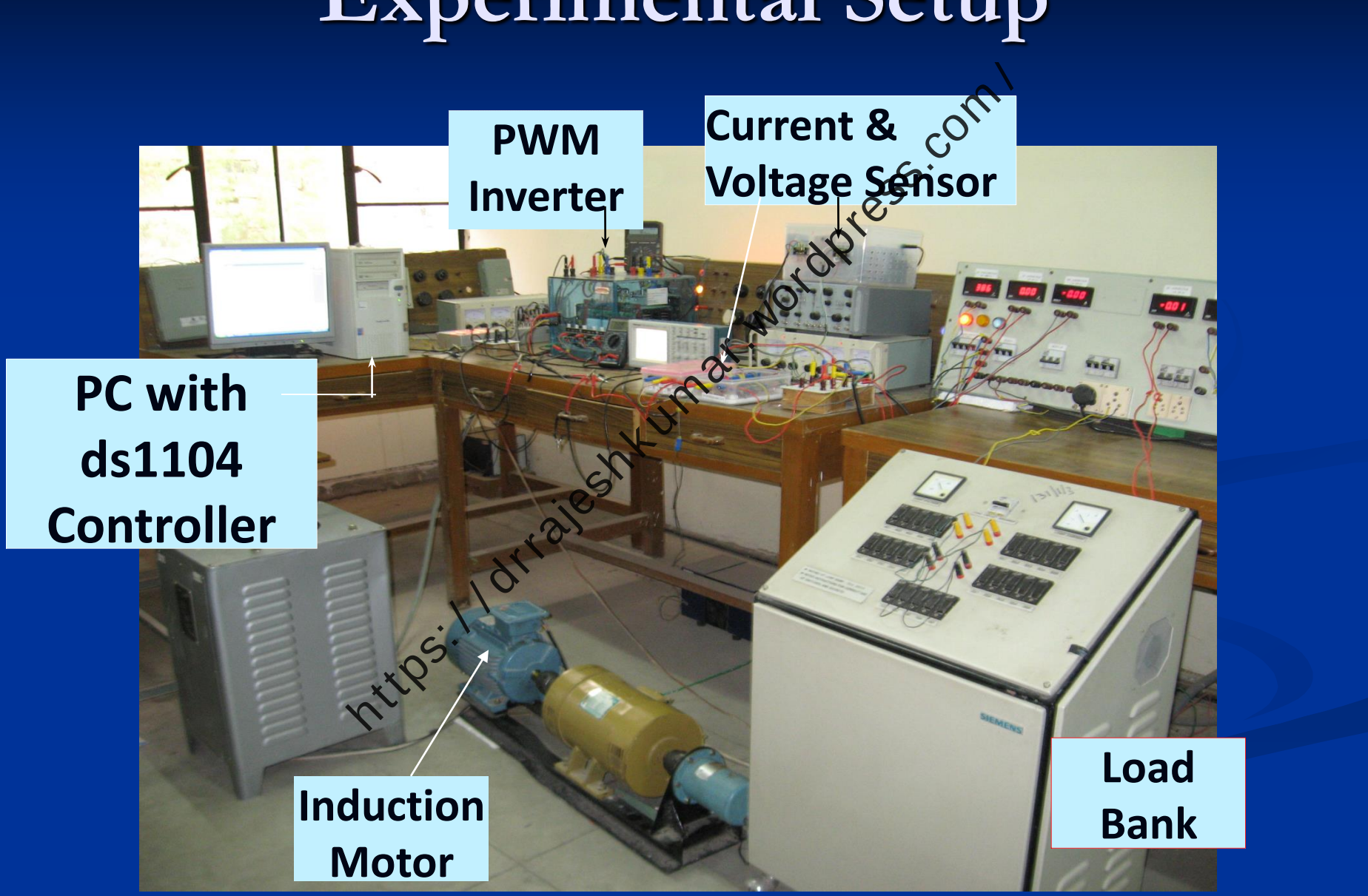
- Current response

$$f_5 = \sum_{j=1}^{n} \left| i_a(j) - i_a^{ref}(j) \right|$$

This index measures the sum of absolute current errors

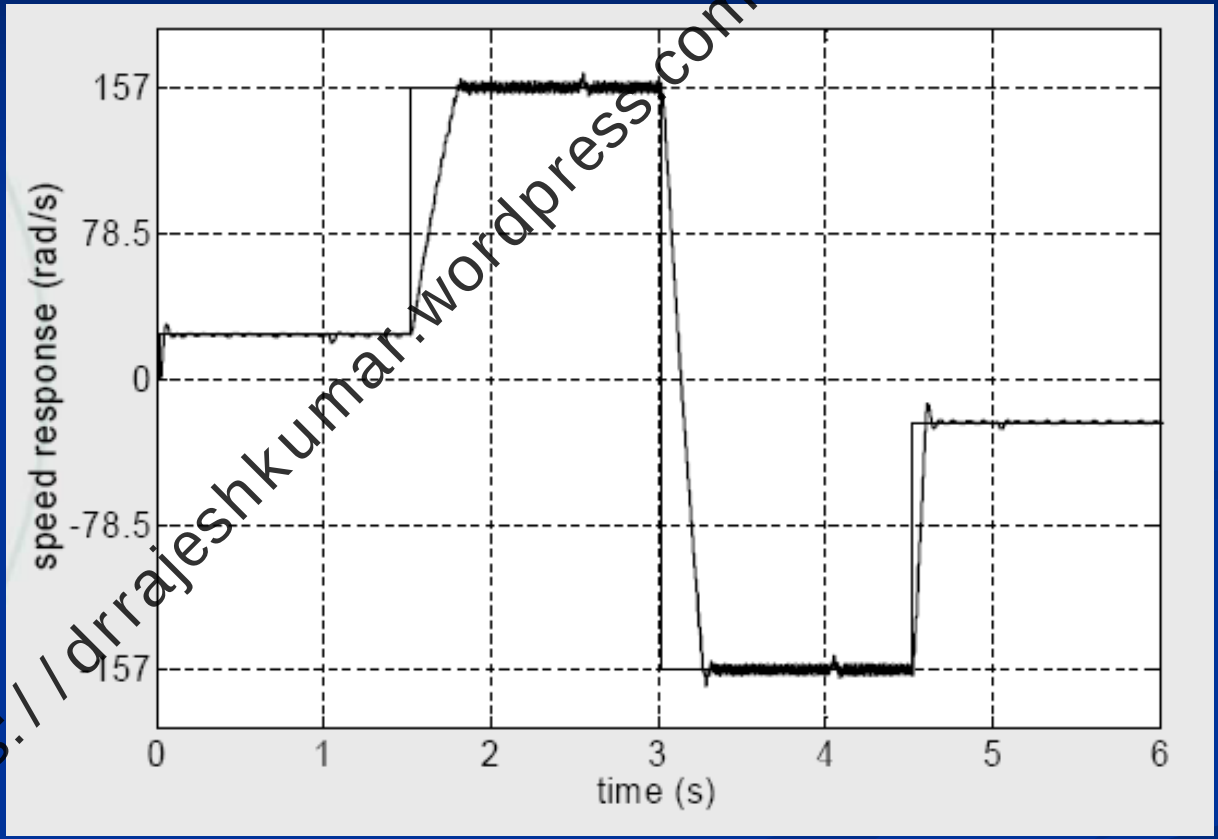


Experimental Setup



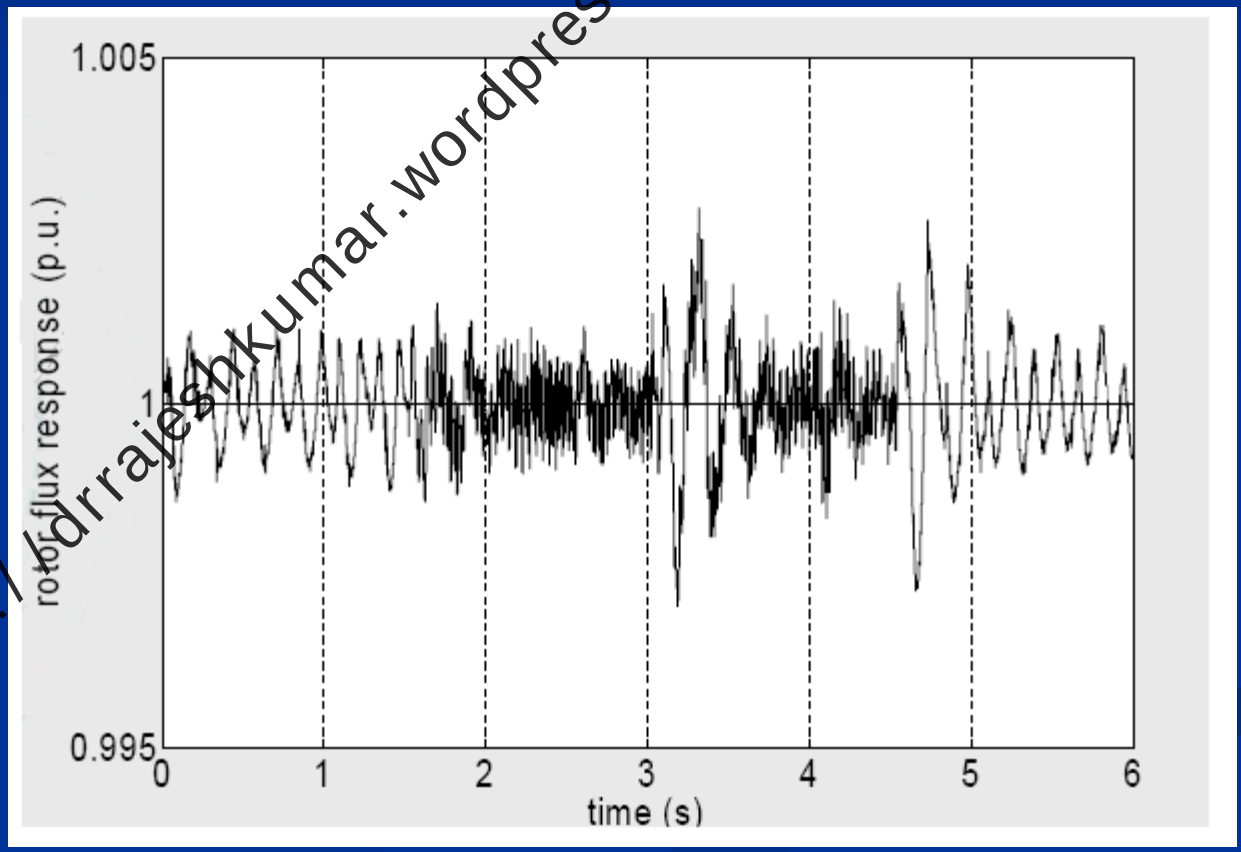
On-line optimization with standard GA: Induction Motor Drive

Speed response at the end of the GA optimization

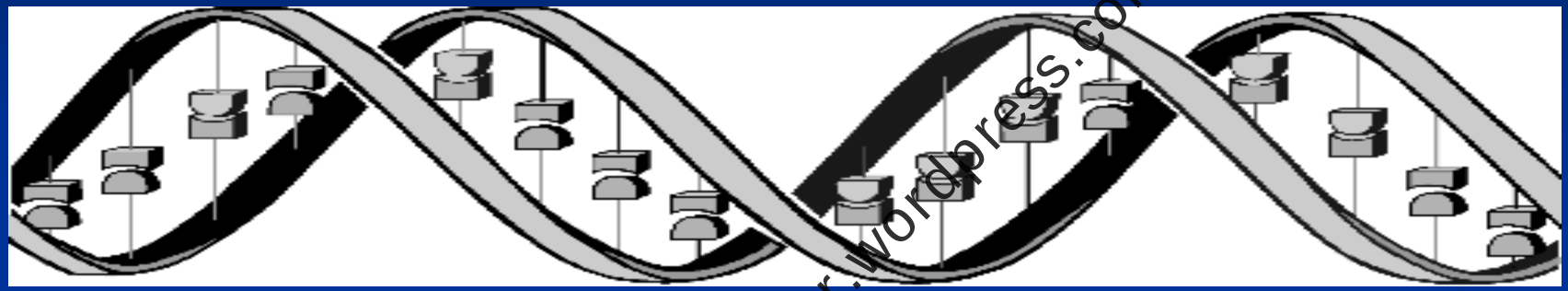


On-line optimization with standard GA: Induction Motor Drive

- Rotor flux response at the end of the GA optimization



Conclusions



Question: *'If GAs are so smart, why ain't they rich?'*

Answer: *'Genetic algorithms **are** rich - rich in application across a large and growing number of disciplines.'*

- David E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*