Genetic Algorithms

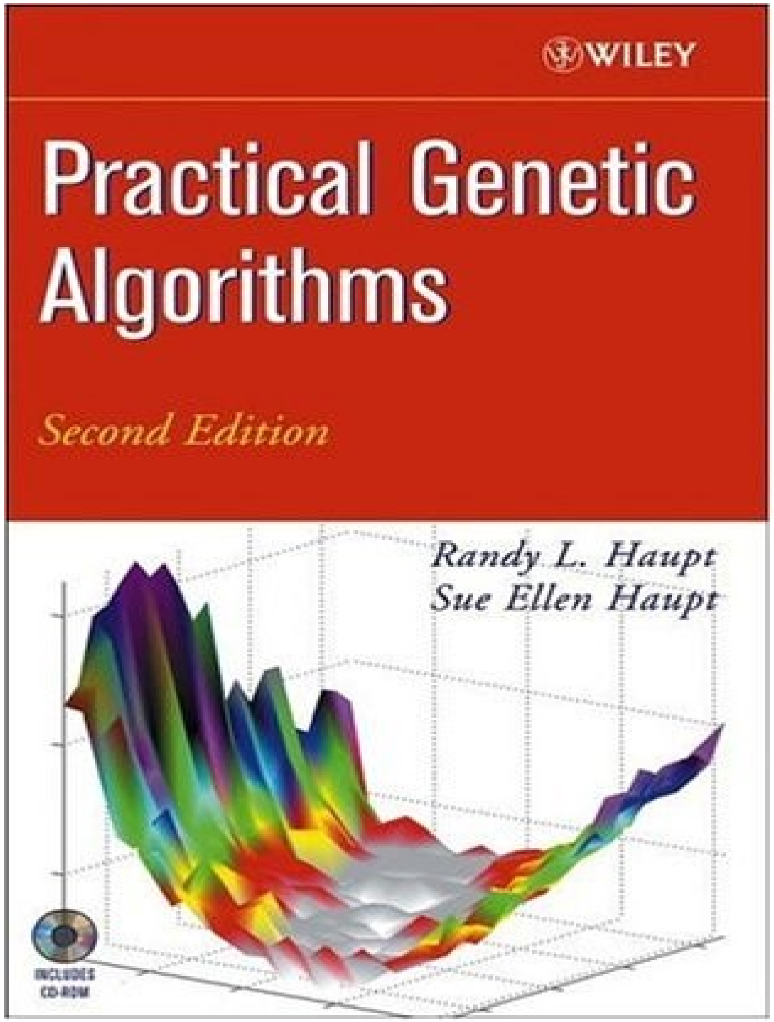
Dr. Bibhudatta Sahoo

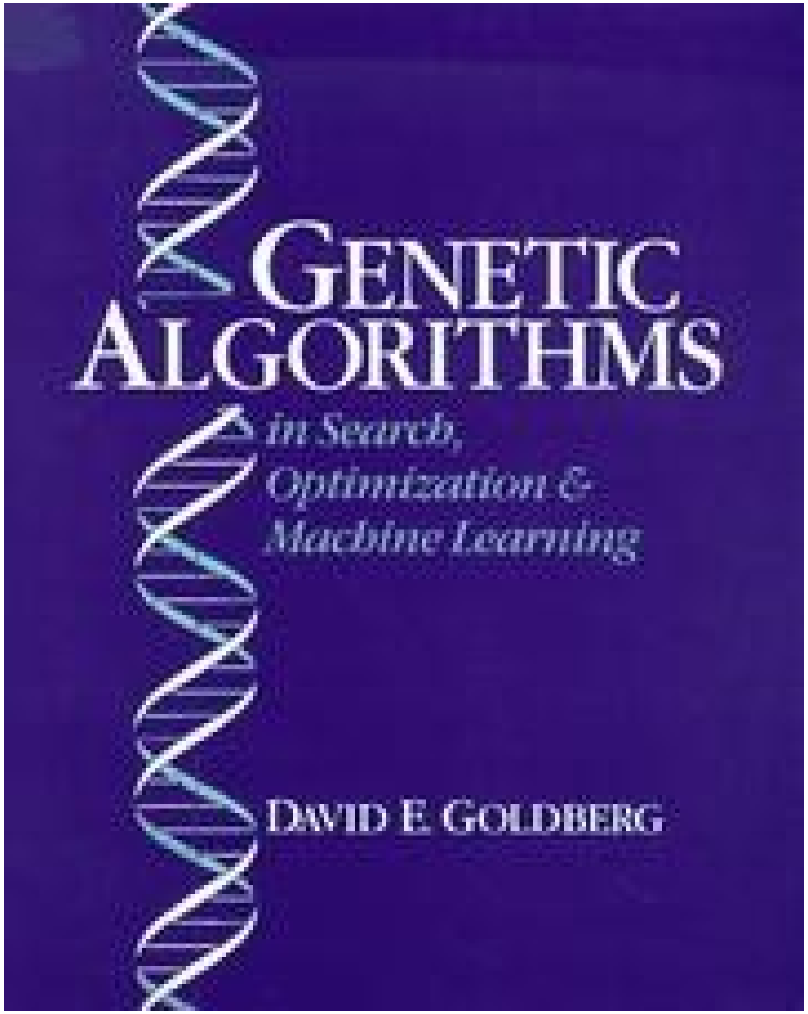
Communication & Computing Group

 # CS215, Department of CSE, NIT Rourkela

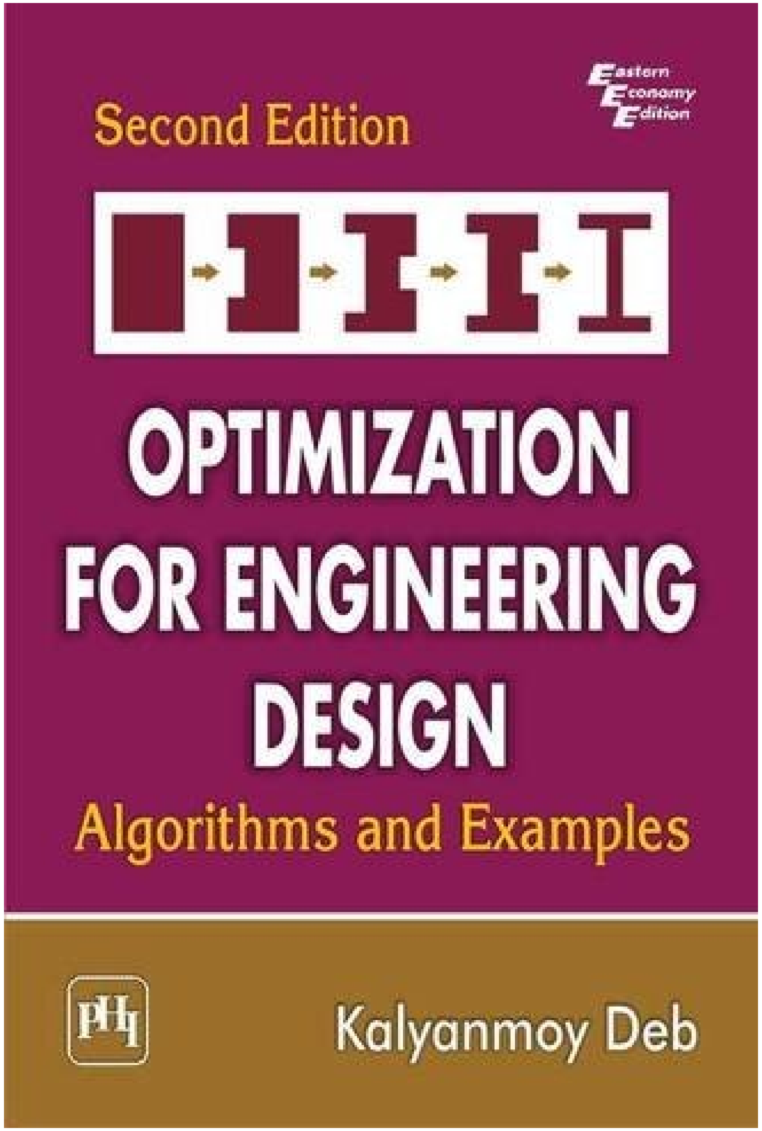
Email: bdsahu@nitrkl.ac.in, 9937324437, 2462358

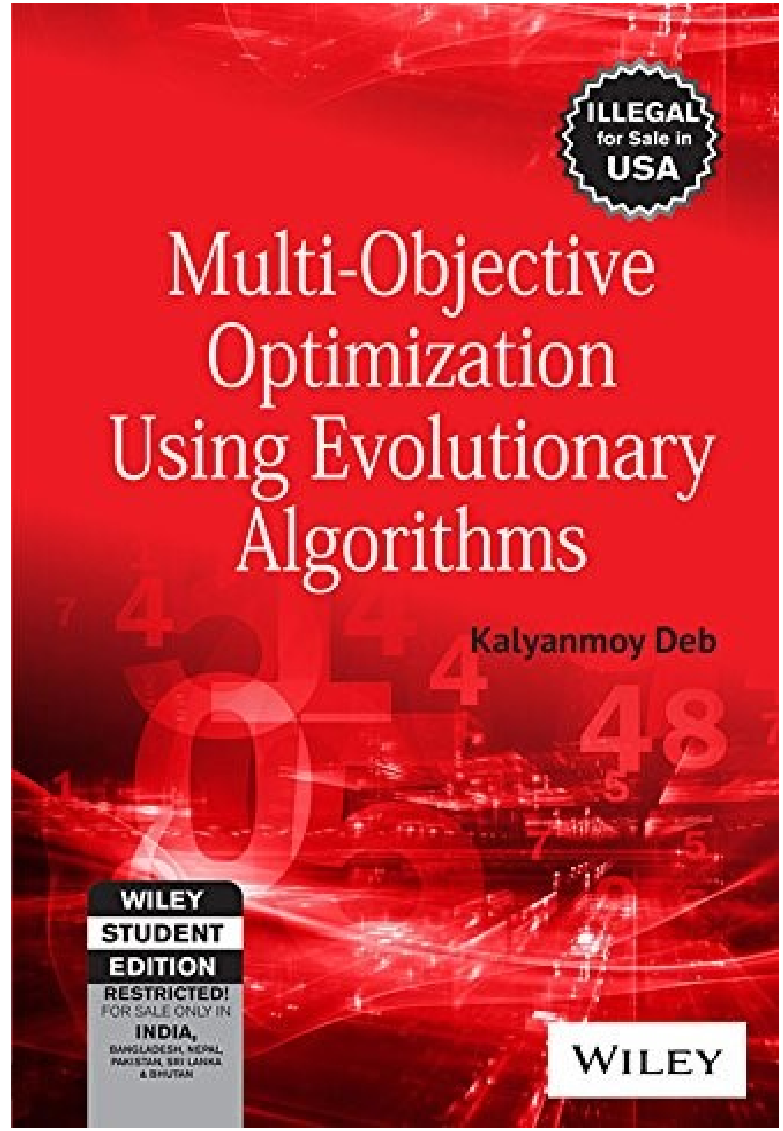
Suggested Reading



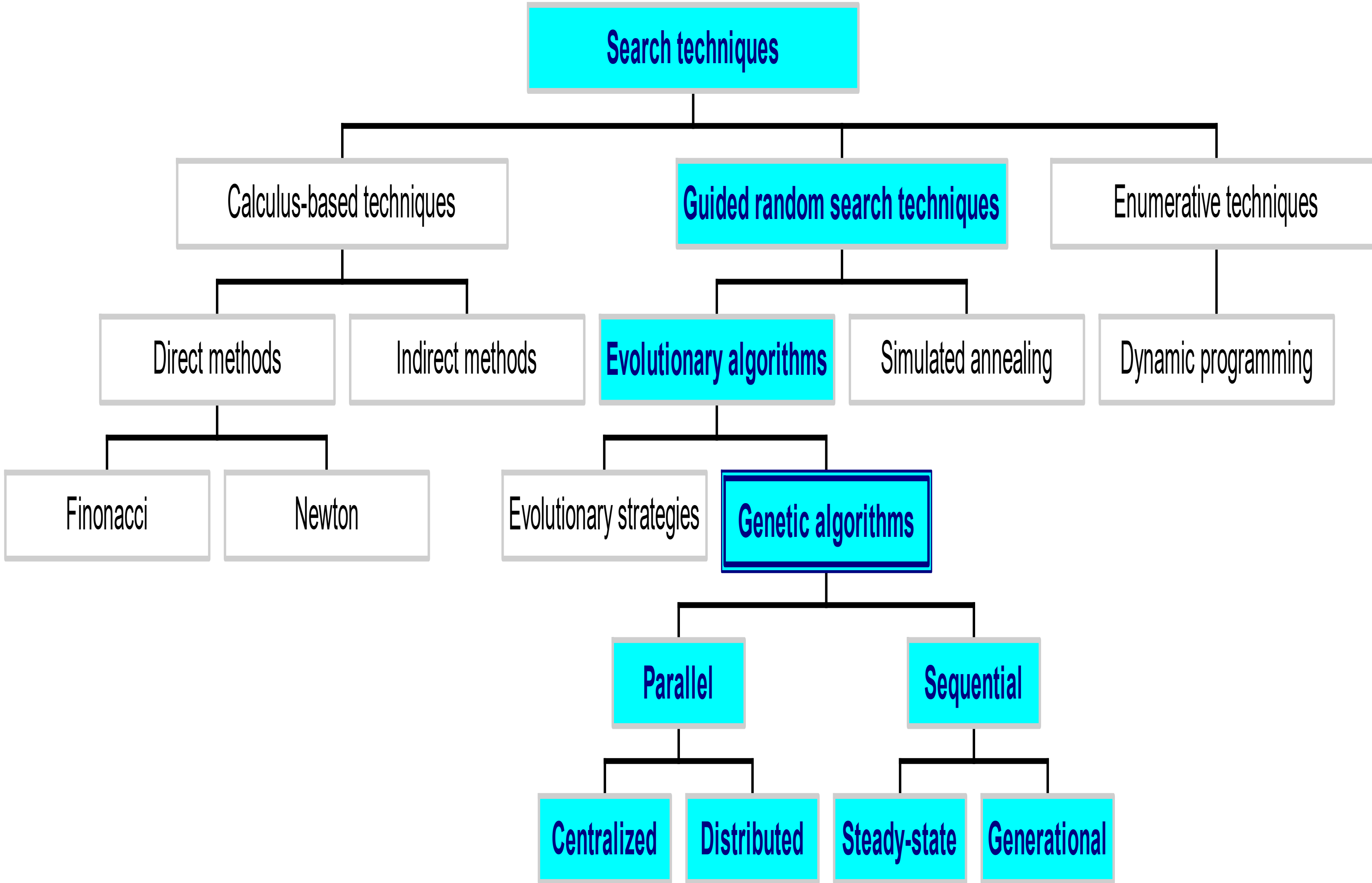


Suggested Reading



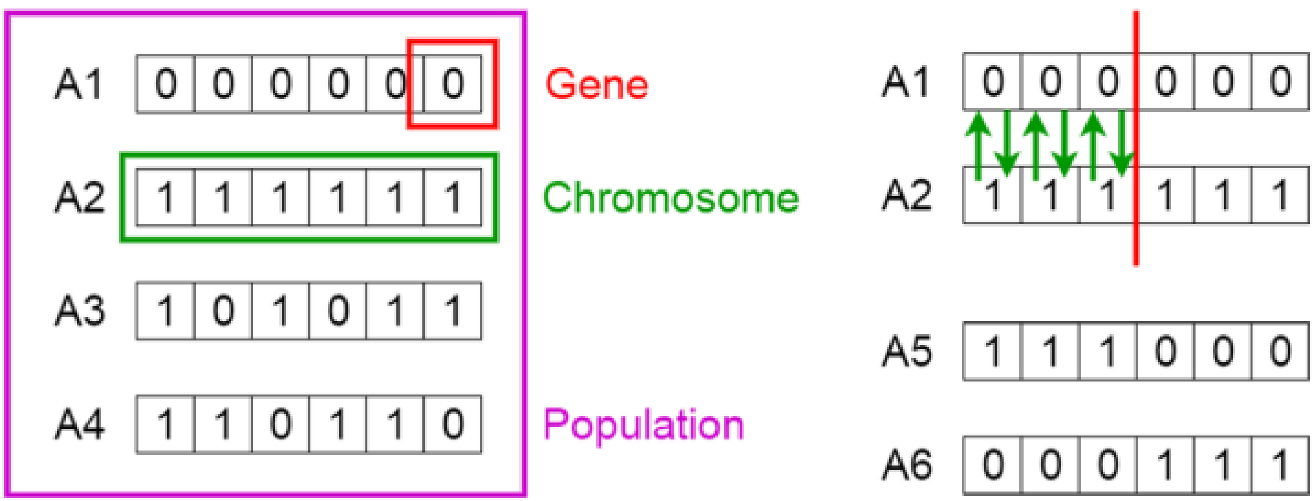


Classes of Search Techniques



An over view of Genetic Search(1)

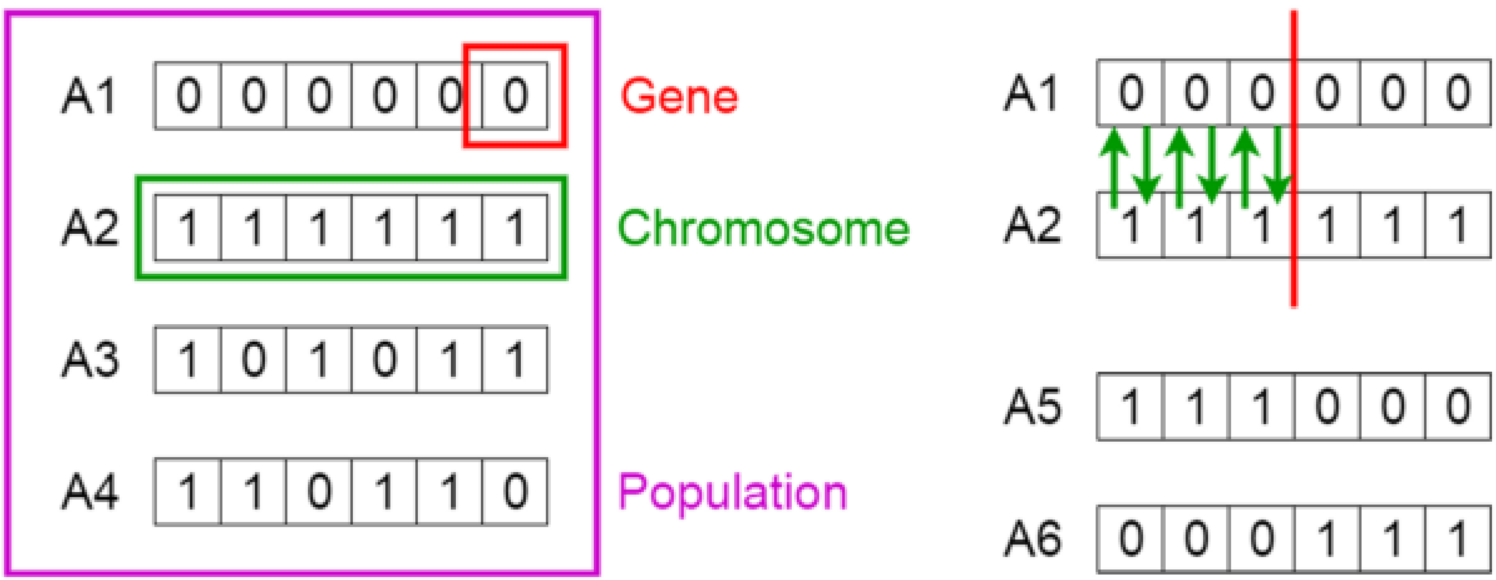
- GAs, differ from more traditional search algorithms in that they work with a number of candidate solutions rather than just one candidate solution or partial solution.
- Each candidate solution of a problem is represented by a data structure known as an **individual**.
- An individual has two parts: a chromosome and a fitness.
- The chromosome of an individual is made up of genes.



Prof. Bibhudatta Sahoo, Department of CSE, NIT Rourkela, 2008

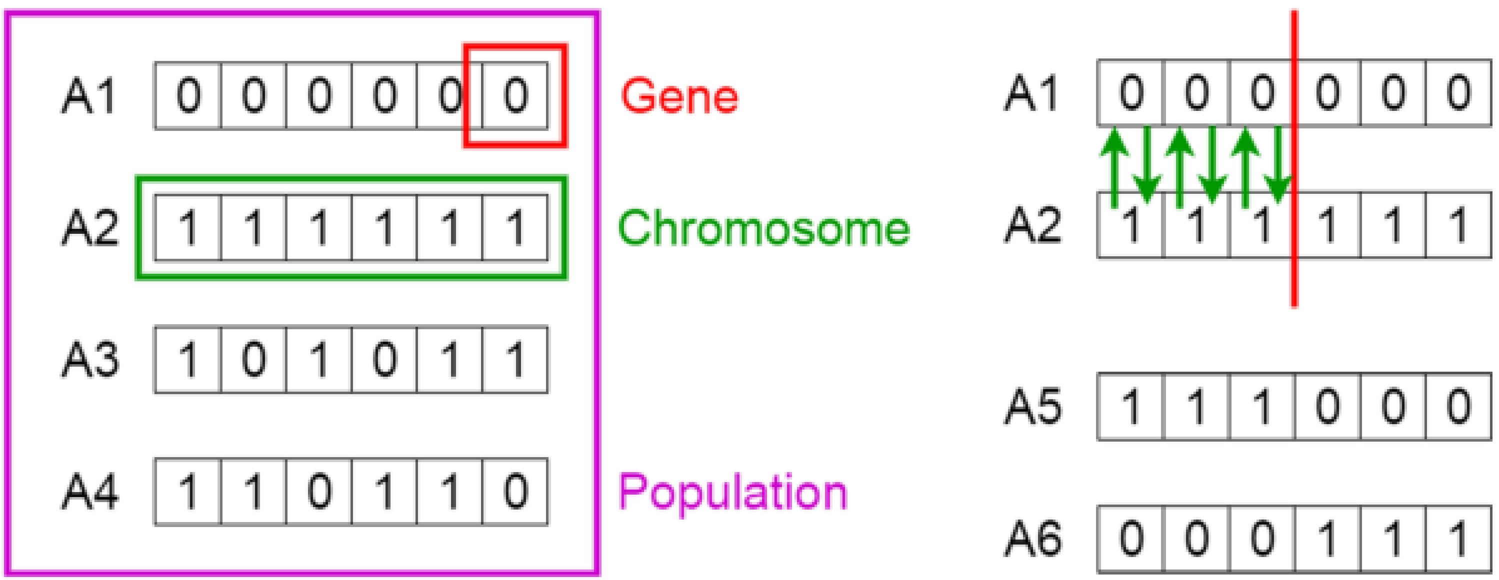
Genetic algorithm (GA)

- An individual is characterized by a set of parameters (variables) known as **Genes**.
- Genes are joined into a string to form a **Chromosome** (solution).
- A set of individuals which is called a **Population**. Each individual is a solution to the problem you want to solve.



Genetic algorithm (GA)

- In a genetic algorithm, the **set of genes** of an individual is represented using a string, in terms of an alphabet.
- Usually, binary values are used (string of 1s and 0s). We say that we encode the genes in a chromosome.



An over view of Genetic Search(2)

- The values that can be assigned to a gene of a chromosome are referred to as the alleles of that gene.
- A group of individuals collectively comprise what is known as a population. For most GAs, the size of the population remains constant for the duration of the search.
- Individuals selected from the current population, called parents, are selected based on their fitness and are allowed to create offspring.

An over view of Genetic Search(3)

- Usually, individuals with above average fitness have an above average chance of being selected.
- After selection, reproductive operators such as crossover and mutation are applied to the parents.
- In crossover, parents contribute copies of their genes to create a chromosome for an offspring.
- Mutation requires only one parent. An offspring created by mutation usually resembles its parent with the exception of a few altered genes.

An over view of Genetic Search(4)

- After the children have been created, the candidate solutions that they represent are evaluated and each child receives a fitness.
- Before the children can be added to the population, some individuals in the current population must die and be removed to make room for the children.
- Usually, individuals are removed based on their fitness with below average individuals having an above average chance of being selected to die.
- This process of allowing individuals to procreate or die based on their relative fitness is called natural selection.
- Individuals that are better fit are allowed to live longer and procreate more often.

An over view of Genetic Search(5)

- An interesting aspect of GAs (and EC in general) is that the initial population of individuals need not be very good.
- Each individual of an initial population usually represents a randomly generated candidate solution.
- By repeatedly applying selection and reproduction, GAs evolve satisfactory solutions quickly and efficiently.

Characteristic of Genetic Algorithms

GAs can be characterized based in terms of **eight** basic attributes:

(1) the genetic representation of candidate solutions

(2) the population size,

(3) the evaluation function,

(4) the genetic operators,

(5) the selection algorithm,

(6) the generation gap,

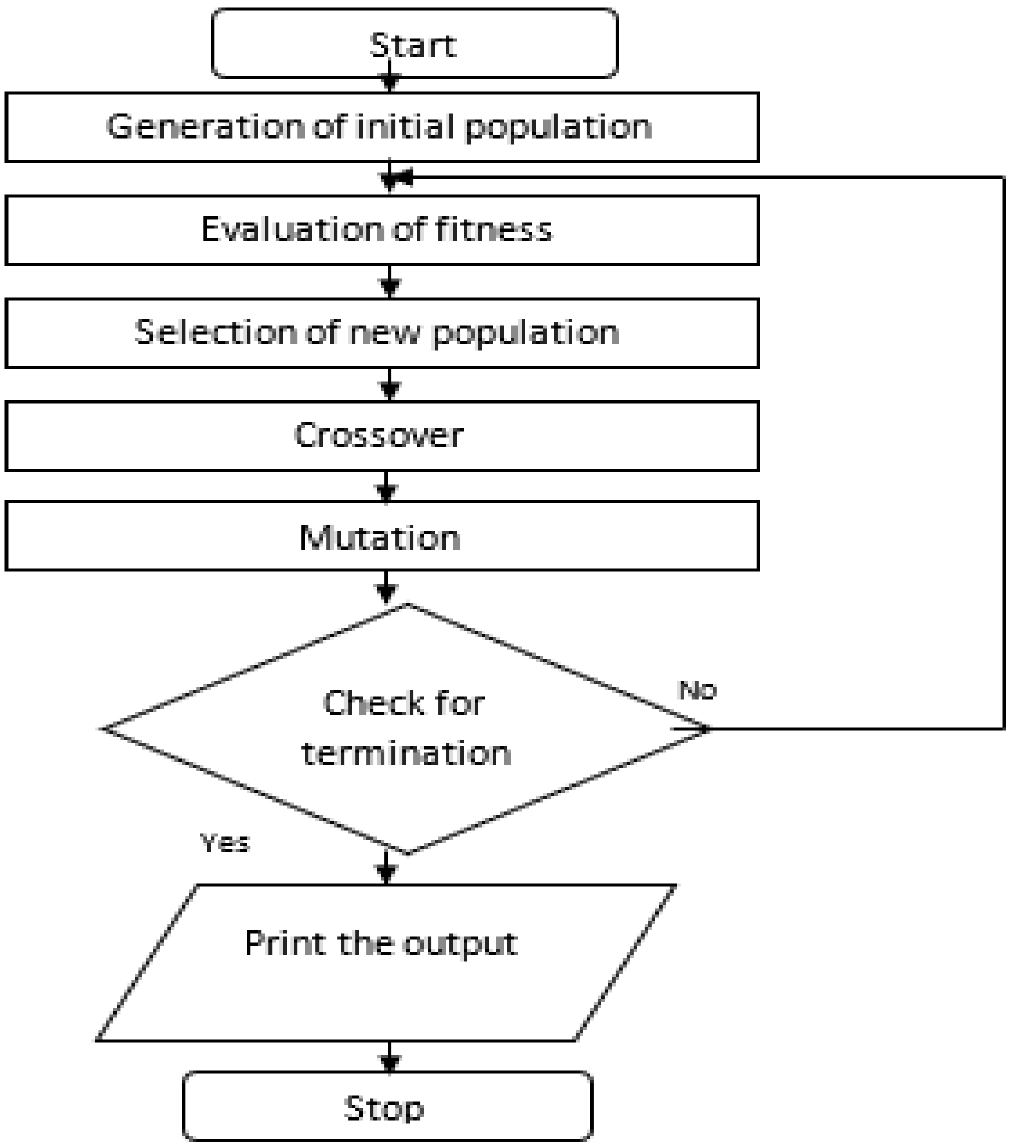
(7) the amount of elitism used

(8) the number of duplicates allowed.

The steps of genetic algorithm

- Step 1: Generate random population of n chromosomes.
- Step 2: Evaluate the fitness $f_{(x)}$ of each chromosome x in the population.
- Step 3: Create a new population by repeating following steps until the new population is complete.
- Step 4: Select two parent chromosomes from a population according to the fitness.
- Step 5: With a crossover probability, crossover the parents to form a new offspring (Children). If no crossover was performed, offspring is an exact copy of parents.
- Step 6: With a mutation probability, mutate new offspring at each position in chromosome.
- Step 7: Place new offspring in a new population.
- Step 8: Use new generated population for a further run of algorithm.
- Step 9: If the end condition is satisfied, stop and return the best solution in current population and go to step 2.

Flowchart for genetic algorithm



The genetic algorithm

- The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution.
- The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation.
- Over successive generations, the population "evolves" toward an optimal solution.
- You can apply the genetic algorithm to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, non-differentiable, stochastic, or highly nonlinear.
- The genetic algorithm can address problems of *mixed integer programming*, where some components are restricted to be integer-valued.

The genetic algorithm

- The genetic algorithm uses three main types of rules at each step to create the next generation from the current population:
- *Selection rules* select the individuals, called *parents*, that contribute to the population at the next generation.
- *Crossover rules* combine two parents to form children for the next generation.
- *Mutation rules* apply random changes to individual parents to form children.
- The genetic algorithm differs from a classical, derivative-based, optimization algorithm in two main ways, as summarized in the following table

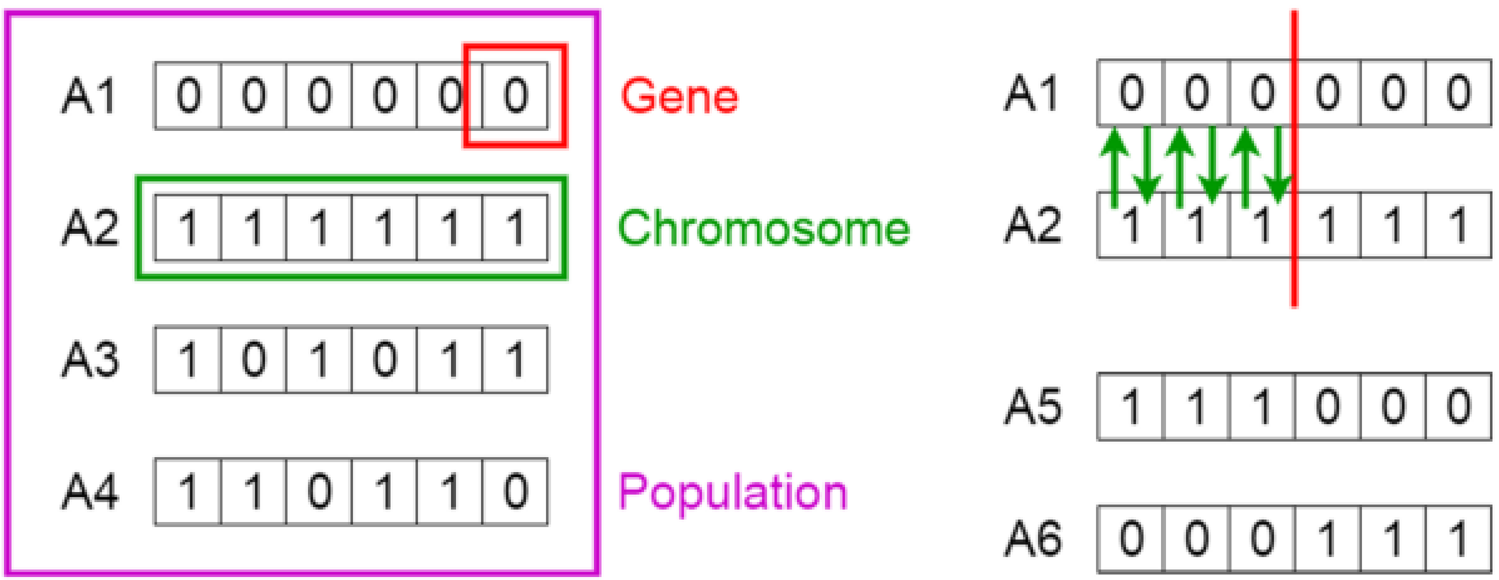
Genetic algorithm (GA)

- A genetic algorithm (GA) is a method for solving both constrained and unconstrained optimization problems based on a natural selection process that mimics biological evolution.
- The algorithm repeatedly modifies a population of individual solutions.
- At each step, the genetic algorithm randomly selects individuals from the current population and uses them as parents to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution.

Classical Algorithm	Genetic Algorithm
Generates a single point at each iteration. The sequence of points approaches an optimal solution.	Generates a population of points at each iteration. The best point in the population approaches an optimal solution
Selects the next point in the sequence by a deterministic computation.	Selects the next population by computation which uses random number generators

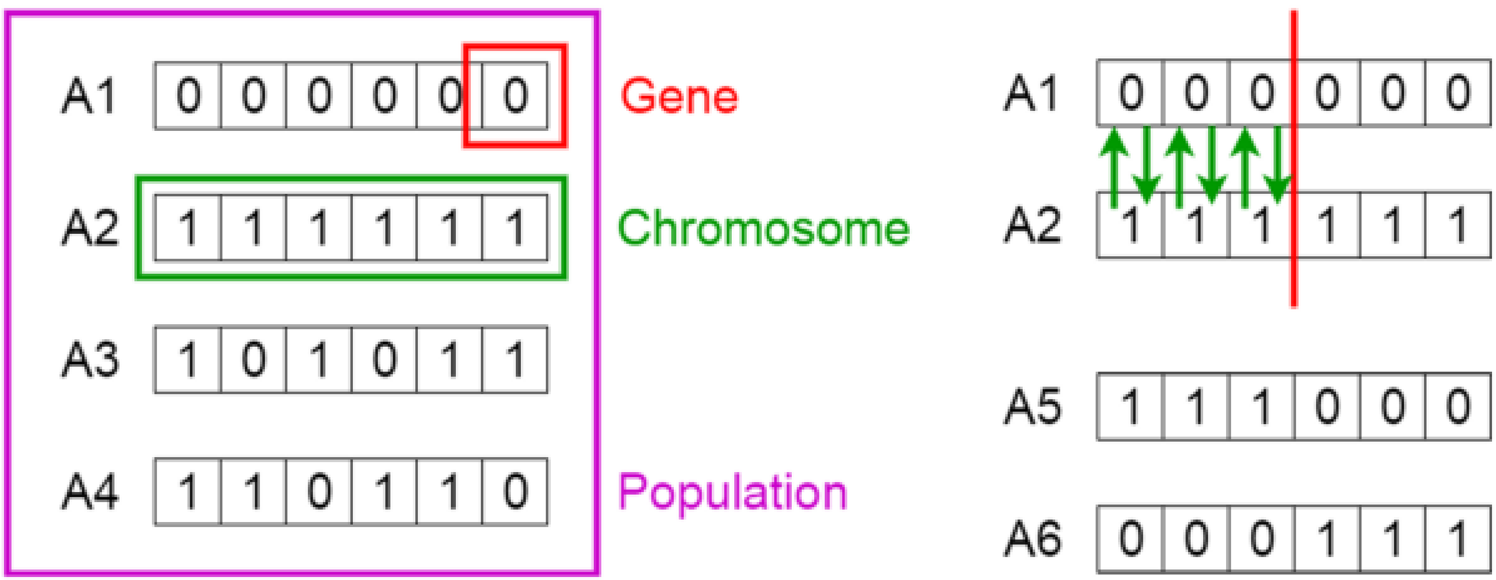
Genetic algorithm (GA)

- An individual is characterized by a set of parameters (variables) known as **Genes**.
- Genes are joined into a string to form a **Chromosome** (solution).
- A set of individuals which is called a **Population**. Each individual is a solution to the problem you want to solve.



Genetic algorithm (GA)

- In a genetic algorithm, the **set of genes** of an individual is represented using a string, in terms of an alphabet.
- Usually, binary values are used (string of 1s and 0s). We say that we encode the genes in a chromosome.



Unconstrained Optimization Problem

Example :Unconstrained Optimization Problem

$$f(x_1, x_2) = \left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2$$

$$0 \le x_1, x_2 \le 6$$

- Binary coding for x_1 and x_2, each of 10 bit
- Roulette wheel selection
- A single point cross over
- Bit wise mutation
- Population size 20
- Maximum number of generation 30
- Cross over and mutation probability to be 0.8 and 0.05

Prof. Bibhudatta Sahoo, Department of CSE, NIT Rourkela, 2008

Unconstrained Optimization Problem

Table 6.1 Evaluation and Reproduction Phases on a Random Population

	String												Mating pool	
	Substring-2	Substring-1	x_2	x_1	$f(x)$	$\mathcal{F}(x)$	A	B	C	D	E	F	Substring-2	Substring-1
1	1110010000	1100100000	5.349	4.692	959.680	0.001	0.13	0.007	0.007	0.472	10	0	0010100100	1010101010
2	0001001101	0011100111	0.452	1.355	105.520	0.009	1.10	0.055	0.062	0.108	3	1	1010100001	0111001000
3	1010100001	0111001000	3.947	2.674	126.685	0.008	0.98	0.049	0.111	0.045	2	1	0001001101	0011100111
4	1001000110	1000010100	3.413	3.120	65.026	0.015	1.85	0.093	0.204	0.723	14	2	1110011011	0111000010
5	1100011000	1011100011	4.645	4.334	512.197	0.002	0.25	0.013	0.217	0.536	10	0	0010100100	1010101010
6	0011100101	0011111000	1.343	1.455	70.868	0.014	1.71	0.086	0.303	0.931	19	2	0011100010	1011000011
7	0101011011	0000000111	2.035	0.041	88.273	0.011	1.34	0.067	0.370	0.972	19	1	0011100010	1011000011
8	1110101000	1110101011	5.490	5.507	1436.563	0.001	0.12	0.006	0.376	0.817	17	0	0111000010	1011000110
9	1001111101	1011100111	3.736	4.358	265.556	0.004	0.49	0.025	0.401	0.363	7	1	0101011011	0000000111
10	0010100100	1010101010	0.962	4.000	39.849	0.024	2.96	0.148	0.549	0.189	4	3	1001000110	1000010100
11	1111101001	0001110100	5.871	0.680	814.117	0.001	0.14	0.007	0.556	0.220	6	0	0011100101	0011111000
12	0000111101	0110011101	0.358	2.422	42.598	0.023	2.84	0.142	0.698	0.288	6	3	0011100101	0011111000
13	0000111110	1110001101	0.364	5.331	318.746	0.003	0.36	0.018	0.716	0.615	12	1	0000111101	0110011101
14	1110011011	0111000010	5.413	2.639	624.164	0.002	0.24	0.012	0.728	0.712	13	1	0000111110	1110001101
15	1010111010	1010111000	4.094	4.082	286.800	0.003	0.37	0.019	0.747	0.607	12	0	0000111101	0110011101
16	0100011111	1100111000	1.683	4.833	197.556	0.005	0.61	0.030	0.777	0.192	4	0	1001000110	1000010100
17	0111000010	1011000110	2.639	4.164	97.699	0.010	1.22	0.060	0.837	0.386	9	1	1001111101	1011100111
18	1010010100	0100001001	3.871	1.554	113.201	0.009	1.09	0.054	0.891	0.872	18	1	1010010100	0100001001
19	0011100010	1011000011	1.326	4.147	57.753	0.017	2.08	0.103	0.994	0.589	12	2	0000111101	0110011101
20	1011100011	1111010000	4.334	5.724	987.955	0.001	0.13	0.006	1.000	0.413	10	0	0010100100	1010101010

A : Expected count
B : Probability of selection
C : Cumulative probability of selection
D : Random number between 0 and 1
E : String number
F : True count in the mating pool

Unconstrained Optimization Problem

Table 6.2 Crossover and Mutation Operators

Mating pool				Intermediate population		Mutation					
Substring-2	Substring-1	G	H	Substring-2	Substring-1	Substring-2	Substring-1	x_1	x_2	$f(x)$	$\mathcal{F}(x)$
0010100100	1010101010	Y	9	0010100101	0111001000	001010$\boxed{1}$101	0111001000	1.015	2.674	18.886	0.050
1010100001	0111001000	Y	9	1010100000	1010101010	101010000$\boxed{1}$	1010101010	3.947	4.000	238.322	0.004
0001001101	0011100111	Y	12	0001001101	0011000010	0001001101	00$\boxed{0}$1000010	0.452	0.387	149.204	0.007
1110011011	0111000010	Y	12	1110011011	0111100111	1110011011	01$\boxed{0}$1100$\boxed{0}$11	5.413	2.082	596.340	0.002
0010100100	1010101010	Y	5	0010100010	1011000011	0010100010	1011000011	0.950	4.147	54.851	0.018
0011100010	1011000011	Y	5	0011100100	1010101010	0011100100	1$\boxed{1}$10101010	1.337	5.501	424.583	0.002
0011100010	1011000011	N		0011100010	1011000011	001110001$\boxed{1}$	1011$\boxed{1}$00011	1.331	4.334	83.929	0.012
0111000010	1011000110	N		0111000010	1011000110	01$\boxed{0}$10$\boxed{1}$0010	1011000110	1.982	4.164	70.472	0.014
0101011011	0000000111	Y	14	0101011011	0000010100	0101011011	0000010100	2.035	0.117	87.633	0.011
1001000110	1000010100	Y	14	1001000110	1000000111	10010$\boxed{1}$0110	1000000111	3.507	3.044	72.789	0.014
0011100101	0011111000	Y	1	0011100101	0011111000	0011100101	0011111000	1.343	1.455	70.868	0.014
0011100101	0011111000	Y	1	0011100101	0011111000	0011100101	0011111000	1.343	1.455	70.868	0.014
0000111101	0110011101	N		0000111101	0110011101	00001$\boxed{0}$1101	011$\boxed{1}$01110$\boxed{0}$	0.264	2.792	25.783	0.037
0000111110	1110001101	N		0000111110	1110001101	0000111110	1110001101	0.364	5.331	318.746	0.003
0000111101	0110011101	Y	18	0000111101	0110011100	0000111101	0110011100	0.358	2.416	42.922	0.023
1001000110	1000010100	Y	18	1001000110	1000010101	1001000110	$\boxed{0}$000010101	3.413	0.123	80.127	0.012
1001111101	1011100111	Y	10	1001111101	0100001001	1001111101	0100001001	3.736	1.554	95.968	0.010
1010010100	0100001001	Y	10	1010010100	1011100111	1010010100	101$\boxed{0}$100111	3.871	3.982	219.426	0.005
0000111101	0110011101	N		0000111101	0110011101	0000111101	0110011101	0.358	2.422	42.598	0.023
0010100100	1010101010	N		0010100100	1010101010	0010100100	1010101010	0.962	4.000	39.849	0.024

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

Prof. Bibhudatta Sahoo, Department of CSE, NIT Rourkela, 2008

Find global minima for highly nonlinear problems

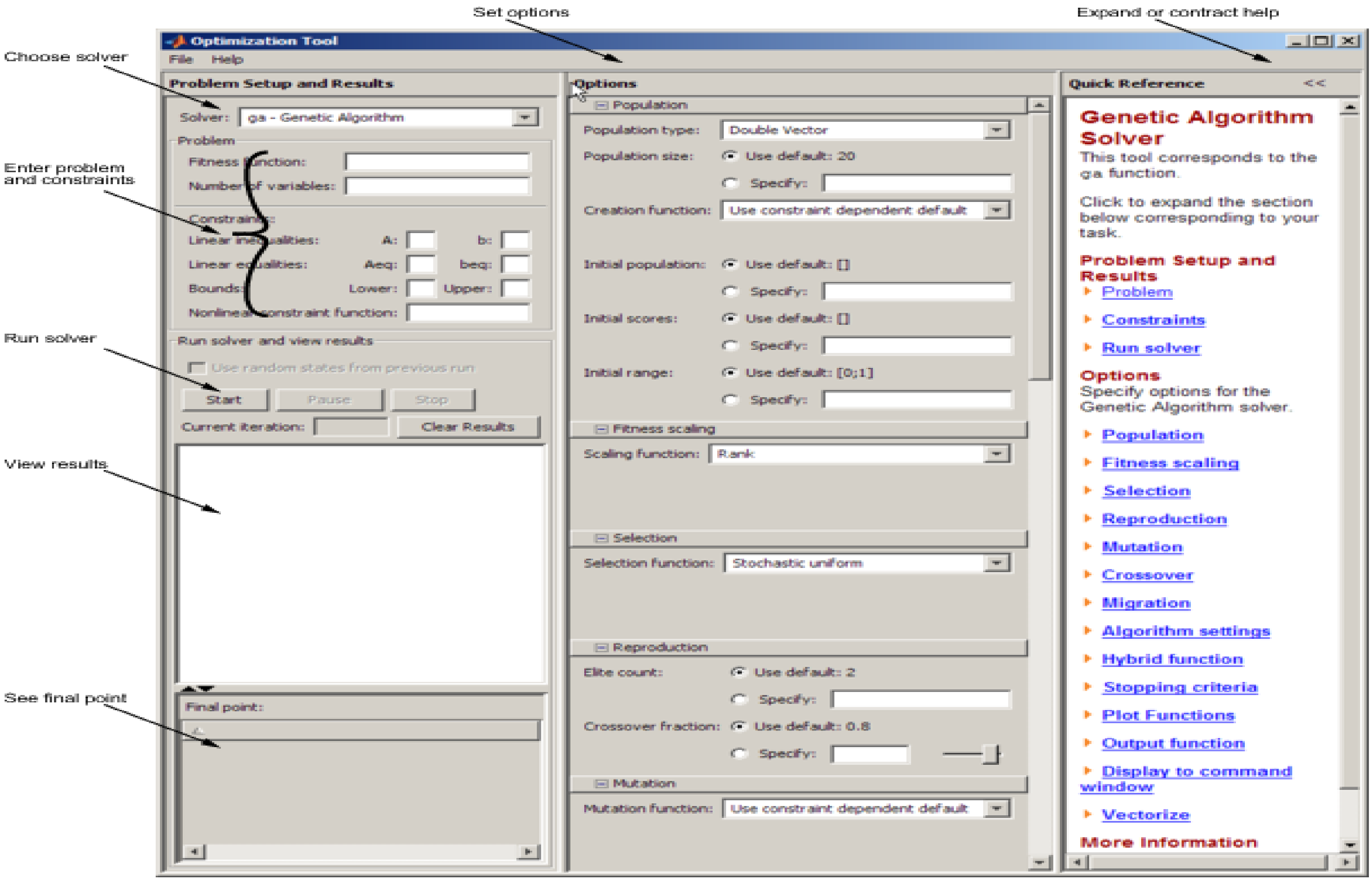
- **Genetic algorithm solver** for mixed-integer or continuous-variable optimization, constrained or unconstrained
- Genetic algorithm solves smooth or non-smooth optimization problems with any types of constraints, including integer constraints. It is a stochastic, population-based algorithm that searches randomly by mutation and crossover among population members.
- A genetic algorithm (GA) is a method for solving both constrained and unconstrained optimization problems based on a natural selection process that mimics biological evolution.
- The algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm randomly selects individuals from the current population and uses them as parents to produce the children for the next generation.
- Over successive generations, the population "evolves" toward an optimal solution.

Using Genetic algorithm tool box : optimtool('ga')

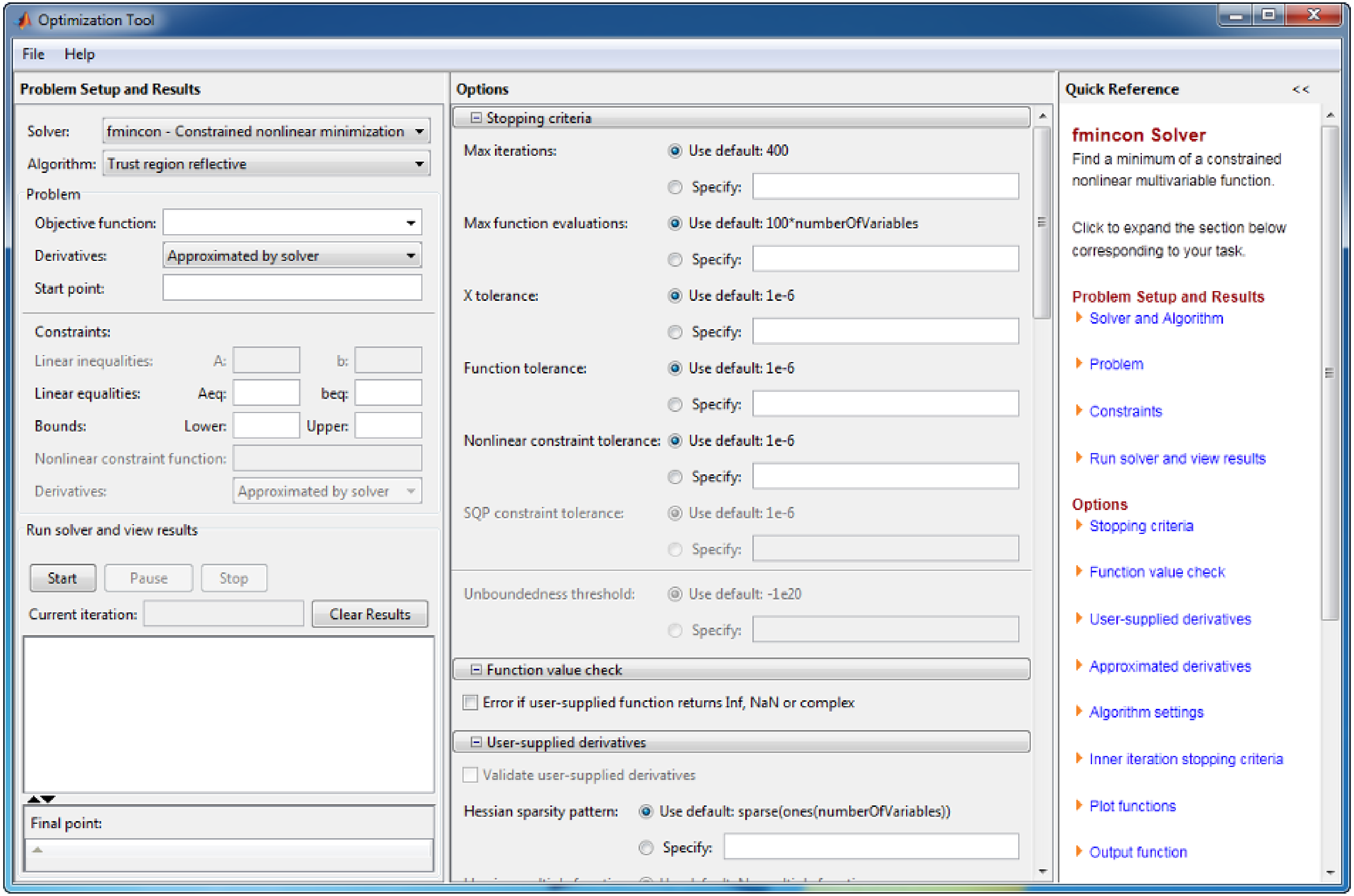
- To use the genetic algorithm at the command line, call the genetic algorithm function ga with the syntax
- [**x** **fval**] = **ga**(**@fitnessfun, nvars, options**)
- where
- **@fitnessfun** is a handle to the fitness function.
- **nvars** is the number of independent variables for the fitness function.
- **options** is a structure containing options for the genetic algorithm.
- If you do not pass in this argument, ga uses its default options.
- The results are given by
  - **x** — Point at which the final value is attained
  - **fval** — Final value of the fitness function

Prof. Bibhudatta Sahoo, Department of CSE, NIT Rourkela, 2008

Calling the Function GA at the Command Line



Calling the Function GA at the Command Line



How to use the Optimization Tool

- **Fitness function** —Enter the fitness function in the form @fitnessfun, where fitnessfun.m is a file that computes the fitness function. The @ sign creates a function handle to fitnessfun.
- **Number of variables** — The length of the input vector to the fitness function.
- You can enter constraints or a nonlinear constraint function for the problem in the **Constraints** pane. If the problem is unconstrained, leave these fields blank.
- To run the genetic algorithm, click the **Start** button. The tool displays the results of the optimization in the **Run solver and view results** pane.

- https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3
- https://brighterion.com/artificial-intelligence-101-genetic-algorithms/
- https://www.tutorialspoint.com/genetic_algorithms/index.htm


Colourbox
Thank you for your attention
Genetic Algorithms

Exercises