Lecture - 09
GLOBAL OPTIMIZATION BASICS

Reference: Book Chapter 8

INTRODUCTION

- The ***objective of global optimization*** is to find the best **global solution** in the possible or known presence of multiple local optima.

- Global optimization seeks a global solution to an unconstrained or constrained optimization problem.

- Global search techniques are often essential for many applications, e.g., advanced engineering design, data analysis, financial planning, process control, risk management, and scientific modeling.

INTRODUCTION

A global optimization problem can be defined as

$$\min_{x} \quad f(x) \longleftarrow \text{objective function}$$

Subject to:

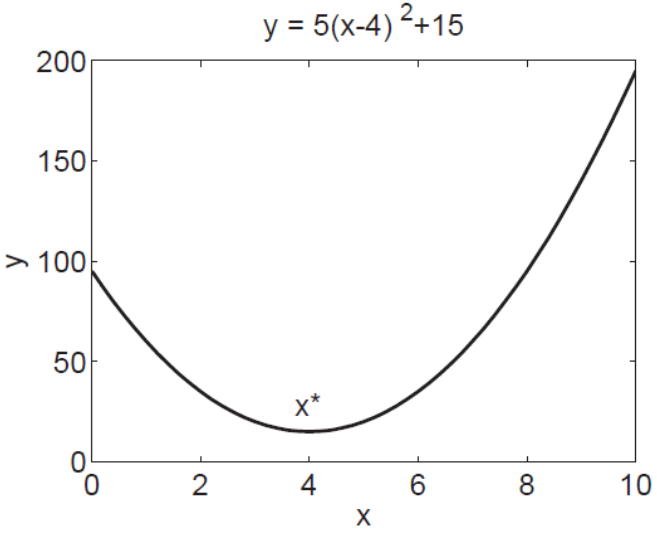
$$g(x) \leq 0 \longleftarrow \text{inequality constraint function}$$

$$h(x) = 0 \longleftarrow \text{equality constraint}$$

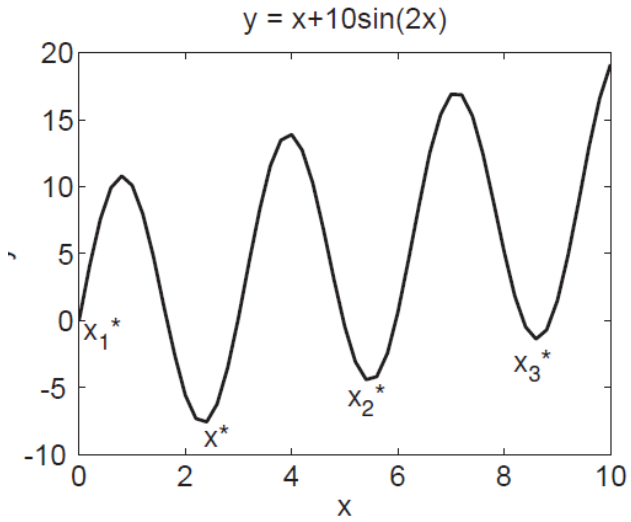
$$x_l \leq x \leq x_u \longleftarrow \text{side constraints}$$

PRACTICAL ISSUES IN GLOBAL OPTIMIZATION

- In ***unimodal*** optimization problem, objective function only has one local minimum, which is also its global minimum.
- In ***multimodal*** optimization problem, objective functions can have global minima and local minima.
- The objective of global optimization is to find ***global optima***.



Unimodoal Function



Multimodal Function

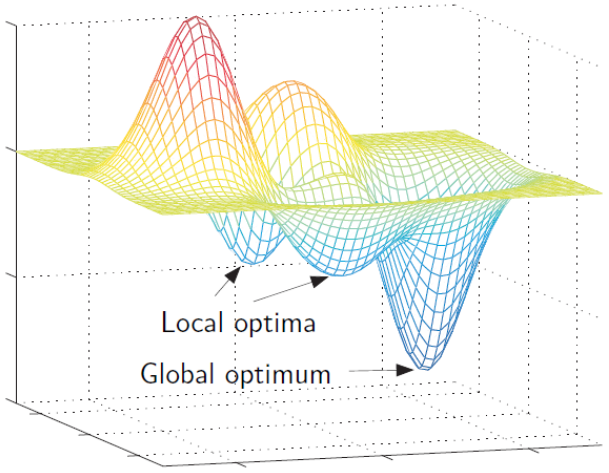
PRACTICAL ISSUES IN GLOBAL OPTIMIZATION

- In a unimodal objective function :

The global minimum can be found by using the **gradient-based optimization methods** (Chapters 14 and 15).

- In a multimodal objective function :

Gradient-based optimization methods will simply yield one of the local minima, which depends on which starting point was used. Finding the global optimum is **not guaranteed** when using gradient based algorithms.



PRACTICAL ISSUES IN GLOBAL OPTIMIZATION

In practical global optimization problems

- The derivatives of objective functions (at some points or in certain intervals) may not be available, or may be expensive to compute.

- It is often even challenging to determine whether a highly nonlinear function is unimodal or multimodal before starting the optimization.

Then

Gradient-based optimization methods may not be appropriate for such global optimization problems. **Derivative-free methods**, such as **evolutionary algorithms**, may be more appropriate in these cases.

EXHAUSTIVE SEARCH

- Exhaustive search enumerates all possible candidates for optimization problems, and the **best solution is the global optimum**.
- Exhaustive search can be applicable to solve optimization problems that comprise a **small number of variable combinations**.
- The **time required** for conducting an exhaustive search is dramatically increased when the number of candidates increases.

EXHAUSTIVE SEARCH

Example

$$\min \quad f(x) = 5(x_1 - 4)^2 + 4(x_2 - 2)^2$$

Subject to

$$x_1 \in \{1, 3, 5.5\}$$
$$x_2 \in \{2, 3.5, 5.5\}$$

Each of the design variables has three possible values. So, there are nine possible combinations.

Design Variable Combination

x_1	1	1	1	3	3	3	5.5	5.5	5.5
x_2	2	3.5	5.5	2	3.5	5.5	2	3.5	5.5
$f(x)$	45	54	94	5	14	54	11.25	20.25	60.25

Objective Function Value

The optimal solution is : x1 = 3 and x2 = 2

The global minimum of f(x) is : 5

MULTIPLE START

- The multiple start approach uses gradient-based methods to find local and global minima.
- It generates multiple starting points, and stores local and global solutions found during the search process.

MULTIPLE START

- The basic **multiple start method** consists of the following steps:

1. ***Generate multiple starting points.***
   The points can either be uniformly distributed within predefined bounds, or generated using a sampling algorithm.

2. ***Filter out the infeasible starting points.***
   This step may help reduce the total computational expense required to find the corresponding feasible optima.

3. ***Use a gradient-based optimization.***
   For each starting point, use a gradient-based optimization method to search for a local minimum.

4. ***Save the multiple local minima returned from Step 3.***

5. ***Compare all the local minima.***
   The local minimum with the lowest objective function value is considered the global minimum.

MULTIPLE START

Example

$$\min \quad f(x) = x + 10sin(2x)$$

Subject to

$$0 \leq x \leq 10$$

1. Choose 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10 as starting points.

2. Use fmincon to solve the optimization problem corresponding to each starting point.

Main Function (to be continued)

```
clear
clc
% Define Linear constraints
A=[ ]; B=[ ]; Aeq=[]; Beq=[];
% Define bounds constraints
LB=[0]; UB=[10];
```

MULTIPLE START

Main Function (to be continued)

```
% Define starting points
xsta = [0 1 2 3 4 5 6 7 8 9 10];
k=11;
% Optimization from 11 starting points
for i=1:1:k
x0=xsta(i)
[xopt,fopt(i)]=fmincon('MS_Func', x0, A, B, ...
Aeq, Beq, LB, UB, 'MS_cons', options)
end
```

Objective Function

```
function [f]=MS_Func(x)
f = x+10*sin(2*x);
```

Constraint Function

```
function [C Ceq]=MS_cons(x)
C = [];
Ceq = [];
```

MULTIPLE START

3 and 4. Save the eleven local optima for 11 Starting Points

x_0	0	1	2	3	4	5	6	7	8	9	10
x^*	0	8.614	2.331	0	5.473	5.473	0	0	8.614	0	2.331
$f^*(x)$	0	-1.373	-7.656	0	-4.515	-4.515	0	0	-1.373	0	-7.656

x_0 Represents starting points

x^* Represents corresponding local optimum

$f^*(x)$ Represents the corresponding local optima of f(x)

5. Compare the local optima given by the multiple start method

The global minimum is found to be −7.656
and the corresponding optimal value of x is 2.331

EVOLUTIONARY ALGORITHMS

- Evolutionary algorithms imitate living beings or adopt natural selection processes to develop powerful computational algorithms.
- Some of the popular techniques that fall under the umbrella of evolutionary algorithms are :
  - Genetic Algorithms (GA)
  - Simulated Annealing (SA)
  - Ant Colony Optimization (ACO)
  - Particle Swarm Optimization (PSO)
  - Tabu Search (TS)

EVOLUTIONARY ALGORITHMS

- The motivation for using biologically inspired computing ideas stems from:
  1) the limitations of mathematical optimization algorithms in solving complex problems in engineering, computing, and other fields.
  2) the realization that many complex problems encountered already exist in nature.

- Engineers and scientists are now exploring the efficient problem solving techniques employed by nature to solve optimization problems.

EVOLUTIONARY ALGORITHMS

- **The relative advantages and limitations of EAs versus traditional optimization methods are as follows:**

1. Traditional algorithms generally generate a single candidate optimum at each iteration, which progresses towards the optimal solution. Genetic algorithms generate a population of points at each iteration. The best point in the population approaches an optimal solution.
2. Traditional algorithms calculate the candidate optimal point at the next iteration by a deterministic computation. EAs usually select the next population by combination of operations that uses random number generators.
3. Traditional algorithms require gradient and/or hessian information to proceed, whereas EAs usually require only function values. EAs can solve a variety of optimization problems in which the objective function is discontinuous.

EVOLUTIONARY ALGORITHMS

- **The relative advantages and limitations of EAs versus traditional optimization methods are as follows:**

4. A drawback of EAs is that they need more function evaluations than do gradient-based methods. The computational time associated with EAs is longer than that of the gradient-based methods.
5. Evolutionary algorithms are stochastic methods that typically involve random choices. Therefore, different runs of the same EA code might yield different optimal solutions.
6. Evolutionary algorithms have no proofs of convergence to an optimal solution; unlike gradient based methods, where at least a local optimum is guaranteed upon convergence.

ROLE OF GENETIC ALGORITHMS IN GLOBAL OPTIMIZATION

- Evolutionary algorithms are **population-based** optimization algorithms, inspired by the principles of nature revolution.

- The **advantage** of evolutionary algorithms is that they do not make limiting assumptions about the underlying objective functions.

  - The objective functions are treated as black-box functions.
  - The definition of objective functions does not require significant insight into the structure of the design space.

ROLE OF GENETIC ALGORITHMS IN GLOBAL OPTIMIZATION

- Evolutionary algorithms are based on the principles of biological evolution described in **Darwin's theory**.

- His theory was summarized as follows:

1. **Variation**: There is variation between individuals in a population.

2. **Competition**: Resources are limited. In such an environment, there will be a struggle for survival among individuals.

3. **Offspring**: Species have great fertility. They make more offspring than can grow to adulthood.

4. **Genetics**: Organisms pass genetic traits to their offspring.

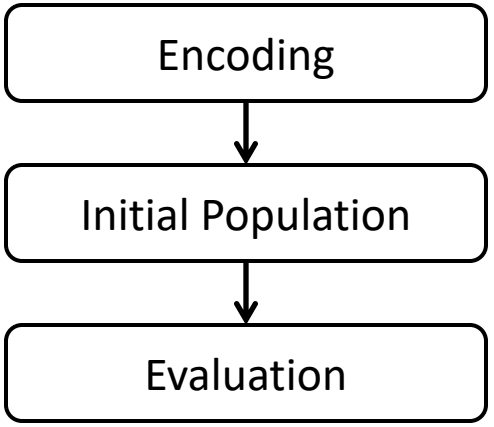
5. **Natural selection**: Those individuals with the most beneficial traits are more likely to survive and reproduce.

ROLE OF GENETIC ALGORITHMS IN GLOBAL OPTIMIZATION

- Genetic Algorithms (GAs) are **population-based metaheuristic** optimization algorithms.
- The GA was first conceived by **J.H. Holland** in 1975.
- The GA uses a population of solutions, whose **individuals** are represented in forms of chromosome. The individuals in the population go through a process of **simulated evolution** to obtain the global optimum.
- The GA repeatedly modifies a set of solutions or individuals in the course of its entire run. At each iteration, the genetic algorithm selects individuals from the current population to be **parents** based on certain criteria.
- The GA uses parents to produce the next generation of individuals, called **children**. Over successive generations, the population evolves toward an optimal solution

ROLE OF GENETIC ALGORITHMS IN GLOBAL OPTIMIZATION

The procedure of the genetic algorithm



***Encoding**:* Encoding is a way to represent individuals in evolutionary algorithms. Typically, individuals are coded as a finite fixed length **string**. This string is also known in the literature as a chromosome.

***Initial population*:** The algorithm begins by generating individuals in the design space. Prior to population initialization, the designer must choose the number of individuals in each population and the number of bits in the encoding.

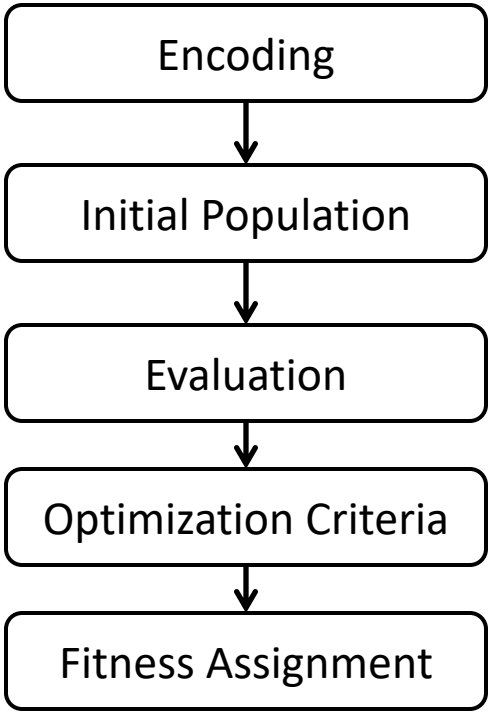
Evaluation: Computation of the objective values for the individual solutions.

ROLE OF GENETIC ALGORITHMS IN GLOBAL OPTIMIZATION

The procedure of the genetic algorithm

Optimization criteria: The stopping criteria of the algorithm. Examples of stopping criteria include number of generations, time limit, and function tolerance.

Fitness assignment: There are several choices of fitness assignment. In rank-based fitness assignment, the individuals are sorted according to their objective values. It creates an order among the individuals.



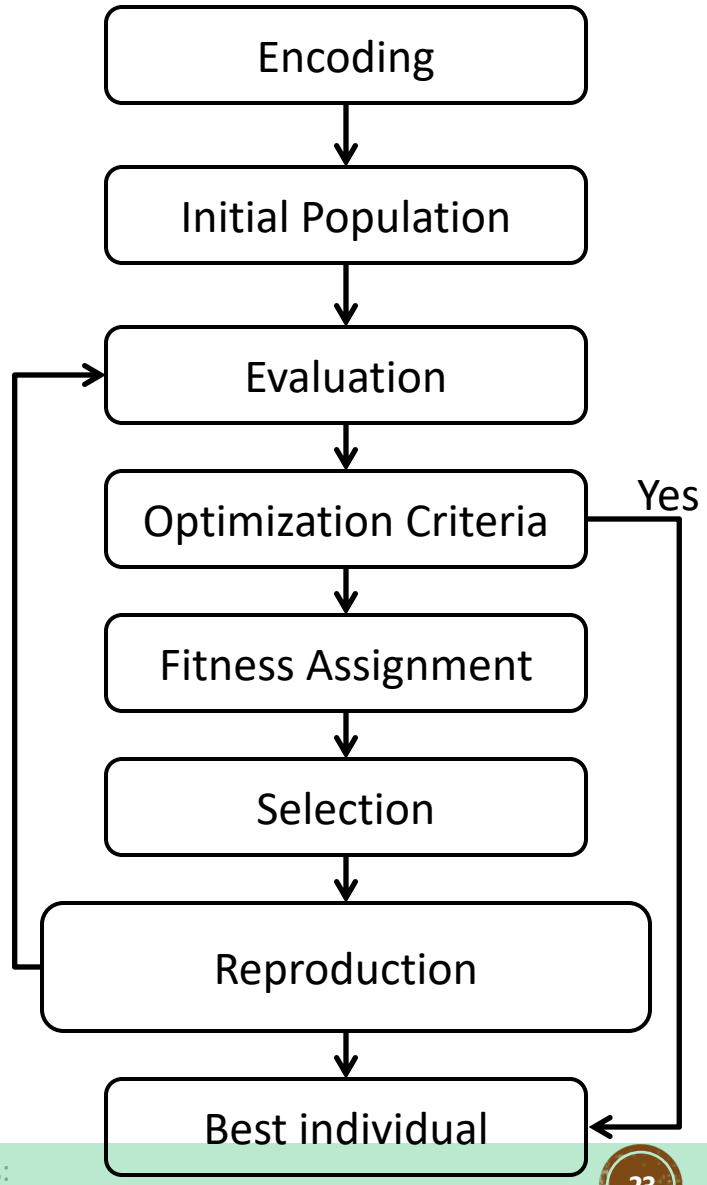
ROLE OF GENETIC ALGORITHMS IN GLOBAL OPTIMIZATION

The procedure of the genetic algorithm

Selection: A selection criteria filters out the candidate solutions with bad fitness and retains those with acceptable fitness to enter the reproduction process with a higher probability.

Reproduction: A new generation in the genetic algorithm is created by reproduction from the previous generation. Three mechanisms (elitist, crossover, and mutation) are mainly used to create a new generation.

Best Individual: The global optimum satisfying the optimization criteria.

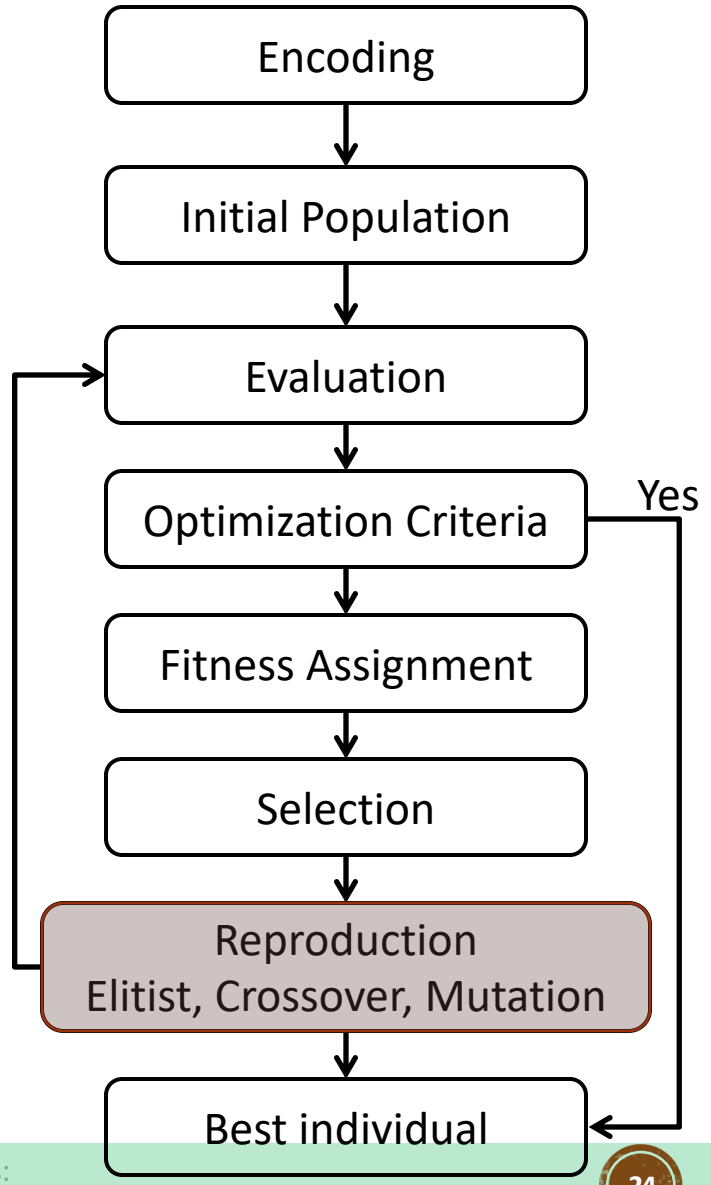


ROLE OF GENETIC ALGORITHMS IN GLOBAL OPTIMIZATION

The procedure of the genetic algorithm

Reproduction:

- **Elitist:** The individuals with the best fitness values in the current generation are guaranteed to survive in the next generation.
- **Crossover**: In this technique, a part of the encoded string of one individual is exchanged with the corresponding string part of another individual.
- **Mutation**: Mutated child solution is generated from a single parent by randomly reversing some bits from 0 to 1, or vice versa.



The procedure of the genetic algorithm

Reproduction:

Example for Crossover :

There are two individuals, 10101 and 11001. Exchange the first two bits of the two individuals. The offspring of the two individuals are 11101 and 10001.

Example for Mutation:

There is one individual, 10101. Through mutation, the 2nd bit and the 5th bit of 10101 are reversed. The new offspring is 11100.

GENETIC ALGORITHMS

- A GA implementation involves the following tasks:

1. Encoding:

- Encoding is a method that represent individuals in evolutionary algorithms.
- Typically, individuals are coded as a fixed length string, e.g., a binary number (0 or 1).
- This string is also known as a chromosome.

Example:

Use a string length of 5 to code a number in binary, e.g., 10001. This string can be decoded into a base 10 decimal number as shown below.

$$1\times2^4+0\times2^3+0\times2^2+0\times2^1+1\times2^0 = 16+1 = 17$$

- A GA implementation:

2. Initial population:

- The algorithm begins by generating a random initial population.
- Important choices such as number of individuals in each population, number of bits in the encoding must be made.
- These choices govern the performance of the GA.

- A GA implementation:

Example:

- Assume six individuals in the population for the present example.
- Each individual is randomly generated by a series of five fair coin flips, heads=1 and tails=0.
- Note that five coin flips are needed because we chose to encode each individual using a 5-bit binary string.

- A GA implementation:
  - The initial population thus generated is 10101, 11001, 01001, 11101, 10111, and 10000.
  - In the decimal system, the initial population is the following set of numbers {21, 25, 9, 29, 23, 16} .
  - Now evaluate the fitness function value ($f(x) = x^2$) at each of the individuals in the initial population. The fitness function value set is {441, 625, 81, 841, 529, 256}

Initial Population →

	Initial random population	decimal equivalent	function value
1	1 0 1 0 1	21	441
2	1 1 0 0 1	25	625
3	0 1 0 0 1	**9**	**81**
4	1 1 1 0 1	29	841
5	1 0 1 1 1	23	529
6	1 0 0 0 0	16	256

- A GA implementation:

3. Reproduction:

- A new generation, called child, in the genetic algorithm is created by reproduction from the previous generation, called the parent.
- The notion of "survival of the fittest" is usually used in genetic algorithms.
- There are three main mechanisms used to create a new generation.
  - Elitist
  - Crossover
  - Mutation

- A GA implementation:

3. Reproduction:

Elitist :

In this approach, the individuals with the best fitness values in the current generation are guaranteed to survive in the next generation.

Example:

- ✓Out of the following six individuals in the parent population:
- ✓{10101, 11001, **01001**,11101,10111}
- ✓The third individual,01001, had the lowest function value for the current minimization problem.
- ✓Such an individual is considered elite, and will become part of the next generation.

	Initial random population	decimal equivalent	function value
1	1 0 1 0 1	21	441
2	1 1 0 0 1	25	625
3	0 1 0 0 1	**9**	**81**
4	1 1 1 0 1	29	841
5	1 0 1 1 1	23	529
6	1 0 0 0 0	16	256

→ **Elite**

- A GA implementation:

3. Reproduction:

Crossover:

In this technique, some bits of the encoded string of one parent individual are exchanged with the corresponding bits of another parent individual.

Example:

- ✓ Assume that the elite individual, 01001, is part of the next generation.
- ✓ **First**, choose which individual is crossed over with which individual, e.g.,
  - individual 1 with individual 2,
  - or individual 1 with individual 3,
  - or individual 1 with individual 4.
  - This choice is usually made randomly;
  - Assume that individual 1, 10101, is crossed over with individual 2, 11001; and individual 4, 11101, is crossed over with individual 5, 10111.

- A GA implementation:

3. Reproduction:

Crossover:

The final choice to be made is the positions of the bits: exchange the first three bits, or the last three bits, etc. For this example, the first three bits will be exchanged for individuals 1 and 2, and bit 2 for individuals 4 and 5 .

Reproduction

Initial random population (parent)	Reproduction option (random choice)	New population (child)	decimal equivalent	function value
1 0 1 0 1	*crossover with 2 -- bits 1, 2, 3*	1 1 0 0 1	25	625
1 1 0 0 1	*crossover with 1 -- bits 1, 2, 3*	1 0 1 0 1	21	441
0 1 0 0 1	**elite**	0 1 0 0 1	9	81
1 1 1 0 1	*crossover with 5 -- bit 2*	1 0 1 0 1	21	441
1 0 1 1 1	*crossover with 4 -- bit 2*	1 1 1 1 1	31	961
1 0 0 0 0	*Mutation (bits 1, 3)*	0 0 1 0 0	**4**	**16**

- A GA implementation:

3. Reproduction:

Mutation:

- Unlike crossovers (which require two parents), mutation children are generated from a single parent by randomly reversing some bits from 0 to 1, or vice versa.
- In most GA implementations, a probability value for a mutation to occur is assumed.

Example:

Make the random choice that individual 6 goes through a mutation on bits 1 and 3.

The bits are reversed for these two positions, leading to a new child.

Reproduction

Initial random population (parent)	Reproduction option (random choice)	New population (child)	decimal equivalent	function value
1 0 1 0 1	*crossover with 2 -- bits 1, 2, 3*	1 1 0 0 1	25	625
1 1 0 0 1	*crossover with 1 -- bits 1, 2, 3*	1 0 1 0 1	21	441
0 1 0 0 1	**elite**	0 1 0 0 1	9	81
1 1 1 0 1	*crossover with 5 -- bit 2*	1 0 1 0 1	21	441
1 0 1 1 1	*crossover with 4 -- bit 2*	1 1 1 1 1	31	961
1 0 0 0 0	*Mutation (bits 1, 3)*	0 0 1 0 0	**4**	**16**

GENETIC ALGORITHMS

- A GA implementation:

4. The function values of the new population thus generated are computed.

- Using a combination of the above reproduction options, the algorithm proceeds further, until a desired stopping criterion is achieved.
- Examples of stopping criteria include number of generations, time limit, and function tolerance.

Example:
The best individual in the child generation shows a decrease in the function value when compared to the parent generation.

Initial random population (parent)	Reproduction option (random choice)	New population (child)	decimal equivalent	function value
1 0 1 0 1	*crossover with 2 -- bits 1, 2, 3*	1 1 0 0 1	25	625
1 1 0 0 1	*crossover with 1 -- bits 1, 2, 3*	1 0 1 0 1	21	441
0 1 0 0 1	**elite**	0 1 0 0 1	9	81
1 1 1 0 1	*crossover with 5 -- bit 2*	1 0 1 0 1	21	441
1 0 1 1 1	*crossover with 4 -- bit 2*	1 1 1 1 1	31	961
1 0 0 0 0	*Mutation (bits 1, 3)*	0 0 1 0 0	**4**	**16**

Example

$$\min \quad f(x) = x + 10sin(2x)$$

Subject to

$$0 \leq x \leq 10$$

- To set up this problem in MATLAB, using the ***ga* command**, three m-files are generated: a main file, an objective function file, and a constraint function file.
- The main file contains the initializations, bounds, options, and the ga command.
- The objective function file contains the objective or the **fitness function** definition.
- The constraint file contains the nonlinear inequality and equality constraints.

ROLE OF GENETIC ALGORITHMS IN GLOBAL OPTIMIZATION

Main Function

```
clear
clc
% Define Linear constraints
A=[ ]; B=[ ]; Aeq=[]; Beq=[];
% Define bounds constraints
LB=[0]; UB=[10];
% Number of design variables
nvars = 1;
% Optimization function ga
[x,fval] = ga(@GA_Func,nvars,A,B,Aeq,Beq,LB,UB,@GA_cons);
display(x)
display(fval)
```

ROLE OF GENETIC ALGORITHMS IN GLOBAL OPTIMIZATION

Objective Function

```
function [f]=GA_Func(x)
f = x+10*sin(2*x);
```

Constraint Function

```
function [C Ceq]=GA_cons(x)
% Define inequality constraints
C = [];
% Define equality constraints
Ceq = [];
```

The **global optimum of the objective function** obtained by MATLAB is **–7.6563**. The **optimum value of the variable is 2.3312**.

This problem can also be solved using the ga command from the graphical user interface of the **GA toolbox**.

Note that the folder that contains the fitness function and the nonlinear constraint function should be the MATLAB Current Working Directory.

ROLE OF GENETIC ALGORITHMS IN GLOBAL OPTIMIZATION

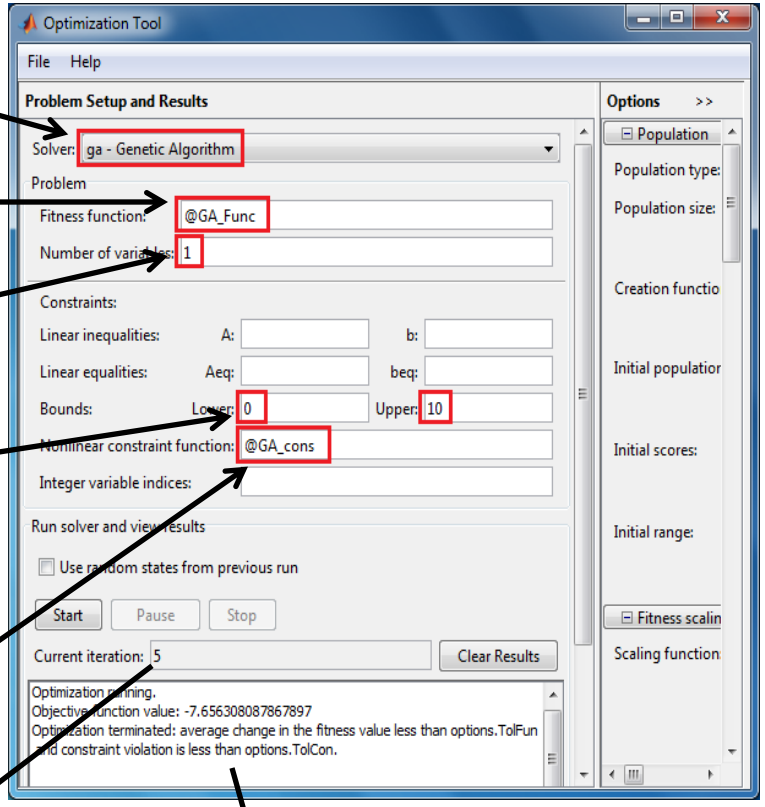
1. Select ga as the solver.

2. Set @GA_Func as the fitness function.

3. Set the number of variables as 1.

4. Set the lower bound as 0 and the upper bound as 10.

5. Set @GA_cons as the nonlinear constraint function(There is no nonlinear constraint for this problem, we can leave it blank).



After 5 iterations, the optimization is terminated.
The global optimum value of the objective function is −7.6563.

MATLAB GLOBAL OPTIMIZATION TOOLBOX

- MATLAB global optimization toolbox provides methods to search for global solutions for problems that contain multiple maxima or minima.
- The optimization solvers in the toolbox include global search, multiple start, pattern search, genetic algorithms, and simulated annealing solvers.
- For **genetic algorithms**, initial population and fitness scaling options can be modified, and parent selection, crossover, and mutation functions can be defined by users.
- The **multiobjective genetic algorithm** solver can be used to solve multiobjective optimization problems by identifying Pareto frontiers. This solver can be used to solve either **smooth or no smooth** optimization problems, **with or without the bound constraints and the linear constraints**.

MATLAB GLOBAL OPTIMIZATION TOOLBOX

- MATLAB global optimization toolbox can solve global optimization problems using the different solver like Genetic Algorithm and Simulated Annealing solver. The following steps are required to solve a global optimization problem using the toolbox.

  1. Select a solver and define an optimization problem.
  2. Set up and inspect the optimization options.
  3. Run the optimization tool and visualize intermediate and final results.

- If the **Parallel computing toolbox** is available, it can be used in conjunction with the global optimization toolbox to reduce computational time using parallel processing.

MULTIOBJECTIVE OPTIMIZATION USING GENETIC ALGORITHMS

- In weight-based multiobjective optimization problems, the multiple objectives are aggregated into a single objective using a weight-based scheme, such as weighted sum, or weighted square sum.
- One of the most significant drawbacks of weight-based methods is the requirement to specify weights, which is usually not intuitively obvious.
- The motivation for using evolutionary algorithms to solve multiobjective problems is two-fold:
  1) EAs simultaneously work with a population of solutions, allowing for a series of Pareto solutions to be found in one converged run, as opposed to traditional techniques where the Pareto solutions are found sequentially one run at a time.
  2) EAs are not affected by the shape of the Pareto frontier (convex or concave) or discontinuous Pareto fronts.

- A drawback of the Vector Evaluated Genetic Algorithm (VEGA) approach is its bias towards some Pareto solutions.
- A so-called non-dominated sorting procedure was later implemented by several researchers to overcome the drawbacks of VEGA.
  - ✓A ranking procedure is adopted to rank individuals in a population.
  - ✓An individual, a, is said to dominate another individual, b, if a is strictly better than b in at least one objective, and a is no worse than b in all objectives.
  - ✓A distance measure is used to compare individuals with equal rank.

Example

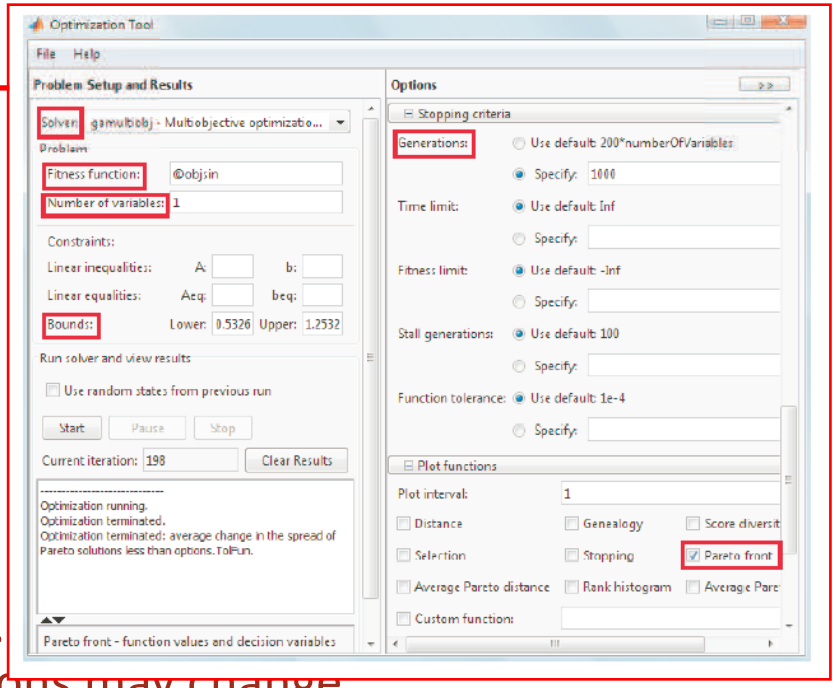
$$\min_x \ \{f_1 = \sin x, f_2 = 1 - \sin^7 x\}$$

$$\text{subject to}$$

$$0.5326 \leq x \leq 1.2532$$

Example

- The **gamultiobj** accepts only linear equality constraints, linear inequality constraints, and bounds on the design variables.
- This problem can be solve by using the graphical user interface (GUI) of the GA toolbox.



- ✓ This screen can be opened by clicking on Start on the lower left corner of the Matlab command window,
- ✓ Toolboxes → Genetic algorithm and direct search → Genetic algorithm tool

Note the selections highlighted by boxes. in the window, based on which the solutions may change.

Example

- Before you can use this tool, you need to create a file that contains the objective function **objsin.m** that defines the two objectives.

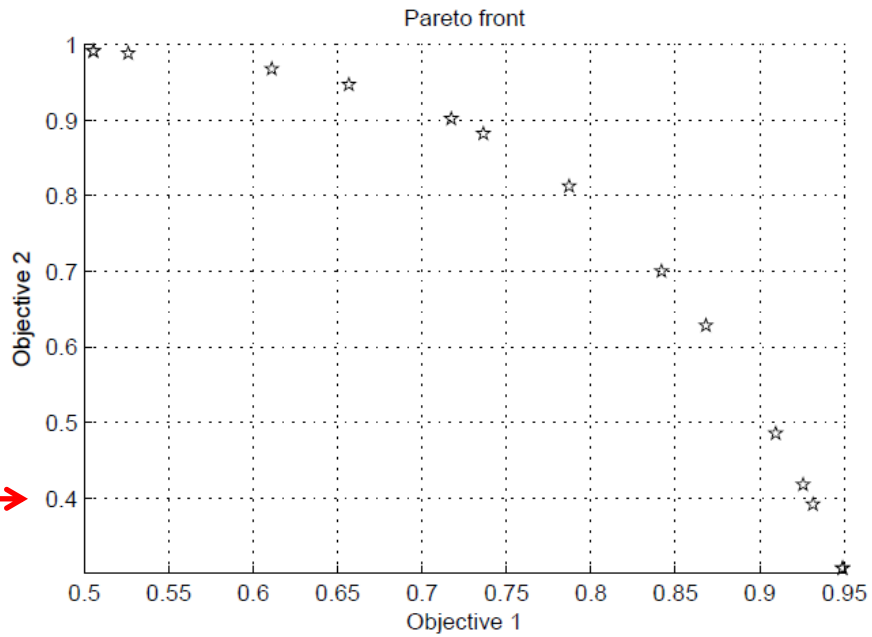
Objective Function File:

```
function f = objsin(x)
f(1) = sin(x);
f(2) = 1 - sin(x)^7;
```

- In the GUI for the GA toolbox, the user can provide the function handle for the fitness function.
- Keep in mind that the **objsin.m** file must be saved in the current directory in Matlab.

Example

Pareto frontier for this problem as generated by Matlab's multi-objective genetic algorithm tool.



- For the latest help on the additional features, refer to Matlab's help.
- By using Parallel Computing Toolbox which is available in the GUI for the GA toolbox, it is possible to set the option as '**in parallel**' for evaluating fitness and constraint functions.
- By using the built-in parallel computing capabilities or defining a custom parallel computing implementation of the optimization problem, it is possible to significantly decrease solution time.

PARTICLE SWARM OPTIMIZATION

- Particle Swarm Optimization
  - In this algorithm, an initial set of randomly generated individuals is used, with each being a candidate solution. These individuals are also known as **particles**, hence the name particle swarm.
  - An iterative process to improve these candidate solutions is started, or is set in motion.
  - The particles iteratively evaluate the fitness of the candidate solutions, and locations of best fitness are remembered or recorded by each individual.
  - An individual's best solution or success is called the particle best, and this information is shared with the neighbors.
  - A swarm is typically modeled by particles in multidimensional space that have a position and a velocity.
  - Movements through the search space are guided by these successes, with the population usually converging.

BASIC PSO DYNAMICS

Diversity Preserving Behavior

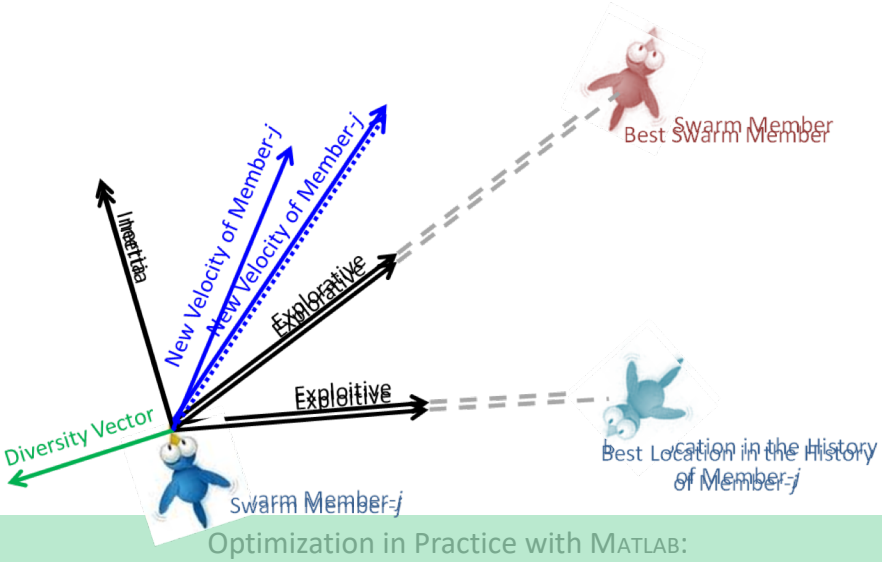
Location Update: $X_i^{t+1} = X_i^t + V_i^{t+1}$ and

Velocity Update: $V_i^{t+1} = \alpha V_i^t + \beta_l r_1 (P_i - X_i^t) + \beta_g r_2 (P_g - X_i^t)$

Inertia

Personal/Exploitive Behavior

Social/Explorative Behavior

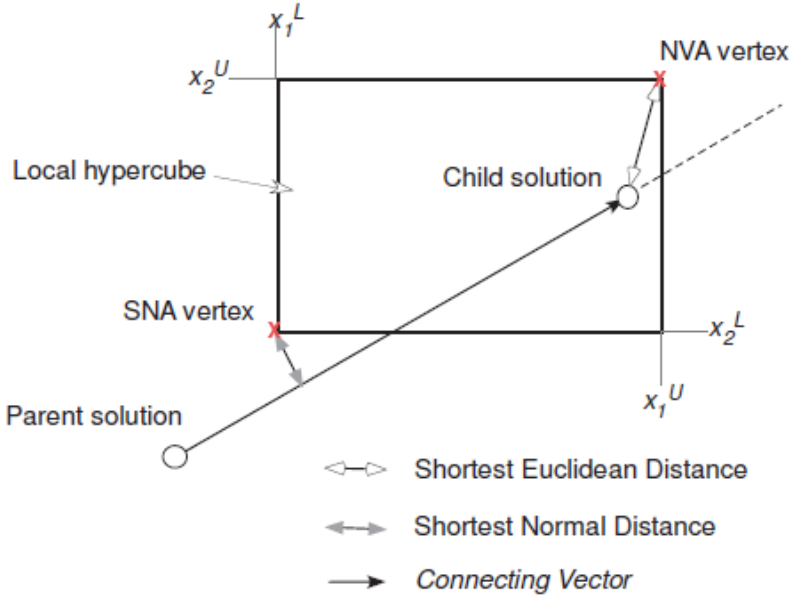


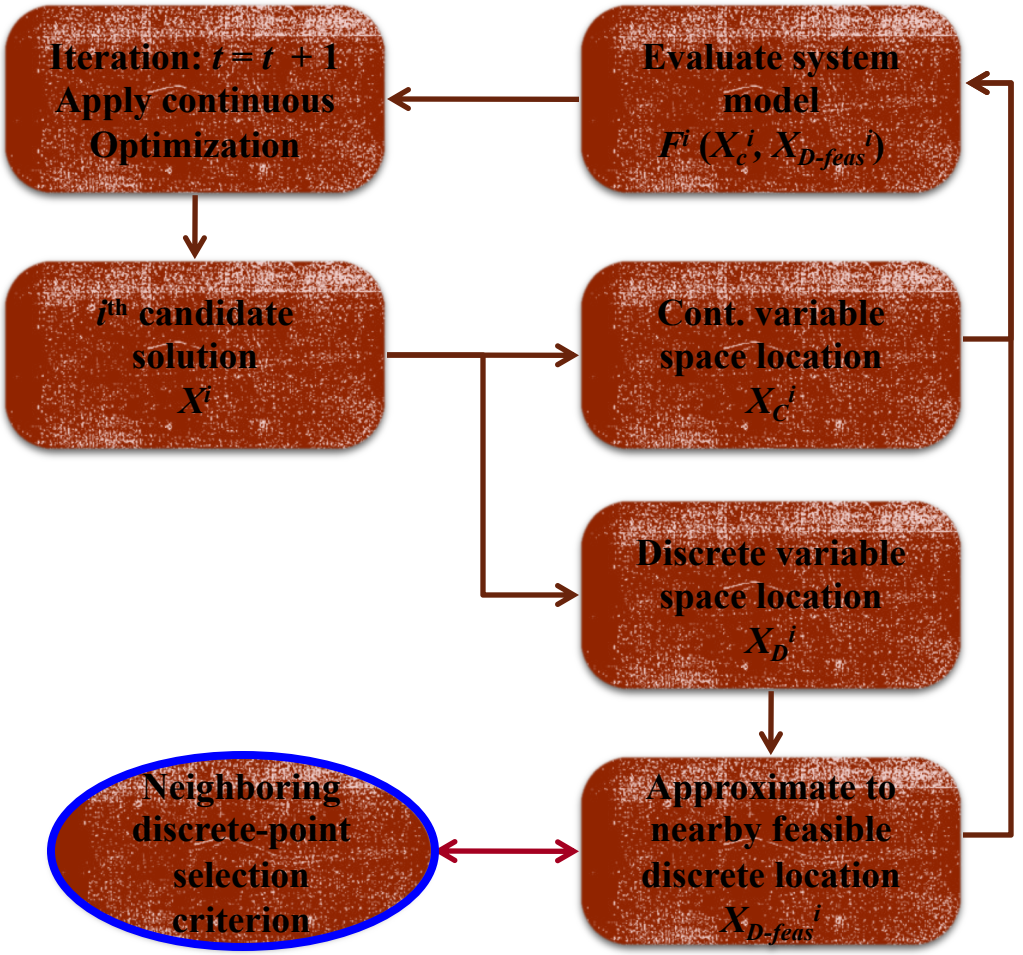
DISCRETE VARIABLES IN PSO

Enclosing Cell

$$H_d = \{(x_1^L, x_1^U),\ (x_2^L, x_2^U),\ \ldots,\ (x_m^L, x_m^U)\}$$
$$x_i^L \le x_i \le x_i^U, \quad \forall\ i = 1, 2, \ldots, m$$

Nearest Vertex Approach





The allowed values of each discrete variable is known a priori