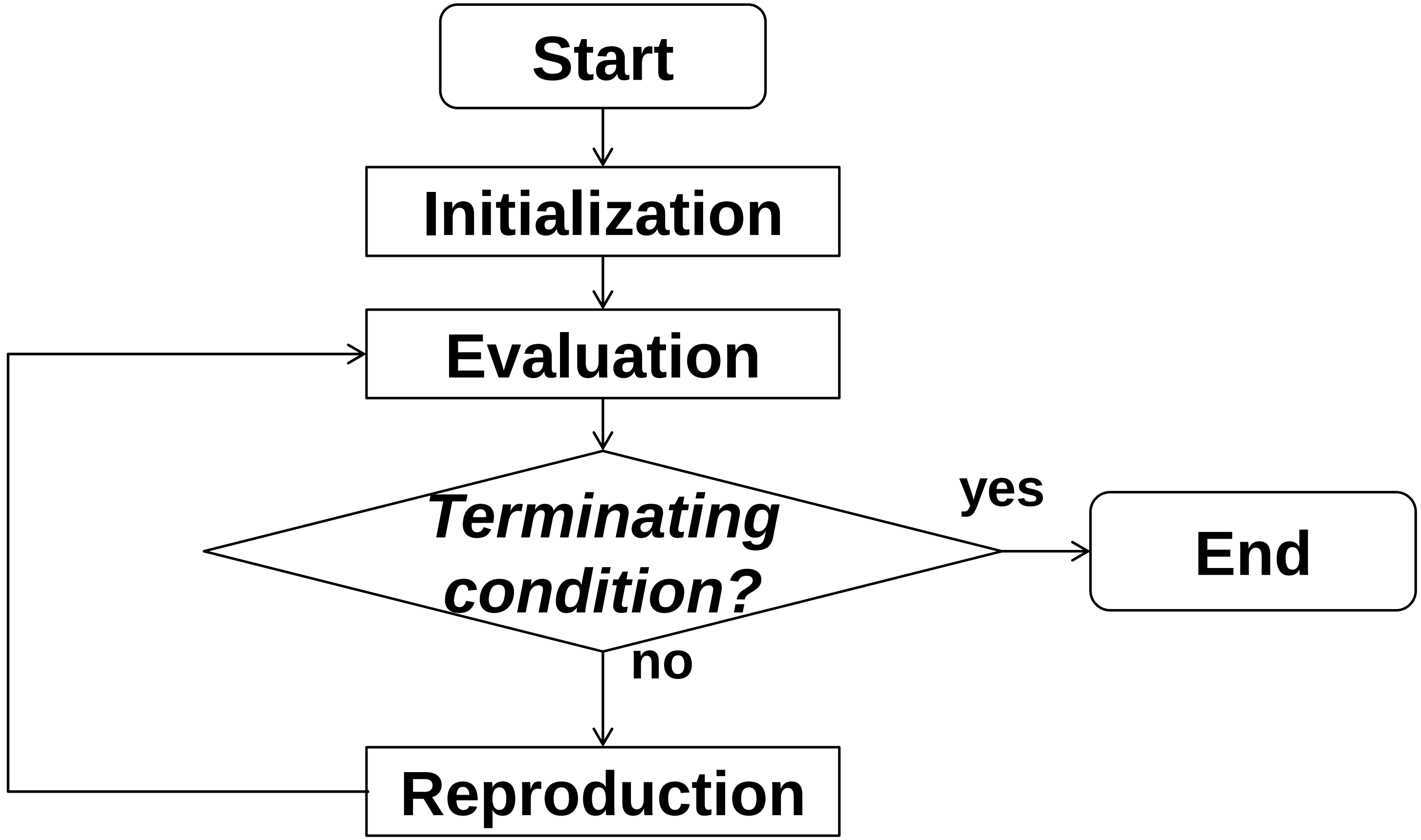
Population-based search

Topics of this lecture

- Genetic algorithm (GA)
  – Individual, population, and generation
  – Genotype, phenotype, and fitness
  – Selection, crossover, and mutation
- Particle swarm optimization (PSO)
  – Particle and swarm
  – Personal factor and social factor

Genetic Algorithm



Properties of GA

- GA is population-based
  - Many points are used for search. Each point corresponds to a potential solution. A solution is called an **individual**, and the set of all individuals is called the **population**. The best individual obtained after evolution is often used as the final result.
- GA is a generate-and-test algorithm
  - The function may not be continuous or derivative. The problem space may not be described by an "equation".
- GA is history preserving
  - New solutions are generated from the old ones via genetic operations, namely selection, crossover, and mutation.

Genotype, phenotype, and fitness

- In GA, a solution is often represented in a binary string.
- This binary string is called the **genotype,** and the solution itself is called **phenotype**.
- The genotype is the genetic code of the individual, and the phenotype is the body of the individual.
- The genotypes of all individuals are initialized at random.
- The phenotype can be built from the genotype, and the goodness or **fitness** of an individual is evaluated based on the phenotype.

Method for evaluation

- This method for evaluation is usually different for different problems.
- The point is that, even if we do not have much information about the problem space, we can use GA to get a good solution, provided that a proper "**method**" is given.
- This method usually tells how good an individual is based on the performance of the phenotype.

The terminating condition

- Although natural evolution is an endless process, GA must stop at a certain point to get a useful solution.
- Usually, we stop the evolution process if the best current solution is good enough or not. The best current solution is used as the final answer.
- Different terminating conditions can be used for solving different problems.
- We may just specify the maximum number of iterations for terminating the program. One iteration is called one **generation** in GA.

The genetic operations

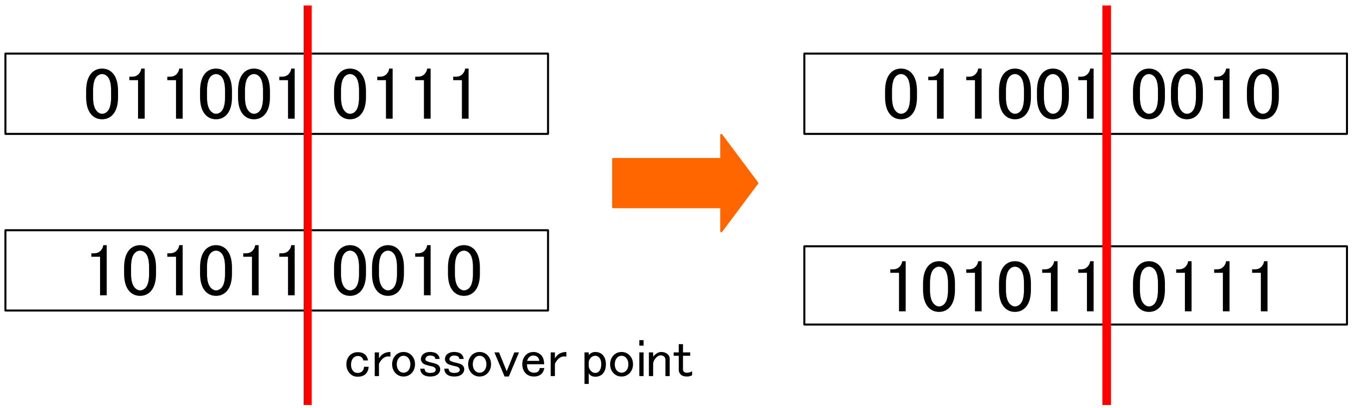
1. Selection
   – Survival strategy
2. Crossover:
   – Generating new solutions by recombination of two or more parents
   – For intensification or exploitation!
3. Mutation:
   – Generating a new solutions with one parent
   – For diversification or exploration!

Operator-1: truncation selection (cont.)

0.66	0.84	0.84	0.84
0.84	0.79	0.79	0.79
0.39	0.75	0.75	0.75
0.51	0.66	0.66	0.66
0.42	0.51		0.85
0.75	0.42		0.50
0.20	0.39		0.91
0.79	0.20		0.28

Operation-2: One-point crossover

- Choose one point at random
- This point is called the crossover point
- Cut each parent into two parts
- Recombine them to generate two children



Operation-3: Bit-by-bit mutation

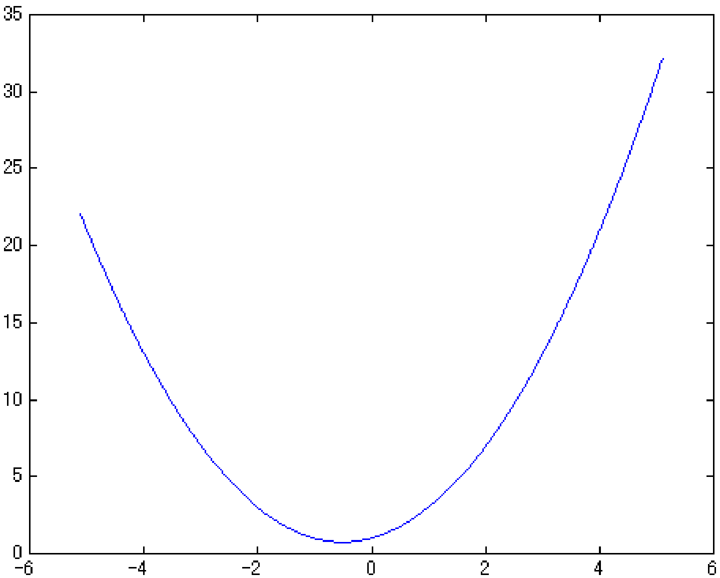
- For each bit of the binary code
- Generate a random number in [0,1]
- If this number is less than p_m, reverse the bit value (0→1 or 1→0)
- p_m is called the mutation rate

0011110011

0001110111

A simple example

- Problem: Find the maximum point of $f(x)=1+x+x^2$ from the domain: $-5.12 < x < 5.12$



Genotype, phenotype, and fitness function

- Genotype: 10 bits binary number (coding)
- Range of fixed point integer: y=[0,1023]
- Phenotype: x=(y-512)/100 (decoding)

- The evaluation method
  - Fitness = f(x)
  - Maximum=32.3344
- The fitness is found in two steps
  - Reconstruct the phenotype x
  - Substitute x into f(x)

[00110 11000] $\rightarrow$ $y = 216$ $\rightarrow$ $x = -2.96$ $\rightarrow$ *fitness* $= 6.8016$

Results

Before evolution:

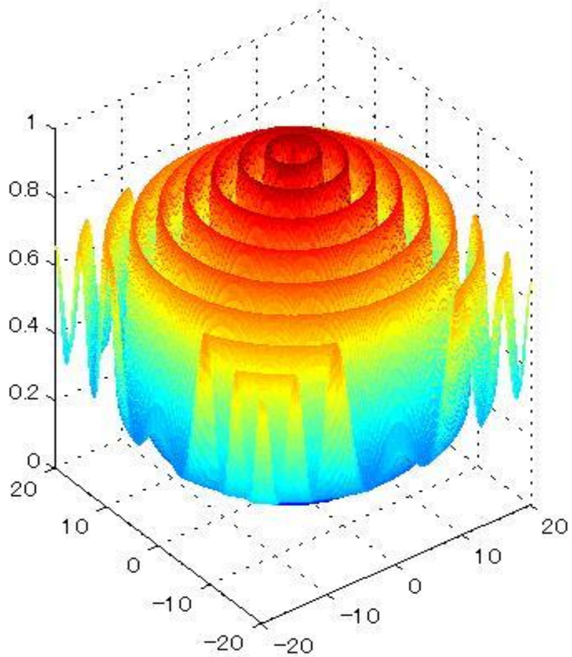
I[0]: 1001000111 3.930000
I[1]: 1111101011 3.510000
I[2]: 1100001011 3.230000
I[3]: 0010011100 -2.840000
I[4]: 0001010101 1.680000
I[5]: 0100111010 -1.420000
I[6]: 0001010001 0.400000
I[7]: 1000000001 0.010000
I[8]: 0100010110 -0.940000
I[9]: 1111011110 -0.170000
The maximum point is 3.930000
The maximum value is 20.37490

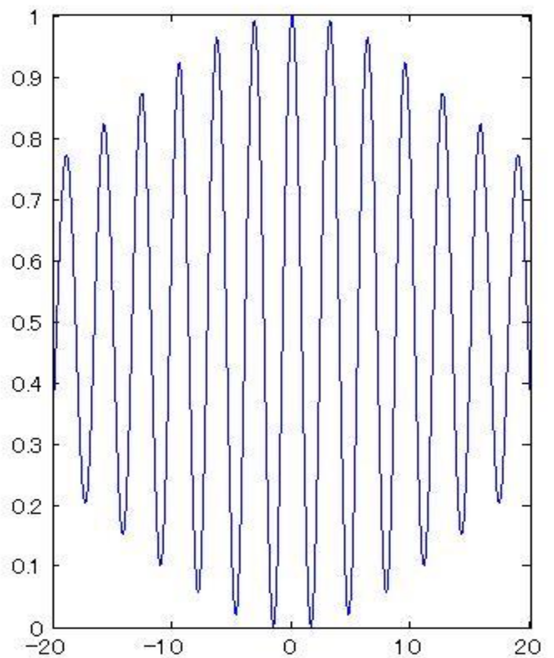
For the 9-th generation:

I[0]: 1000101111 4.650000
I[1]: 0110001111 4.540000
I[2]: 1010010111 4.210000
I[3]: 1010000000 -5.070000
I[4]: 1001000111 3.930000
I[5]: 1111100000 -4.810000
I[6]: 1110010000 -4.730000
I[7]: 1101101011 3.470000
I[8]: 1101111100 -2.610000
I[9]: 1110100001 0.230000
The maximum point is 4.650000
The maximum value is 27.272500

Example 8.4 pp. 180-181

$$f(\mathbf{x}) = 0.5 - \frac{(\sin\sqrt{x_1^2 + x_2^2})^2 - 0.5}{(1.0 + 0.001(x_1^2 + x_2^2))^2}$$



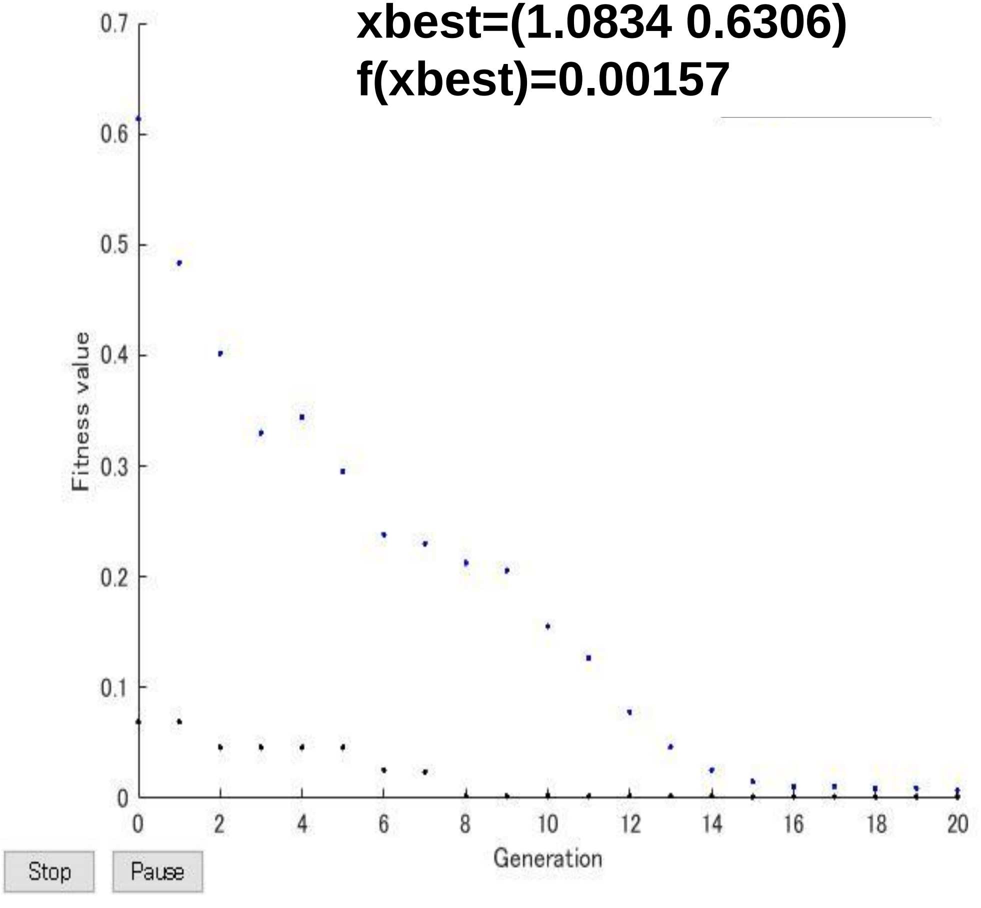


Example 8.4 pp. 180-181

```
1. F6=@(x)(0.5-((sin(x(1)^2+x(2)^2))^2-
   0.5)/(1.0+0.001*(x(1)^2+x(2)^2))^2);
2. options=gaoptimset('Generations',20,'PopulationSize',20,
   'PlotFcns',@gaplotbestf);
3. lb=[-20, -20];  % lower bound of (x1,x2)
4. up=[20, 20];   % upper bound of (x1,x2)
5. x=ga(F6,2,[],[],[],[],lb,up,[],options)
```

Example 8.4 pp. 180-181

xbest=(1.0834 0.6306)
f(xbest)=0.00157



Why GA works?

- The population used in GA can be considered the OPEN LIST used in graph-based search.
- That is, the population keeps a set of potential solutions for further investigation.
- Crossover and mutation generate new solutions, and the fitness-based selection provides a heuristic for conducting “best-first search”.
- The main difference is that, in GA we consider the whole population a state, and transit the population from one state to another during evolution.

Particle Swarm Optimization

- PSO is another population based search algorithm.
- Instead of using genetic operations, in PSO, each individual tries to learn directly by itself.
- Through learning (self-study), the individuals may become good quickly.
- If the environment for evolution is constant, PSO can be more efficient than GA.

Basic considerations

- In PSO, the population is called swarm, and each individual is called a particle.
- For each particle, we need to record
  - The current position;
  - The best position found so far; and
  - The velocity.
- For the whole swarm, we need to record the best particle (the global leader) found so far.

The algorithm

- Step 1: Randomly initialize the swarm.
- Step 2: Evaluate all particles.
- Step 3: For each particle
  - Update its velocity;
  - Update its position;
  - Evaluate the particle.
- Step 4: Update if necessary the leader of the swarm and the best position obtained by each particle.
- Step 5: Stop if terminating condition satisfied; return to Step 3 otherwise.

Update the velocity

- The velocity of a particle is updated as follows:

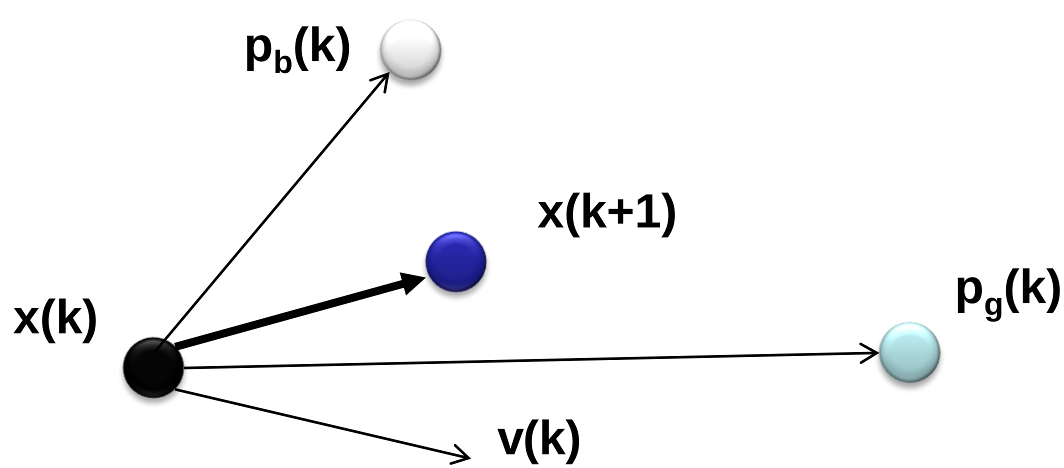
$$\mathbf{v}^{new} = a\mathbf{v}^{old} + bw_1 \times (\mathbf{x}_{my\_best} - \mathbf{x}^{old}) + cw_2 \times (\mathbf{x}_{best} - \mathbf{x}^{old})$$

where a is the inertia weight, b and c are the learning factors called personal factor and social factor, respectively, and w_1 and w_2 are random numbers taken from $[0,1]$.

Update the position

- Based on the new velocity, the new position is obtained as follows:

$$\mathbf{x}^{new} = \mathbf{x}^{old} + \mathbf{v}^{new}$$



Physical meaning of PSO

- Each particle tries to learn from the current leader as well as the best position found by itself.
- The amount of information (influence) obtained from the leader and the best position found so far depends on the learning factors *b* and *c*.
- Also depends on the random factors w_1 and w_2. Thus, we will obtain a different solution in a different run.

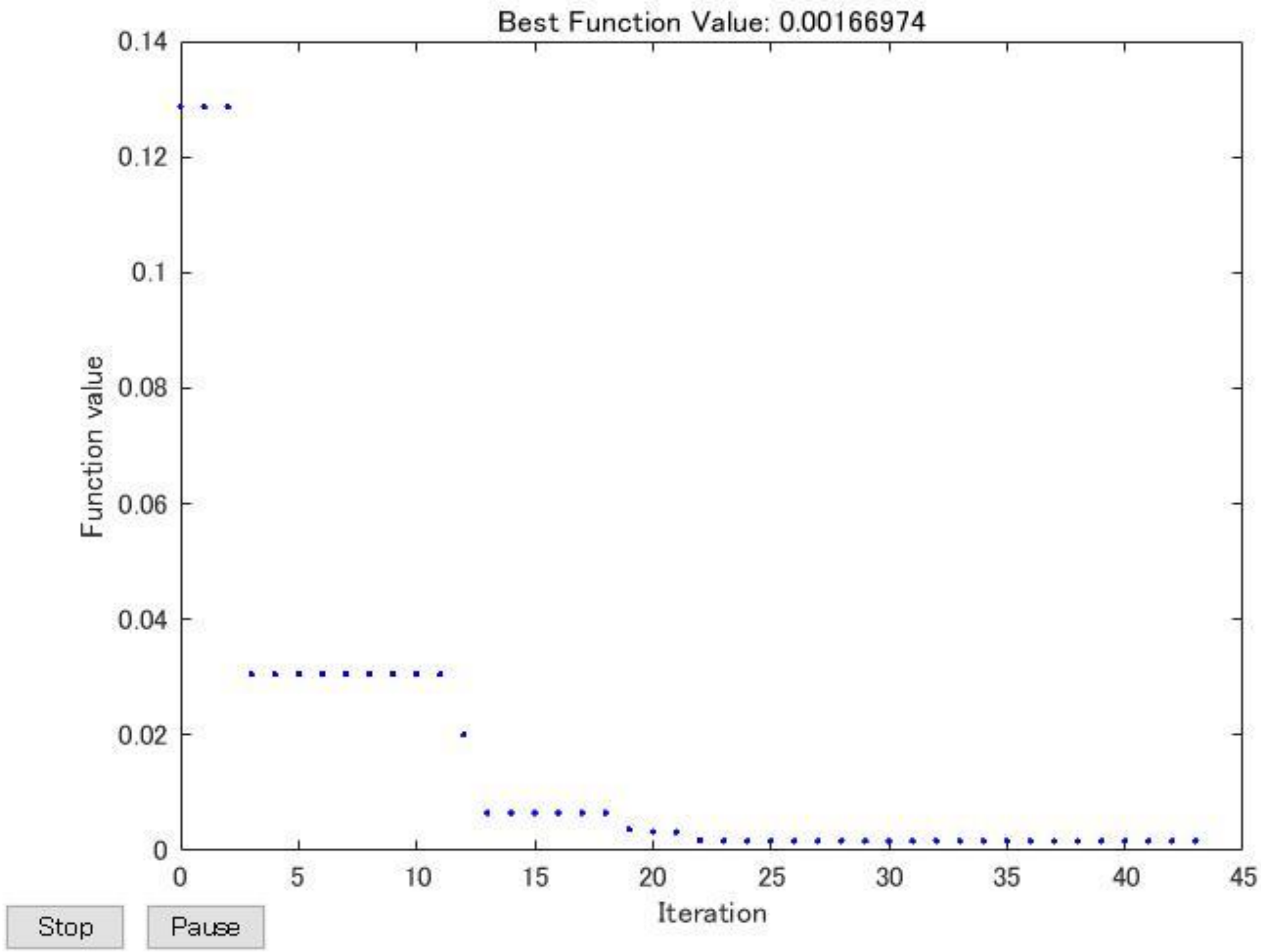
Parameters and Their Values

- n: Number of particles (population size).
- *a*: the inertia weight, usually increases linearly during learning from a0 (=0.4) to a1 (=0.9).
- *b*:Cognitive (personal) learning factor which controls the movement towards its own success (=2).
- *c*: Social learning factor which controls the movement towards the leader (=2).
- A neighborhood size will also be used to define ***local*** leaders for different regions.

Example 8.5 pp. 183-184

```
1. F6=@(x)(0.5-((sin(x(1)^2+x(2)^2))^2-
   0.5)/(1.0+0.001*(x(1)^2+x(2)^2))^2);
2. options =
   optimoptions(@particleswarm,'SwarmSize',20,'PlotFcns',
   @pswplotbestf);
3. lb = [-20,-20];
4. ub = [20,20];
5. x=particleswarm(F6,2,lb,ub,options)
```

Example 8.5 pp. 183-184



Homework for lecture 14 (1)

- Try to solve the problem given in Example 8.4 in the textbook, using the program given in Table 8.7.
- Plot the result, and submit to the TA during the exercise class

Homework for lecture 14 (2)

- Try to revise the program given in Table 8.7, and solve Problem 8.5 given in p. 181 of the textbook.
- Put the matlab program into “prog_15.m” and the results into “result_15.txt”.
- Plot the result, and write your observations in “summary_15.txt”.

Quizzes for lecture 14

- To use genetic algorithm we usually encode a solution or individual into a binary string. This string is called genotype of the solution. To evaluate the goodness of the solution, we should decode the genotype to__________________. Only genotype evolves during evolution.
- The goodness of a solution is called the _____________________. We need a method to evaluate the_____________________of a given solution. This method may not be given in a closed form formula.
- There are mainly three genetic operations in GA, namely, selection, crossover, and _________ . Together they produce new candidate solutions for further evolution. ___________________ is important for preserving the diversity of the population.
- In PSO, each candidate solution is called a___________________. We need to keep the current position and velocity of a ______________ in the search process.
- In PSO, each particle learns by itself. There are main two factors for learning. One is the personal factor, and another is ______________ factor. The latter is important for “information sharing”.