Genetic Algorithm

Feng-Cheng Yang 楊烽正

computer automation

Enthusiasm • Leadership • Innovation • Teamwork • Excellence

NTU 國立臺灣大學工業工程學研究所

Genetic Algorithm

- Created by Holland (1975)
- Optimization problems from the industries and engineering are very complex in nature
  - Hard to solve by conventional optimization techniques:
    - linear programming, mathematic programming, …
- In 1960s, a trend of imitating living beings to solve hard optimization problems
  - Simulate the natural evolutionary process of human beings
    - Stochastic optimization techniques
    - Evolutionary algorithms
  - Three main categories of evolutionary algorithms
    - GA, genetic algorithms
    - EP, evolutionary programming
    - ES, evolution strategies (solution search)

- Well known applications in area of industrial engineering
  - Scheduling and sequencing
  - Reliability design
  - Vehicle routing and scheduling
  - Group technology
  - Facility layout and location
  - Transportation
  - Machine Learning

General Structure of GA

- Stochastic search techniques based on the mechanism of natural selection and natural genetics
- Start with an initial set of random solutions
  - Population (a set of solution agents)
  - For simple problems
    - Each individual in the population is called a chromosome
    - Representing a solution to the problem
  - For complex problems
    - A set of chromosomes represents a solution
      - Called an organism
    - Each organism represents a solution to the complex problems

General Structure of GA

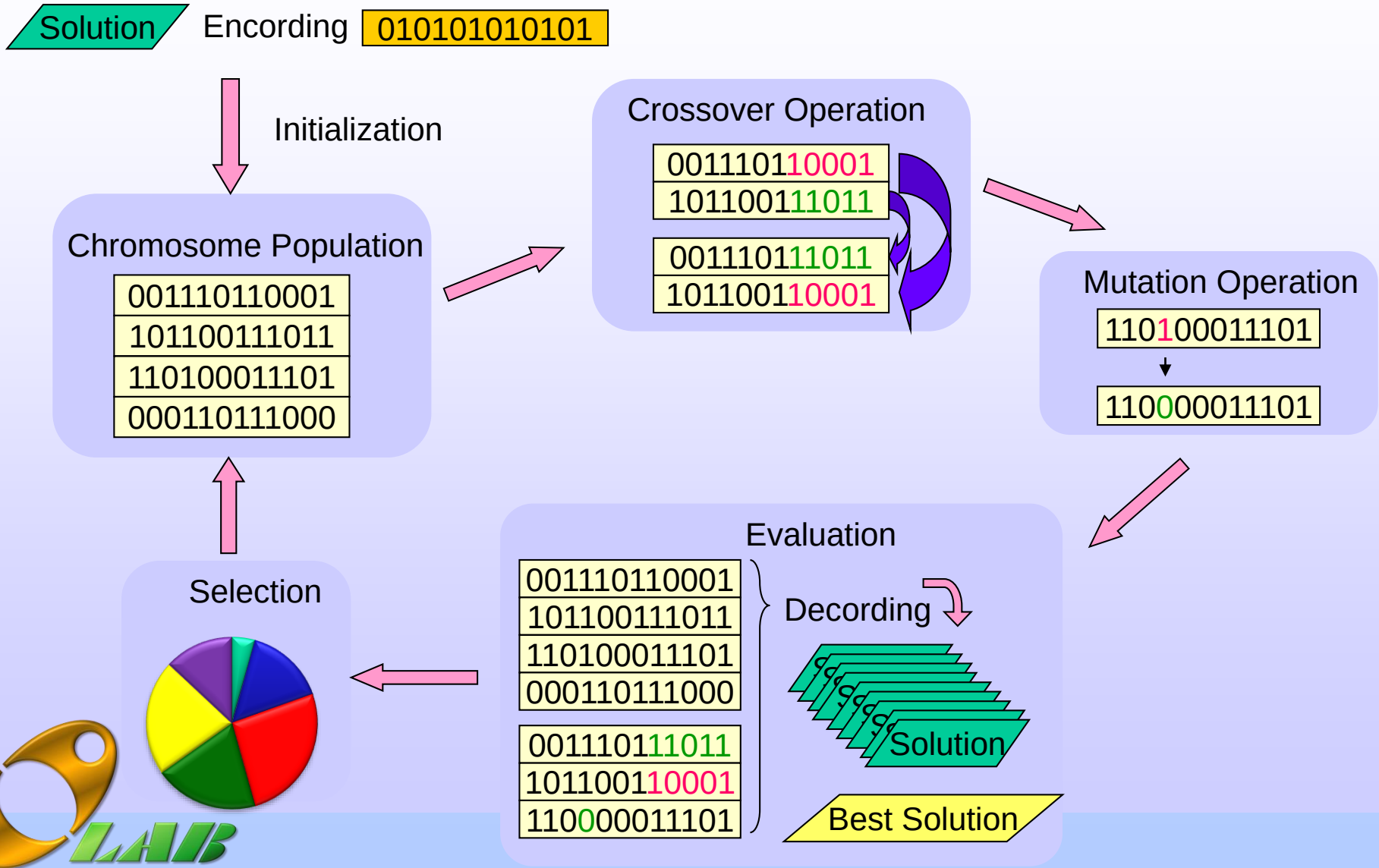
- Chromosome
  - A string of symbols
    - Typically is a binary bit string
  - Evolve through successive iterations, called generation
  - Evaluated by a measure of fitness of the single chromosome or the set of chromosomes (the organism)

- New chromosomes are called offspring
  - Formed by merging two chromosomes
    - By using a crossover operator
  - Formed by modifying a chromosome
    - By using a mutation operator
- New generation (population)
  - Formed by selecting some of the parents and offspring and rejecting others
    - Fitter chromosomes (high fitness) have higher probabilities of being selected
- The algorithm converges to the best chromosome
  - The optimum or suboptimal solution to the problem

Genetic Algorithm

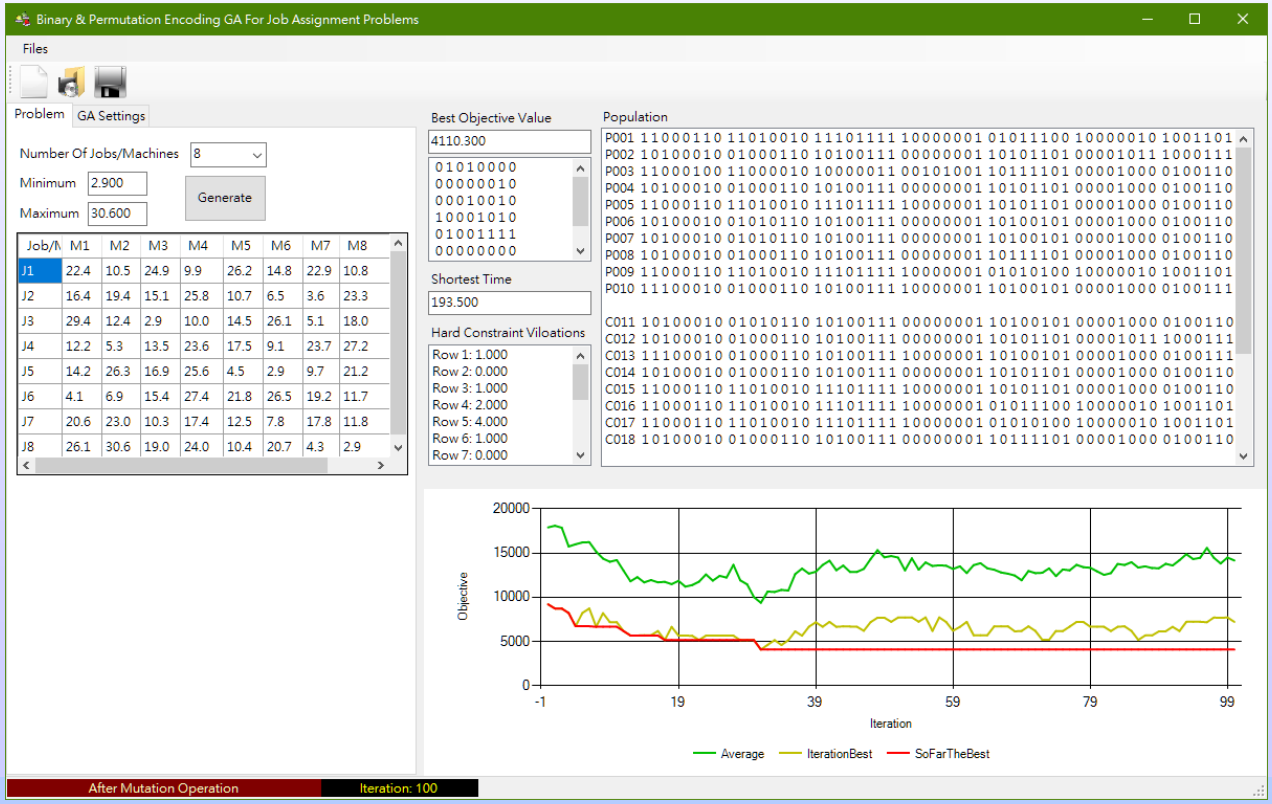
- $P(t)$: **parents in current generation** t
- $C(t)$: **offspring in current generation** t

```
Begin
        t←0;
        initialize P(t)
        evaluate P(t);
        while( not termination condition ) do
                recombine P(t) to yield C(t)
                evaluate C(t);
                select P(t+1) from P(t) and C(t);
                t ← t+1;
        end
end
```


Solution
Encording
010101010101
Initialization
Chromosome Population
001110110001
101100111011
110100011101
000110111000
Crossover Operation
001110110001
101100111011
001110111011
101100110001
Mutation Operation
110100011101
110000011101
Evaluation
001110110001
101100111011
110100011101
000110111000
001110111011
101100110001
110000011101
Decording
Solution
Best Solution
Selection

GA Solver Preview

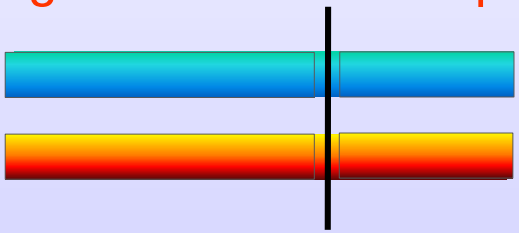
Job assignment optimization problem: N jobs to be assigned to N machines
A job can only be assigned to one machine; A machine can only process one job
A linear programming problem → should be solved by hard computing method
0-1 variables are used subject to the above assignment requirements
X_{ij}= 1, job i is assigned to machine j



- **Initialization**
  - **Usually, random initialization**
- **Recombination**
  - **Usually, via crossover and mutation operations**
- **Genetic operations**
  - **(1) Crossover and Mutation**
  - **(2) Selection**
    - Mimic the process of heredity of genes
    - Mimic the Darwinian evolution

Crossover operation

- **Main genetic operation**
  - **Operate on two chromosomes to generate offspring by combining parent's features**
  - **Simple way**
    - Choose a random cut-point and generate offspring by combining the left segment of one parent with the right segment of the other parent



  - **Determine the performance of GA**
- **Crossover rate, P_c**
  - **Allow only P_c x pop_size chromosomes to perform crossover operation**

Mutation Operation

- Produce random changes in various chromosomes
- Simple way
  - Alter one or more genes
- Mutation rate, P_m
  - The percentage of the total number of genes to be operated by the mutation
  - Control the rate at which new genes are introduced into the population

Solution Search

- Problems do not have priori information about the sequence of steps leading to a solution
  - Search is required
- Search
  - Blind strategy
    - Use no information about the problem
  - Heuristic strategy
    - Use additional information to guide the search along with the best search directions
- Two important issues in solution search
  - Exploitation (探究)
    - For the nearest best solution
  - Exploration (探索)
    - For the possible best solution

Solution Searches

- Hill-climbing search (Steepest descent search)
  - Exploit the best solution for possible improvement, ignoring the exploration of the search space
- Random search
  - Explore the search space ignoring the exploitation of the promising regions

GA vs. Conventional Optimization and Search Procedures

- Work with a coding of solution set
  - Not the solutions themselves
- Solution search from a population of solutions
  - Not a single solution
- Uses payoff information (fitness function)
  - Not derivatives or auxiliary knowledge
- Use the probabilistic transition rules
  - Not deterministic rules

Solution Searches

- **Genetic Search**
  - **A class of general-purpose search method**
  - **Combine directed and stochastic search**
  - **Initialization**
    - Explore all solution space
  - **Crossover operation**
    - Explore the neighborhood of each chromosome
    - One can not always use simple operations for all kinds of Problem
      - Use information of the problem domain to design the operations
      - Otherwise, it is a blind search

- **Traditional Optimization Search (hard computing)**
  - **Point-to-point**
  - **Generate a deterministic sequence of computation based on the gradient or higher-order derivatives of objective function**
  - **The point is improved along the deepest descending or ascending direction**
  - **Risk of falling in local optima**

Genetic search

- **Population-to-population**
  - Not point-to-point (single-to-single)
- **Multiple directional search by a population of potential solutions**
- **Escape from local optima**
- **The population undergoes a simulated evolution**
  - The relatively good solutions are reproduced
  - The relatively bad solutions die
  - Use probabilistic transition rules

GA Optimization Method

- **Previous Approach (Classic Approach)**
  - **Used as a generic tool for many difficult-to-solve problem**
    - Chromosomes: Fixed-length binary strings
    - Simple binary crossover and mutation operators
    - Domain-independent
  - **Emphasis on the robust adaptive systems**
  - **Difficult to apply directly and successfully into difficult-to-solve problem**

GA Meta-Heuristic

- **Current Approach**
  - **GA is used as a meta-heuristics**
    - Nonstandard implementations are created for particular problem
  - **Various data structure for nature representation**
  - **Various sets of genetic operators applied to these data structures**

Advantages of GA

- Have little mathematical requirements about the optimization problem
  - Can handle any kind of objective functions and constraints (linear or nonlinear)
  - Discrete, continuous, or mixed solution space
- Genetic operators are very effective in performing global search
  - Traditional one (hard computing) performs local search by a stepwise procedure
- Flexibility to hybridize with domain-dependent heuristics to make an efficient implementation for a specific problem

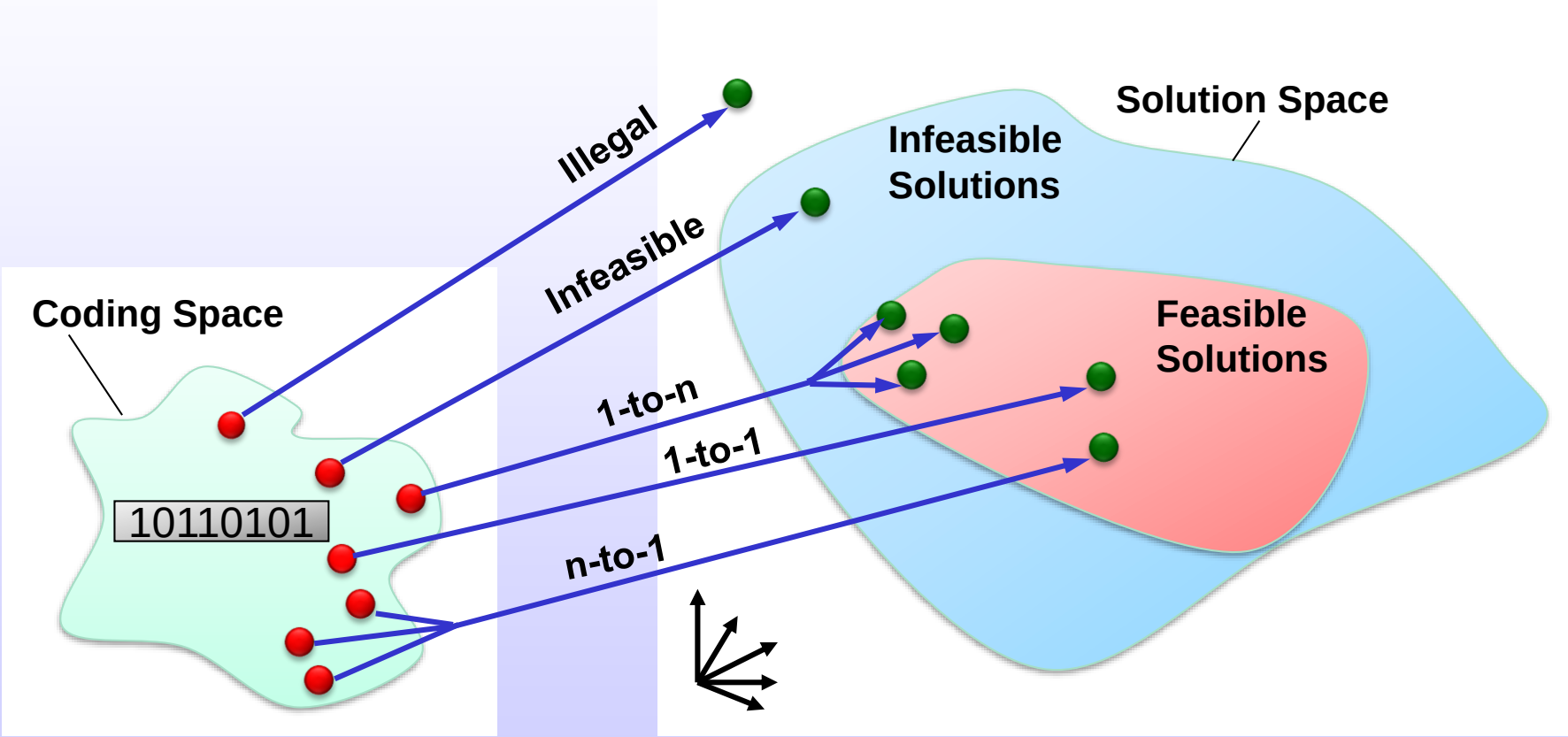
Vocabulary of GA

- Biologic organism
  - One or more chromosomes specify the complete organism
  - Genotype (基因型)→ the complete set of chromosomes
  - Phenotype(顯形) →the organism
- Chromosome
  - Comprises a number of genes
- Gene
  - Represents a particular feature of human being
  - Encodes a particular feature of the organism and the location

Encoding Problem

- **Coding Space vs. Solution Space**
  - **Coding space**
    - Genetic operations work on
  - **Solution space**
    - Where the evaluation operation works on
    - Where the selection operation works on

- **Critical issues**
  - **Legality of a chromosome**
    - Whether a chromosome represents a solution?
      - Problem: No such a solution exists
  - **Feasibility of a chromosome**
    - A solution decoded from a chromosome is a feasible solution?
      - Problem: The solution is infeasible
  - **Uniqueness of mapping**
    - Whether a chromosome represents only one solution and the solutions is represented by the only one chromosome?
      - Problem: A chromosome represents many solutions
      - Problem: Several chromosomes represent the same solution



Strategies

- **Illegality**
  - **Resulted from crossover and mutation operations**
  - **Rejecting method**
  - **Penalizing method**
  - **Repair method**
    - Convert an illegal chromosome to a legal one
    - Better than rejecting and penalizing strategies
- **Infeasibility**
  - **Penalty methods**
  - **Violation amounts are penalized and added to the objective function to degrade the fitness**
- **Uniqueness**
  - **1-to-n**
    - Hard to deal with; should avoid it
  - **N-to-1**
    - Inefficient GA

Encoding Types

- Binary Representation and Encoding
- Integer Representation and Encoding
- Real Representation and Encoding
- Symbol Representation and Encoding

Binary Representation and Encoding

- **Can be applied for various type of optimization problems**
  - **Binary Optimization Problem**
    - Directly use binary encoding
  - **Integer optimization problem**
    - Find the required number of digitals for each integer variable
    - Encode the binary representation for the integer
    - If negative integers are involved, digital shifting operation is conducted
    - One special case of real number optimization problem
  - **Real number optimization Problem**
    - Compute the required numbers of digits to cover the range of each real variable, including the required decimal precision

Real Number Encoded as Binary Genes

Encoding

$l \le x \le u;$

n : number of desired digits in decimal part

m : required length of the binary string for x_i

$2^{m-1} < (u-l) \times 10^n \le 2^m - 1$

Binary string encording : m is calculated first

$b = Encording(x, l, u, m) = B\left(\left\lceil (x-l)(2^m - 1)/(u-l) \right\rceil, m\right)$

$B(i, m)$: Binary expression for integer i in m bits

$Example: -3 \le x = -2.6880 \le 12.1, n = 4;$

$(12.1 - (-3)) \times 10^4 = 151000$

$2^{17} = 131072 < 151000 \le 2^{18} = 262144; m = 18$

$b = Encording(-2.688, -3, 12.1, 18) =$

$B\left(\left\lceil (-2.588 + 3)(2^{18} - 1)/(12.1 + 3) \right\rceil, 18\right) =$

$B(\lceil 5416.46 \rceil, 18) = B(5417, 18) = 000001010100101001$

Decoding

Decording from binary string encording:

$$x = Decoding(b, l, u, m) = l + D(b)\frac{u-l}{2^m - 1}$$

$D(b)$: Decimal value of a binary number b

$$Example: x = Decoding(b, l, u, m)$$

$$Decoding(000001010100101001, -3, 12.1, 18)$$

$$x = -3.0 + D(000001010100101001)\frac{12.1+3}{2^{18}-1}$$

$$= -3.0 + 5417 \times \frac{12.1+3}{2^{18}-1} = -2.6879$$

Crossover Operations for Binary Representation

- **Binary Representation**
  - One-point
  - Two-point
  - N-point

01110101000100101100101010001
10010111010001010001010000100

01110101000100101001010001000
10010111010001010100101010001

01110101000101000100100010000
10010111010010100100100101010

01110101010010101001000100
10010111001010001010010101

Mutation Operation for Binary Representation

- **Mutation Operator**
  - **Determine the number of genes to be mutated**
  - **Randomly determine the positions of these genes in the whole population**
  - **Duplicate the parents whose genes are selected as proto-children**
  - **Alter the genes' values to resulting new offspring**

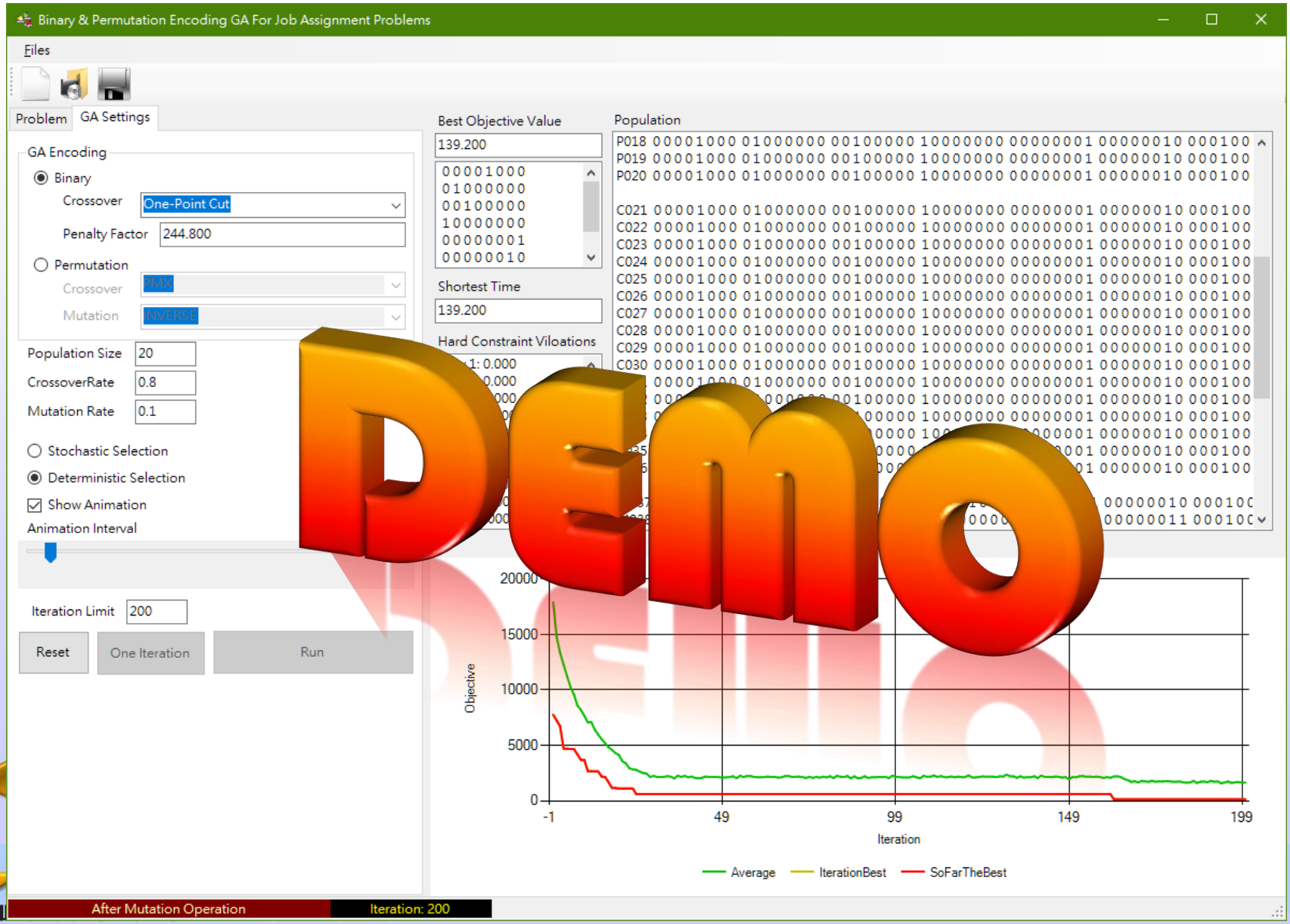
```
01110101000101000100
10001000100010010100
11010101111101010101
10101010001000100010
11010101000010001000
11101010001000100100
11110010101010010100
11101001010010100101
```

P_m=0.04
Pop x *length* = 8x20 =160
Number of mutated genes = 0.04x160= 6

Offspring

```
01110101100101000100
10101110000000110010
11010100000010001000
11101001010010100001
```

Binary Encoding GA for Job Assignment



Integer Representation and Encoding

- For sequencing and grouping optimization problems
- Directly use integer symbols to encode the chromosomes
  - E.g.;
    - [5 3 2 1 6 8 7 4 9] : for permutation encoding
    - [2 2 1 3 3 1 1 2 4] : for object grouping problems

Crossover Operations for Integer Encoding

- **Permutation representation (E.g., TSPs, JSPs, SAPs)**
  - **Canonical Approach:**
    - Extension of two-point or multi point crossover
    - Repairing procedures are embedded
    - Based on blind random mechanism
      - No guarantee that offspring is better
    - Operators
      - Partial-mapped Crossover (PMX)
      - Order Crossover (OX)
      - Position-based Crossover
      - Order-based Crossover
      - Cycle Crossover (CX)
      - Subtour Exchange
  - **Heuristic Approach**
    - Problem-dependent Heuristic Crossovers

TSPs: Traveling Salesman Problem; JSPs: Job shop Scheduling Problems; SAPs: Single line Assembly Problems

Partial-mapped Crossover PMX

- Proposed by Goldberg and Lingle
  - Step 1: Select two positions to obtain a substring (mapping section)
  - Step 2: Exchange the substring of parents to produce proto-children
  - Step 3: Determine the mapping relationship between two mapping sections
  - Step 4: Legalize offspring with the mapping relationship

Parent 1	1	2	3	4	5	6	7	8	9
Parent 2	6	4	5	7	9	8	3	1	2

Proto-child 1	1	2	5	7	9	8	7	8	9
Proto-child 2	6	4	3	4	5	6	3	1	2

5	7	9	8
3	4	5	6

3←→5←→9
4←→7
6←→8

Offspring 1	1	2	5	7	9	8	4	6	3
Offspring 2	8	7	3	4	5	6	9	1	2

Order Crossover (OX)

- Step 1: Select a substring from parent 1
- Step 2: Copying the substring into the corresponding position for proto-child 1
- Step 3: Mark out the genes in the parent 2
- Step 4: Add the unmarked genes left in parent2 sequentially into the void positions at proto-child 1 to complete offspring 1
- Step 5: Repeat the same steps starting from parent 2 and against parent 1 to produce offspring 2

Parent 1	1	2	3	4	5	6	7	8	9
Proto-child 1			3	4	5	6			
Parent 2	6	4	5	7	9	8	3	1	2

Offspring 1	7	9	3	4	5	6	8	1	2

Parent 1	1	2	3	4	5	6	7	8	9
Proto-child 2			5	7	9	8			
Parent 2	6	4	5	7	9	8	3	1	2

Offspring 2	1	2	5	7	9	8	3	4	6

Position-based Crossover

- Step1: Randomly select a set of positions
- Step2: Coping genes at these positions to the same position to produce proto-child 1
- Step 3: Mark out the genes of these position in parent 2
- Step 4: Add the unmarked genes in parent2 sequentially into the void positions at proto-child 1 to complete offspring 1
- Step 5: Repeat the same steps starting from parent 2 and against parent 1 to produce offspring 2

Parent 1	1	2	3	4	5	6	7	8	9
Proto-child 1		2			5	6			9
Parent 2	6	4	5	7	9	8	3	1	2

Offspring 1	4	2	7	8	5	6	3	1	9

Parent 1	1	2	3	4	5	6	7	8	9
Proto-child 2		4			9	8			2
Parent 2	6	4	5	7	9	8	3	1	2

Offspring 2	1	4	3	5	9	8	6	7	2

Order-based Crossover

- A variation of position-based crossover
- The order of genes in the selected position in parent 1 is imposed on the corresponding genes in parent 2

Parent 1	1	2	3	4	5	6	7	8	9
Proto-child 1	2		5		6				9
Parent 2	6	4	5	7	9	8	3	1	2

Parent 1	1	2	3	4	5	6	7	8	9
Proto-child 2		6			5	9			2
Parent 2	6	4	5	7	9	8	3	1	2

Offspring 1	2	4	5	7	6	8	3	1	9
Offspring 2	1	6	3	4	5	9	7	8	2

Cycle Crossover (CX)

- Step1: Find the cycle defined by the genes between parents
- Step2: Copy the genes in the cycle to a child with the corresponding positions of parent 1
- Step3: Mark out the genes of these position in parent 2
- Step 4: Add the unmarked genes in parent2 sequentially into the void positions at proto-child 1 to complete offspring 1
- Step 5: Repeat the same steps starting from parent 2 and against parent 1 to produce offspring 2

Parent 1	1	2	3	4	5	6	7	8	9
Parent 2	6	4	5	7	9	8	3	1	2

Cycle 1: 1-6-8-1
Cycle 2: 2-4-7-3-5-9-2

Proto-child 1	1					6		8	
Parent 2	6	4	5	7	9	8	3	1	2

Parent 1	1	2	3	4	5	6	7	8	9
Proto-Child 2	6					8		1	

Offspring 1	1	4	5	7	8	6	3	8	2
Offspring 2	6	2	3	4	5	8	7	1	9

Subtour Exchange Crossover

- **Step1: Define a subtour length**
- **Setp2: Select subtour from parents that contains common cities**
- **Step3: Exchange subtour to produce two offspring**

Parent 1	1	2	3	4	5	6	7	8	9
Parent 2	6	4	5	7	9	8	3	1	2

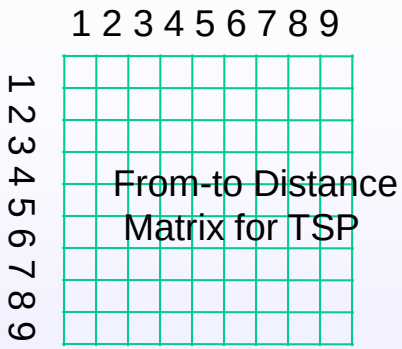
Subtour Length = 4;
Subtour1: 4-5-6-7

Offspring 1	1	2	3	6	4	5	7	8	9
Offspring 2	4	5	6	7	9	8	3	1	2

Heuristic Crossover

- Problem dependent
- Follows the goal of maximizing the fitness
- For example: TSP
  - Step 1: For a pair of parents, pick a random city for the start
  - Step2: Choose the shortest edge leading from the current city which does not lead to cycle.
    - If two edges lead to a cycle, choose a random city to continue
  - Step3: If the tour is completed, stop; otherwise, goto step 2

Parent 1	1	2	3	4	5	6	7	8	9
Parent 2	6	4	5	7	9	8	3	1	2



3 (for offspring 1)
3-shortest(3-4, 3-2, 3-1, 3-8)
3-8
3-8-shortest(8-9,8-7)
3-8-7
3-8-7-shortest(7-6,7-9,7-5)
3-8-7-9
3-8-7-9-shortest(9-1)
3-8-7-9-1
3-8-7-9-1-shortest(1-2)
3-8-7-9-1-2
3-8-7-9-1-2-shortest(2-6)

3-8-7-9-1-2-6
3-8-7-9-1-2-6-shortest(6-5,6-4)
3-8-7-9-1-2-6-5
3-8-7-9-1-2-6-5-shortest(5-4)
3-8-7-9-1-2-6-5-4

Offspring 1	3	8	7	9	1	2	6	5	4

7 (for offspring 2)
7-shortest(7-9, 7-8, 7-6,)
…

Offspring 2	7	6	4	3	1	2	9	8	5

Mutation Operators

- **Inversion Mutation**
  - Select two positions on a chromosome to invert the substring between these two positions

Parent	1	2	3	4	5	6	7	8	9
Child	1	2	6	5	4	3	7	8	9

- **Insertion Mutation**
  - Select a gene and insert it in a random position

Parent	1	2	3	4	5	6	7	8	9
Child	1	2	4	5	6	3	7	8	9

- **Displacement Mutation**
  - Select a subtour and insert it in a random position

Parent	1	2	3	4	5	6	7	8	9
Child	1	2	6	7	8	3	4	5	9

Reciprocal Exchange Mutation

- Select two positions to swap the genes

Parent	1	2	3	4	5	6	7	8	9
Child	1	2	8	4	5	6	7	3	9

Heuristic Mutation

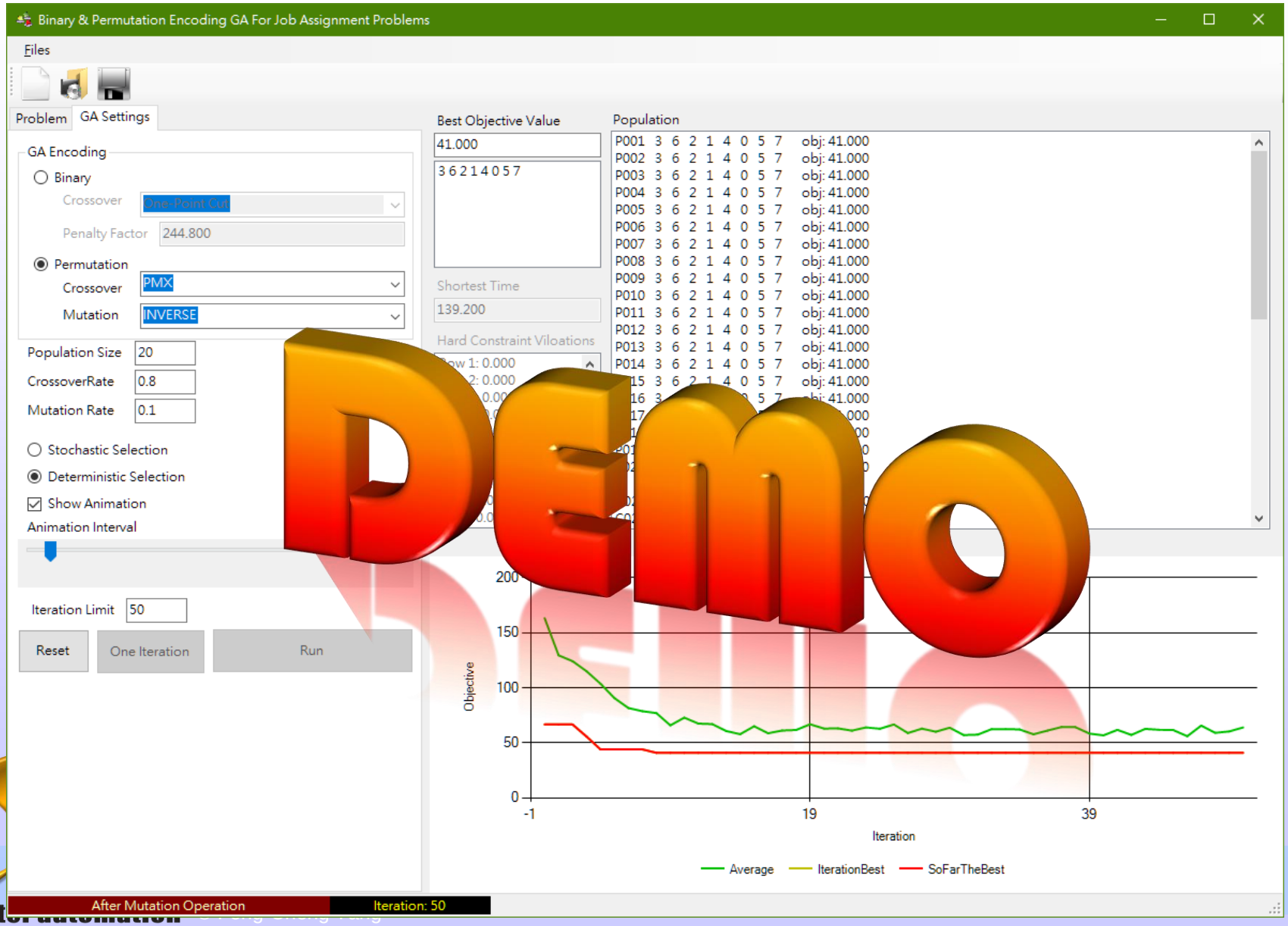
- Step1: pick up *c* genes
- Step2: generate neighbors from all possible permutations of the selected genes
- Step3: evaluate all neighbors and select the best one as offspring

Parent	1	2	3	4	5	6	7	8	9

Neighbor 1	1	2	3	4	7	6	5	8	9
Neighbor 2	1	2	5	4	7	6	3	8	9
Neighbor 3	1	2	5	4	3	6	7	8	9
Neighbor 4	1	2	7	4	5	6	3	8	9
Neighbor 5	1	2	7	4	3	6	5	8	9

Children

Permutation Encoding GA for Job Assignment



Real Number Representation

- Directly use a serial of real numbers to represent genes of a chromosome which represents a solution
  - Each variable should have upper and lower bounds
- Crossover Operators
  - One point or two-point cuts to select segments of genes to be crossovered
  - Apply arithmetical operator on these genes

$$\mathbf{x}_1' = \alpha \mathbf{x}_1 + \beta \mathbf{x}_2$$

$$\mathbf{x}_2' = \beta \mathbf{x}_1 + \alpha \mathbf{x}_2$$

- Convex crossover: $\alpha+\beta=1.0$, $\alpha,\beta \geqq 0.0$
- Affine crossover: $\alpha+\beta=1.0$
- Linear crossover: $\alpha,\beta \in R$

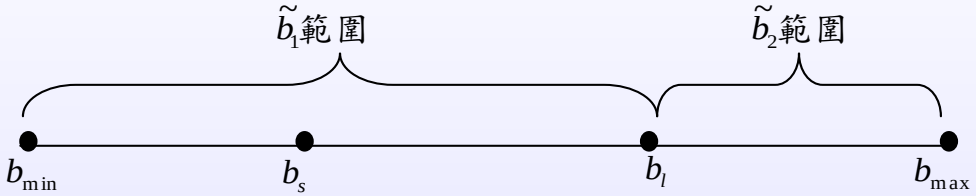
CALab Proposed Five Crossover Operators for Real Representation

- 大值分隔兩段式(Large Value Divided, LVD)
- 小值分隔兩段式(Small Value Divided, SVD)
- 中段和任一端段式(Middle and One End Segments, MOES)
- 兩端兩段式(Two End Segments, TES)
- 中段往復兩段式 (Forward and Backward Middle Segment, FBMS)

五種實數交配演算的交配值落點範圍

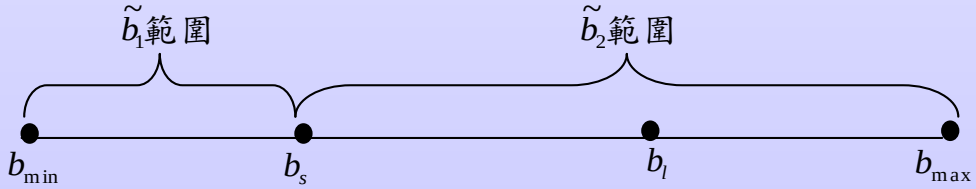
LVD 大值分隔兩段式

$$\tilde{b}_1 = \alpha b_{\min} + (1-\alpha) b_l \qquad \tilde{b}_2 = \alpha b_l + (1-\alpha) b_{\max}$$



SVD 小值分隔兩段式

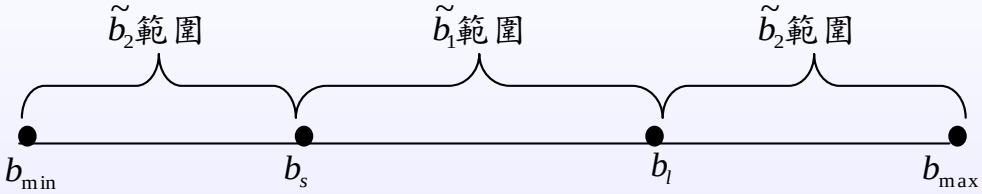
$$\tilde{b}_1 = \alpha b_{\min} + (1-\alpha) b_s \qquad \tilde{b}_2 = \alpha b_s + (1-\alpha) b_{\max}$$



五種實數交配演算的交配值落點範圍

MOES 中段和任一端段式

$$\tilde{b}_1 = \alpha b_s + (1-\alpha) b_l \qquad \tilde{b}_2 = \begin{cases} \alpha b_l + (1-\alpha) b_{\max}, & \text{if } \sim \mathrm{U}(0,1) > 0.5 \\ \alpha b_{\min} + (1-\alpha) b_s, & \text{otherwise} \end{cases}$$

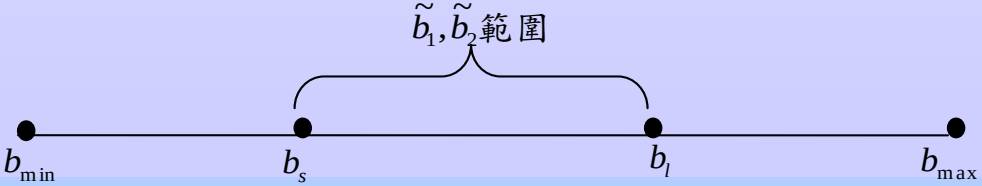


TES 兩端兩段式

$$\tilde{b}_1 = \alpha b_{\min} + (1-\alpha) b_s \qquad \tilde{b}_2 = \alpha b_l + (1-\alpha) b_{\max}$$

FBMS 中段往復兩段式

$$\tilde{b}_1 = \alpha b_s + (1-\alpha) b_l \qquad \tilde{b}_2 = \alpha b_l + (1-\alpha) b_s$$



Mutation Operation for Real Number Representation

Dynamic Mutation (Uniform Mutation)

- **For fine tuning for high precision**

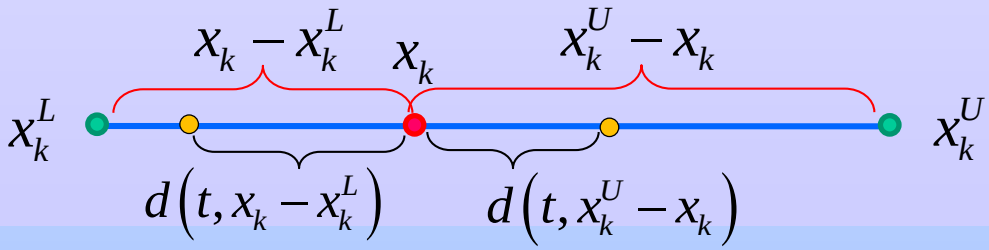
$$\boldsymbol{x} = [x_1 x_2 \cdots x_n],\ \boldsymbol{x}' = [x_1 x_2 \cdots x_k' \cdots x_n]$$

$$x_k' = \begin{cases} x_k + d\left(t, x_k^U - x_k\right) \\ x_k - d\left(t, x_k - x_k^L\right) \end{cases},\ d(t, y) = \sim U(0,1) \cdot y \cdot \left(1 - \frac{t}{T}\right)^b$$

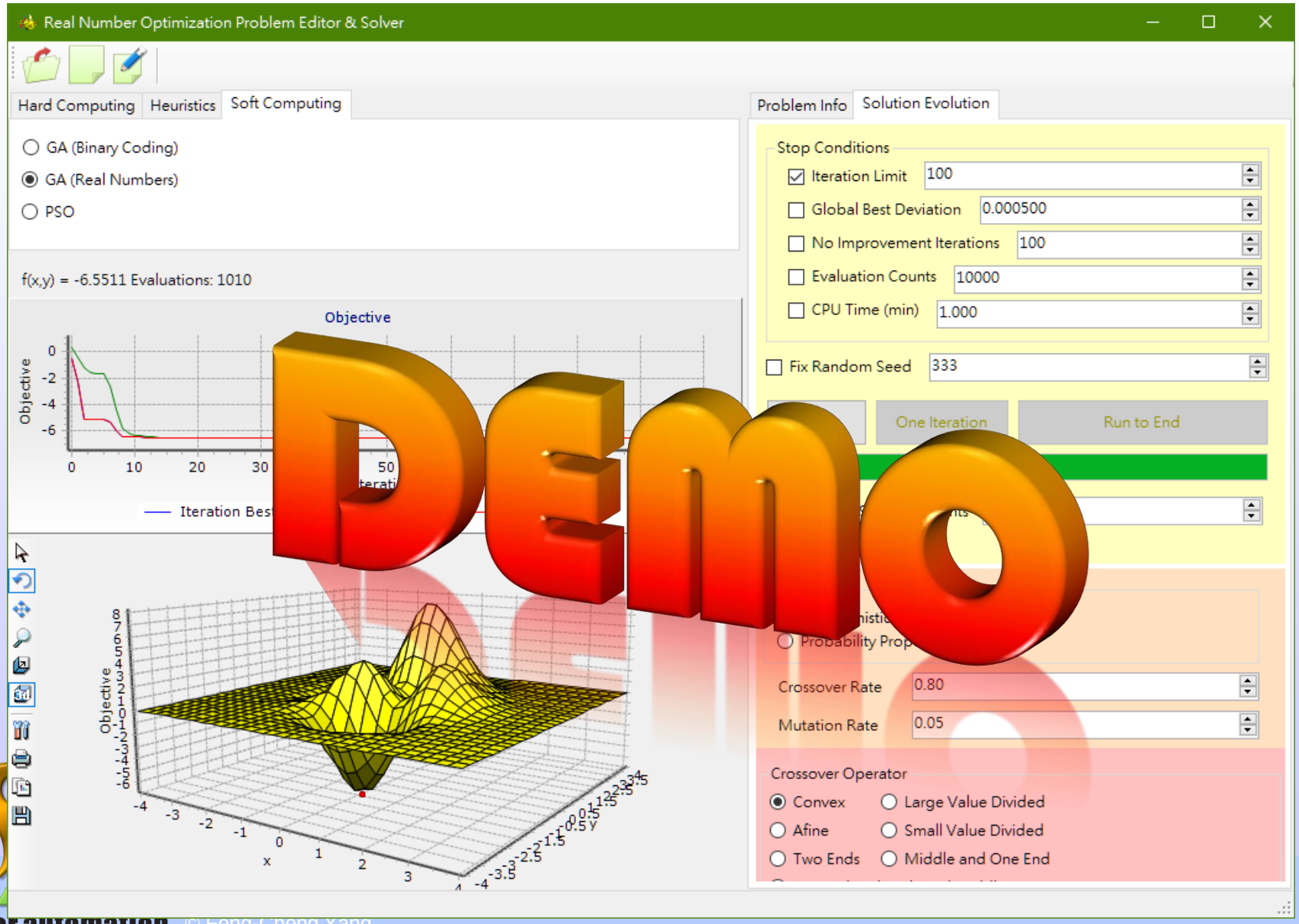
t: iteration index

T: total number of iterations

b: parameter determines the degree of nonuniformity



Real Number Encoding GA for Continuous Optimization Problems



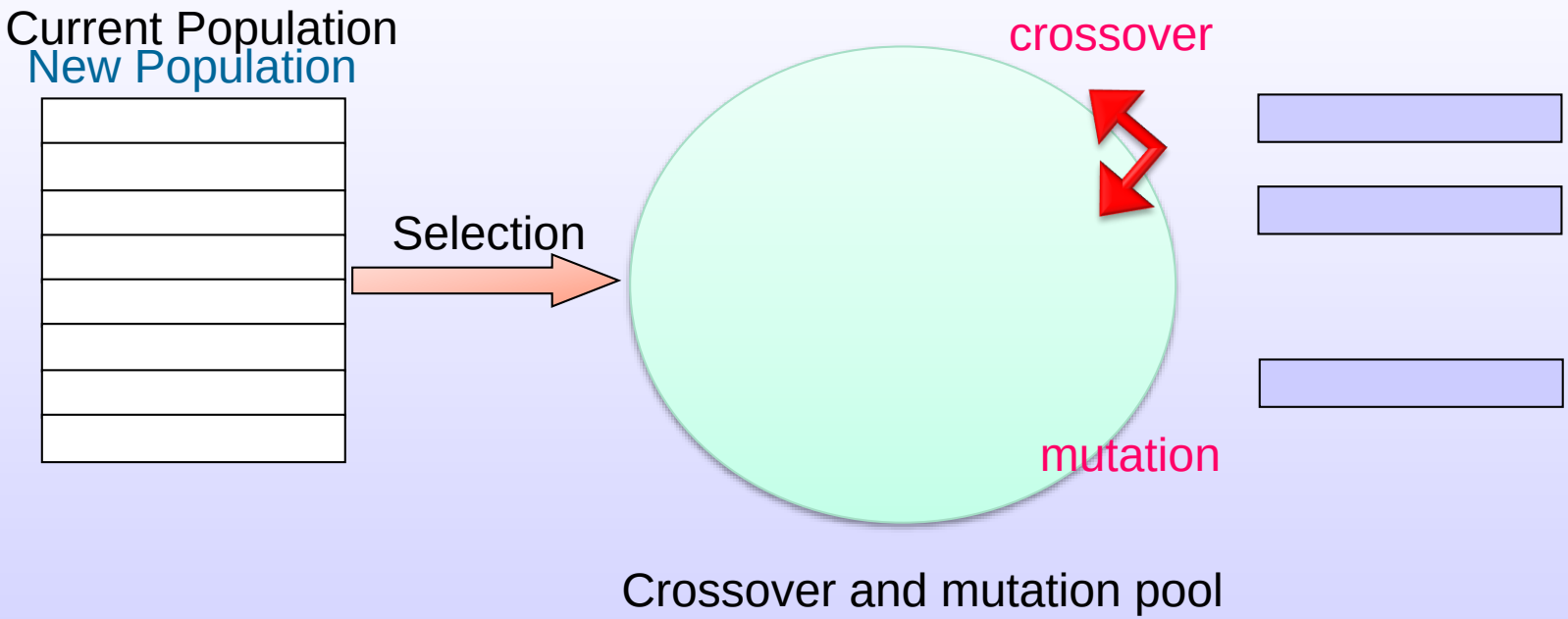
Selection Operation (篩選)

- Darwinian natural selection
- A critical step in GA:
  - A inappropriate selection:
    - Search will terminate prematurely
    - Progress will be slower than necessary
- Low selection pressure ← at the start
  - Favor for a wide exploration of search space
- High selection pressure ← at the end
  - To exploit the most promising region

Genetic Selection

- **Regular sampling space for selection**
  - **Size:** *pop_size*
  - **Selection targets: parents**
  - **Holland's original GA**
    - Randomly select chromosomes from the current generation
    - Parents are replaced by their offspring at once
    - Replacement Strategies
      - select parents for reproduction & offspring replaces parents

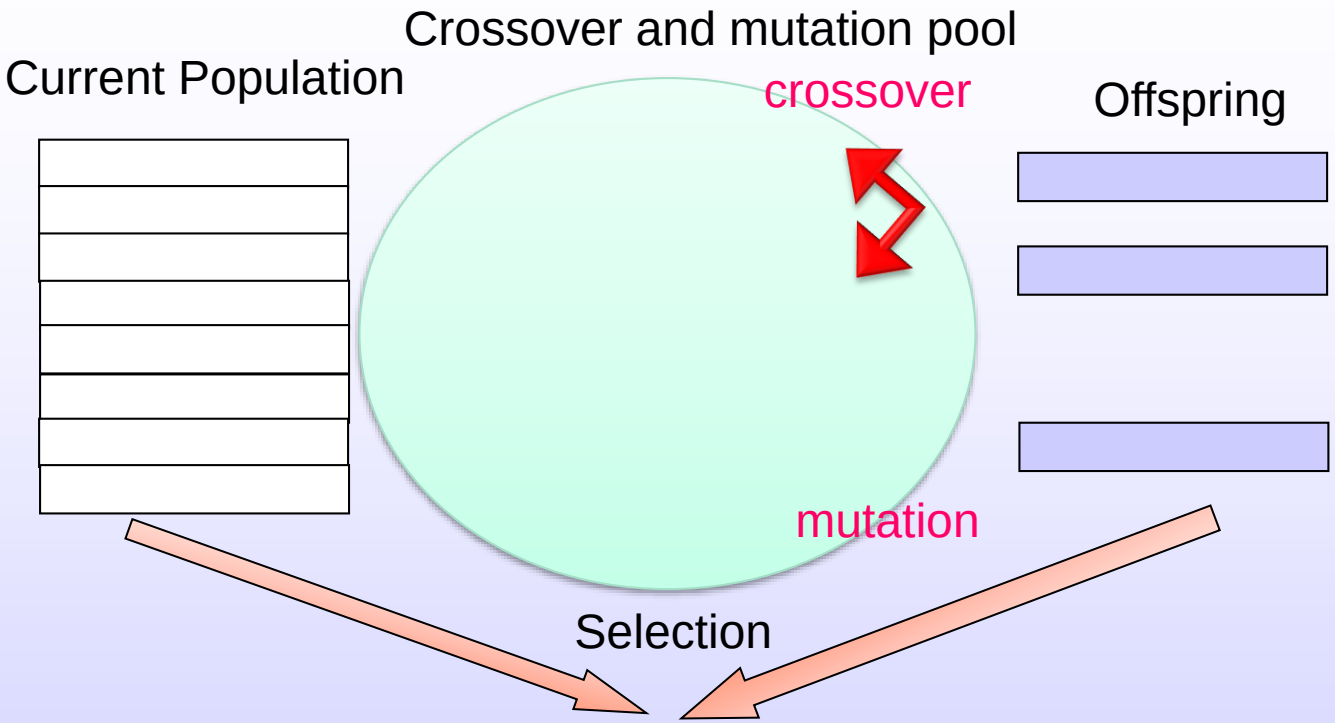
Holland's Selection and Replacement



Enlarged sampling space

- **Enlarged sampling space for selection**
  - **Size:** *pop_size* **+** *off_size*
  - **Selection targets: all parents and offspring**
  - **Both parents and offspring have the same chance of competing for survival**
  - *pop_size* **best out of offspring and parents are selected (***off_size* **+** *pop_size***)**

Enlarged sampling space



Best Sampling Mechanism

- **Stochastic Sampling**
  - **Number of copies of each chromosome is based on its survival probability**
    - Either 0, 1, or multiple copies are selected for each chromosome
- **Deterministic Sampling**
  - **The best *pop_size* chromosomes are selected from the sampling space**
    - Either 0 or 1 copy is selected for each chromosome
- **Mixed Sampling**

Stochastic Sampling

- **Probability proportionate selection**
  - **Roulette wheel selection**
  - **Selection probability for each chromosome proportional to the fitness value**
  - **Selection process is spinning the roulette wheel** *pop_size* **times**

Roulette wheel selection

$$p_k = \frac{f_k}{\sum_{i=1}^{pop\_size+off\_size} f_i}, \; f_k > 0$$

$$q_k = \sum_{i=1}^{k} p_i, k \in \{1, 2, \ldots, pos\_size + off\_size\}$$

for $i = 1$ to pop_size

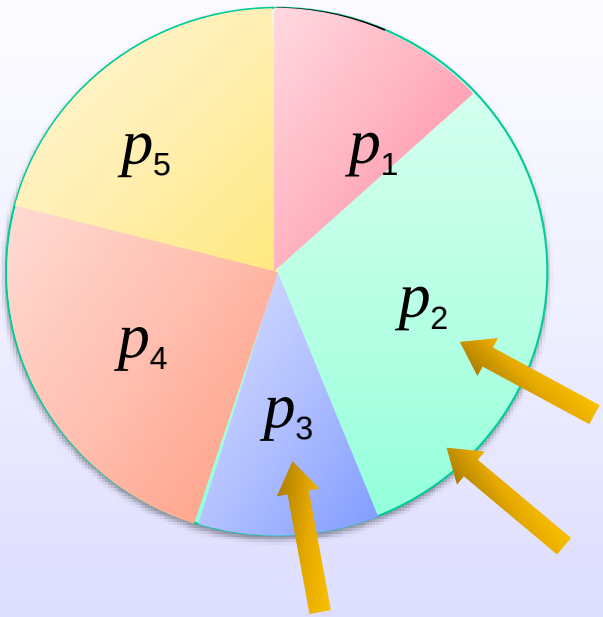
$q \leftarrow U(0,1)$

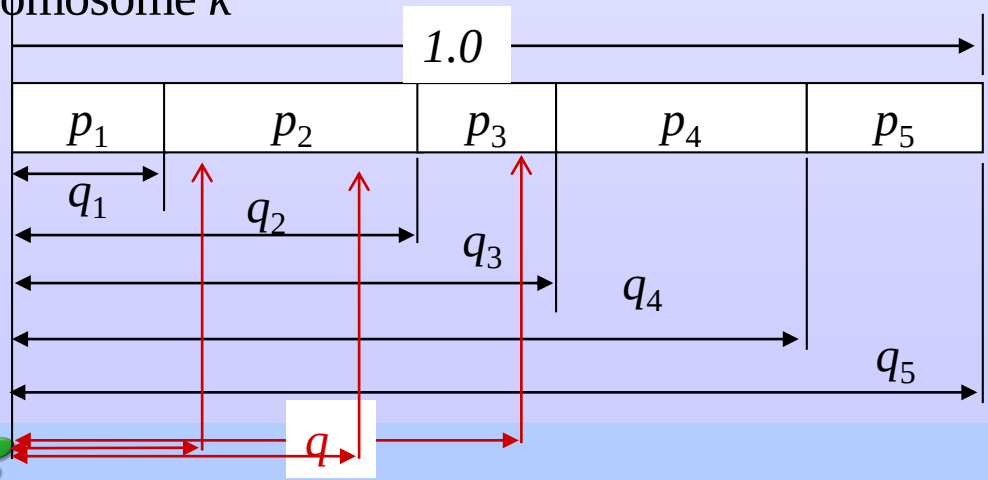
$k \rightarrow 1$

while $(q > q_k)\, k \leftarrow k + 1$

select chromosome k

enfor

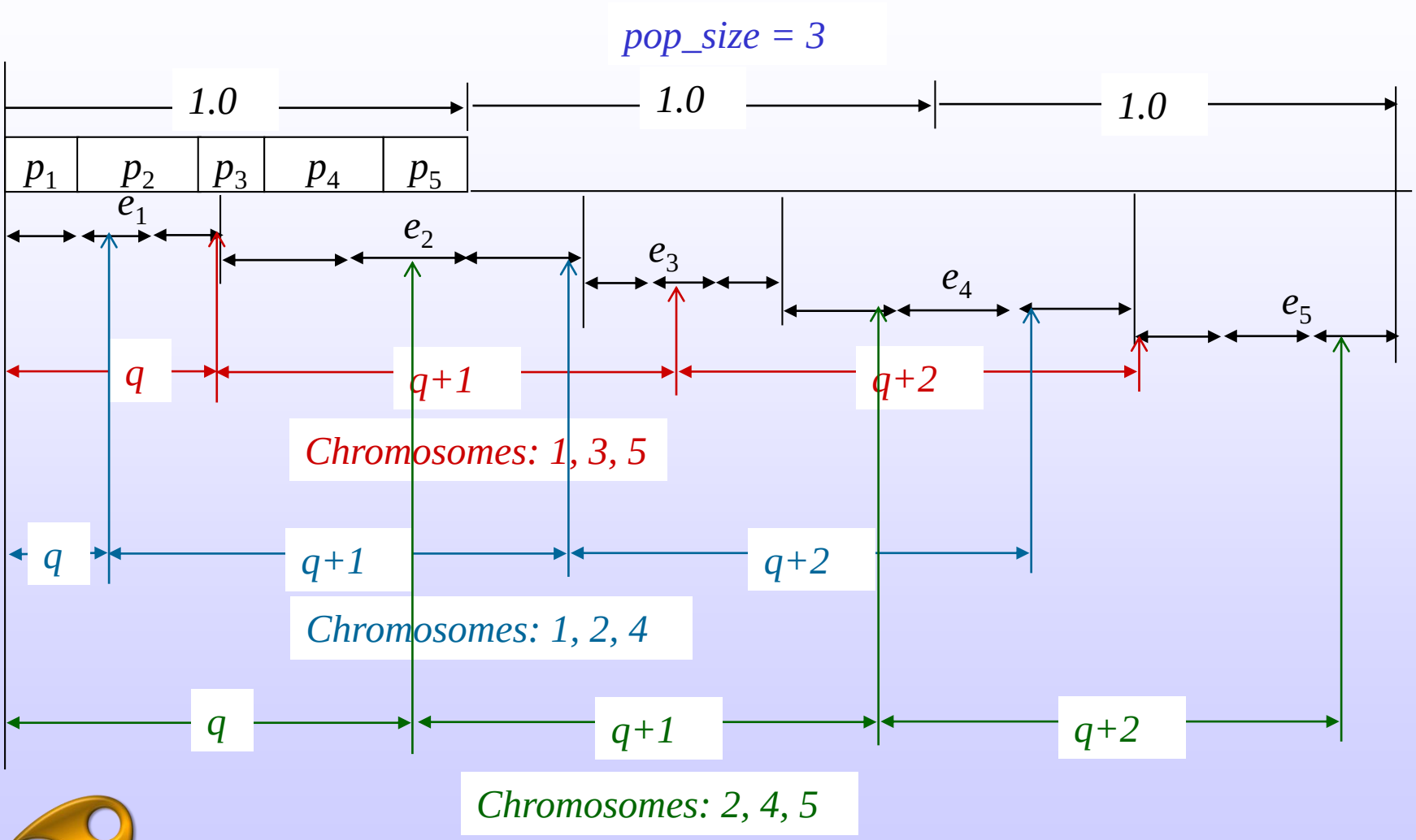




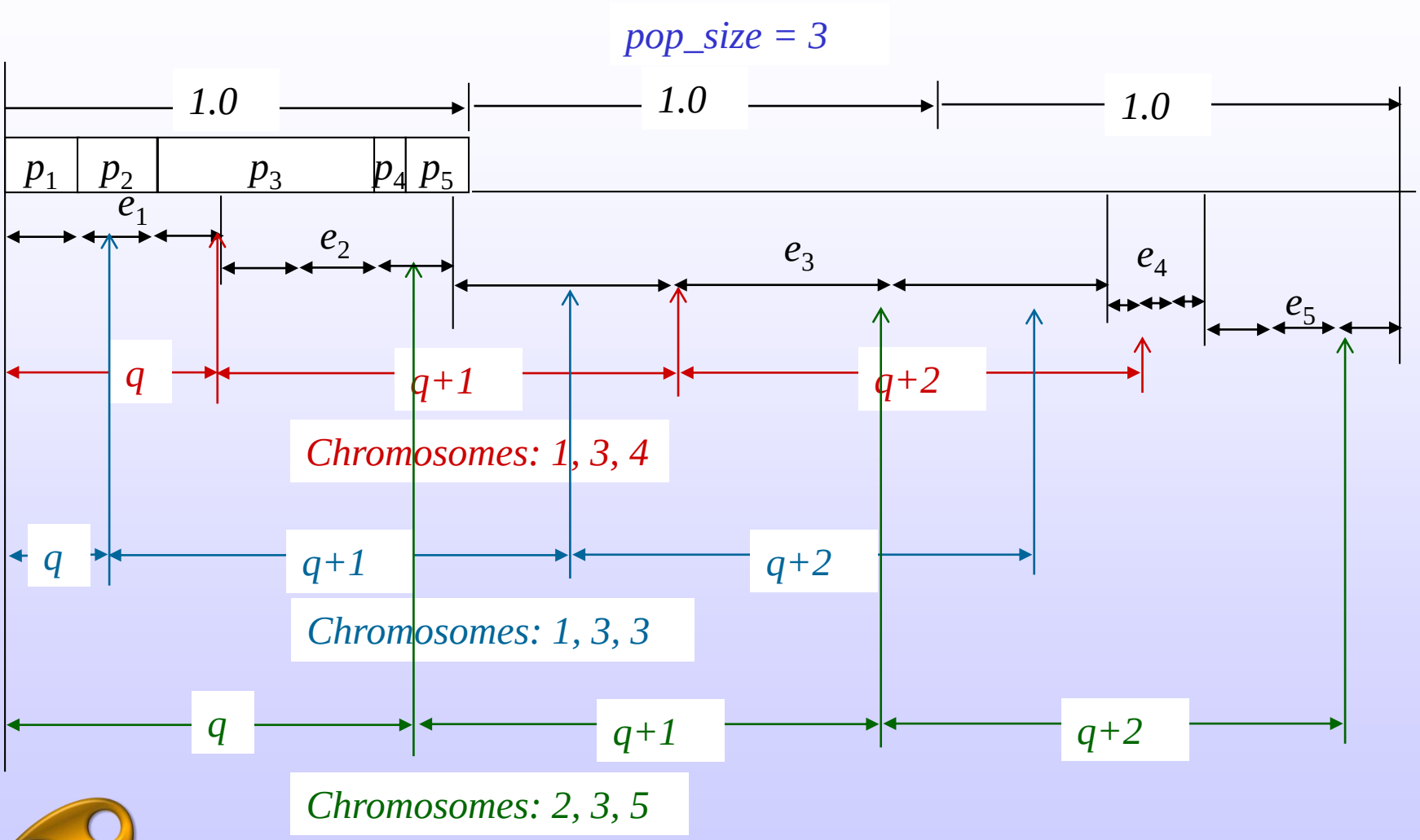
- **Alternative**
  - **Only one roulette wheel turn to get** *pop_size* **chromosomes**

$$e_k = p_k \times pop\_size, \forall k = 1,2,\ldots, pop\_size + off\_size$$
$$sum \leftarrow 0$$
$$q \leftarrow U(0,1)$$
$$count \leftarrow 0$$
$$\textbf{for } k = 1 \textbf{ to } (pos\_size + off\_size)$$
$$\quad sum \leftarrow sum + e_k$$
$$\quad \textbf{while } (sum > q \wedge count < pop\_size)$$
$$\quad\quad \text{select chromosome } k$$
$$\quad\quad count \leftarrow count + 1$$
$$\quad\quad q \leftarrow q + 1$$
$$\quad \textbf{end while}$$
$$\textbf{end for}$$

Demo



Sample (multiple)



Deterministic Sampling

- Best *pop_size* chromosomes are selected from the sampling space
- Truncation Selection
  - Threshold T is defined, T% best chromosomes are selected
  - Each receives 100/T copies
- Block Selection
  - The *pop_size*/s best chromosomes get s copies

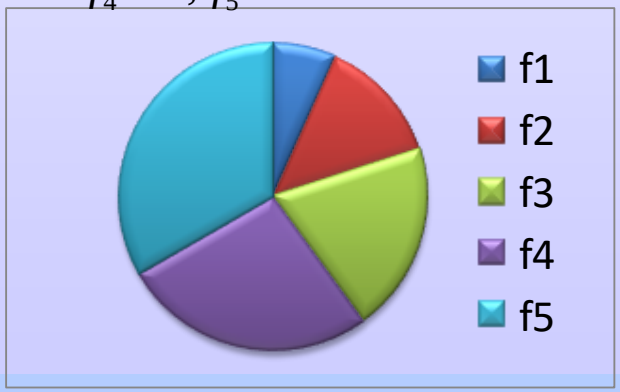
Mixed Sampling

- Contain both random and deterministic features
- Tournament selection
  - Randomly choose *tournament_size* chromosomes and pick the best one
    - Binary tournament: *tournament_size* = 2
- Stochastic tournament selection
  - Roulette wheel method but select two successive chromosomes and pick the best one

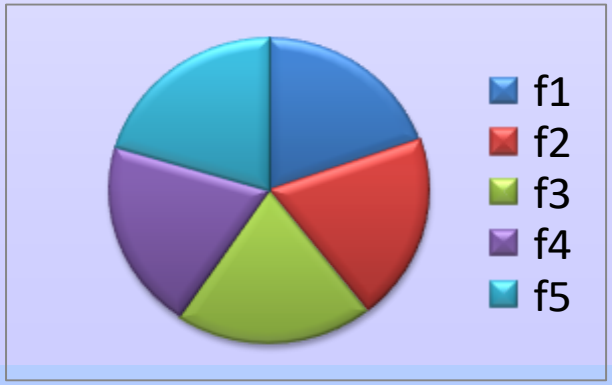
Selection Probability (fitness)

- **Proportional Selection Approach**
  - **Selection probability is proportional to fitness**
  - **Best practice requirement:**
    - Earlier: require low selection pressure; few super chromosomes are recognized
    - Later: require high selection pressure; to exploit the promising solutions

$f_1 = 1,\ f_2 = 2,\ f_3 = 3,$
$f_4 = 4,\ f_5 = 5$



$f_1 = 101,\ f_2 = 102,\ f_3 = 103,$
$f_4 = 104,\ f_5 = 105$



- **Improve methods**
  - **Scaling mechanism**
    - Maps raw objective values to positive real values as theirs fitnesses to determine the survival probability
  - **Ranking mechanism**
    - Ignores the actual objective values
    - Uses a ranking of chromosomes to determine the survival probability

Fitness Scaling

- Maintain a reasonable differential between relative fitness ratings of chromosomes
- Prevent a rapid takeover by some super chromosomes to limit competition early, but to stimulate it later

$$Scaled\ fitness: \tilde{f}_k = g(f_k)$$

- Transformation function $g()$: linear scaling sigma truncation, power law scaling, logarithmic scaling, ...
  - Static scaling
  - Dynamic Scaling
    - According to the scatter situation of fitness values
    - According to the number of generations

Scaling

- **Linear Scaling** $\tilde{f}_k = a \cdot f_k + b$
  - **Parameters a and b are selected to let average chromosome receive one offspring and the best receives a specified number of copies (usually two)**
- **Dynamic linear scaling**

$$\tilde{f}_k = a \cdot f_k + b_t$$

$$b_t = -f_{\min}$$

Sigma Truncation

- Improve linear scaling
- Deal with the negative values
- To incorporate the problem dependent information into the mapping

$$\tilde{f}_k = f_k - \left(f_{avg} - c \cdot \sigma \right)$$

- c is a small integer, σ is the population standard deviation, f_{avg} is the average raw fitness

- **Power Law Scaling**

$$\tilde{f}_k = f_k^{\alpha}$$

  - **The value α is problem-dependent, *α = 1.005* is suggested**
  - **When *α* increases, the gap between the best and worst chromosomes increased**
- **Another power law scaling**

$$\tilde{f}_k = \left(a \cdot f_k + b\right)^{\alpha}$$

- **Logarithmic scaling**
  - **For mapping the objective function of minimization problem**

$$\tilde{f}_k = b - \log(f_k),\ b > \log(f_k)$$

- **Windowing**
  - **Use a moving baseline technique**
  - **Maintain constant selection pressure**

$\tilde{f}_k = f_k - f_w$, w: the window size $2 \sim 10$,

f_w: the worst value observed in the w most recent generations

- **Normalizing**
  - **One kind of dynamic scaling**
  - **For maximization problems**

$$\tilde{f}_k = \frac{f_k - f_{\min} + \gamma}{f_{\max} - f_{\min} + \gamma}, \; f_{\max} : \textit{the best raw fitness}$$

  - **γ is a small positive real number**
    - To prevent zero division
- **Boltzmann Selection**

$$\tilde{f}_k = e^{f_k / T}$$

  - **Selection pressure is low when T is high**

Ranking

- Sort the population from the best to the worst
- Assign probability according to the ranking but not its raw fitness
- Linear Ranking

$$p_k = q_{\max} - (k-1) \cdot \frac{q_{\max} - q_{\min}}{pop\_size - 1};$$

$q_{\max}, q_{\min}$: the probability for the best and worst chromosomes

- Exponential Ranking

$$(1)\ p_k = q - (1-q)^{k-1}$$

Large q, stronger selective pressure

$$(2)\ p_k = q^{k-1}$$

q is typically 0.99; the best 1, the worest q^{pop_size-1}

Objective Values to Fitness

o_k : objective value of chromosome k

f_k : fitness of chromosome k

b : minimum fitness

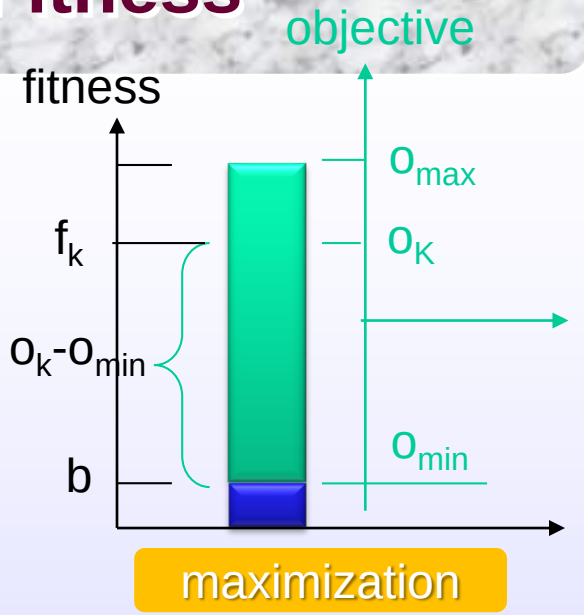
α : least fitness fraction, $0 < \alpha < 0.5$

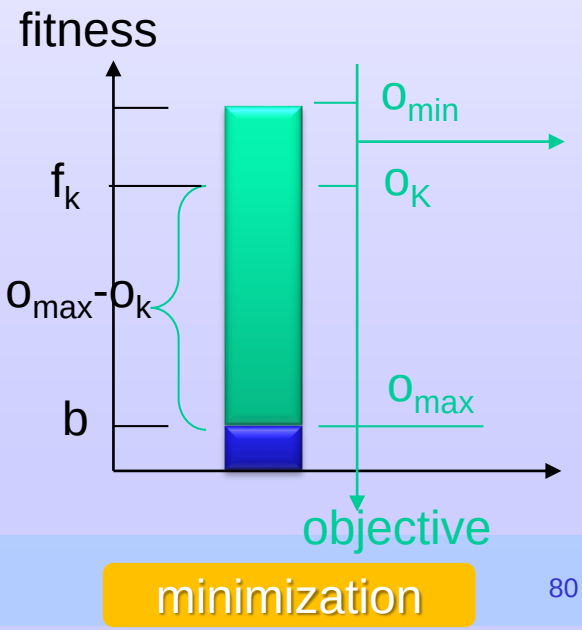
$$o_{\max} = \max_{\forall k}\{o_k\}$$

$$o_{\min} = \min_{\forall k}\{o_k\}$$

$$b = \max\{\alpha(o_{\max} - o_{\min}), 10^{-5}\}$$

$$f_k = b + \begin{cases} o_k - o_{\min}, \text{for maximization problem} \\ o_{\max} - o_k, \text{for minimization problem} \end{cases}$$





GA Applications

- Optimization Problems
  - Multi-objective Optimization Problem
  - Stochastic Optimization Problem
  - Nonlinear Goal Programming
  - Interval Programming
  - Combinatorial Optimization Problem

Application Problems

- Reliability Optimization Problem
  - Network Reliability Design
  - Tree-Based Network Reliability and LAN Design
  - Multiobjective Reliability Design
- Word-Matching Problem
- Traveling Salesman Problem
- Quadratic Assignment Problem
- Film-Copy Deliverer Problem
- Location-Allocation Problem
- Obstacle Location-Allocation Problem

- Set-Covering Problem
  - Airline Crew Scheduling Problem
- Bin-Packing Problem
- Knapsack Problem
  - Binary Representation Approach
  - Order (permutation) Representation Approach
  - Variable-Length Representation Approach

- Minimum Spanning Tree Problem
  - Quadratic MSTP
  - Degree-Constrained MSTP
  - Bicriteria MSTP
- Fuzzy Optimization Problem
  - Fuzzy Linear Programming
  - Fuzzy Nonlinear Programming
  - Fuzzy Nonlinear Mixed-Integer Goal Programming
  - Fuzzy Multiobjective Integer Programming

- **Scheduling Problem**
  - **Job-Shop Scheduling Problem**
    - IP Model
    - LP Model
    - Graph Model
  - **Flow-shop Sequencing Problem**
  - **Grouped Job Scheduling Problem**
  - **Resource-Constrained Project Scheduling Problem**
  - **Single-Machine Scheduling Problem**
  - **Earliness and Tardiness Machine Scheduling Problem**
  - **Parallel Machine Scheduling Problem**
  - **Multiprocessor Scheduling**

- Transportation Problem
  - Linear TP
  - Bicriterial LP
  - Bicriteria Solid Transportation Problem
  - Fuzzy Multicriteria Solid Transportation Problem
- Advanced Transportation Problem
  - Multiobjective Transportation Problem
  - Fixed-Charge Transportation Problem
  - Capacitated Plant Location Problem
  - Bicriteria Transportation Problem with Fuzzy Coefficients

- Network Design and Routing Problem
  - Shortest path Problem
  - Adaptive Network Routing
  - Centralized Network Design Problem
  - Computer Network Expansion Problem
  - Multistage Process Planning
  - M/G/s Queuing Facility Location Problem
- Manufacturing Cell Design Problem
- Facility Layout Design Problem
  - Single-Row MLP
  - Multiple-Row MLP
  - Fuzzy Facility Layout Problem
- Fuzzy Vehicle Routing and Scheduling Problem
- Production Plan Problem