Genetic algorithm (GA)

Intelligent control part II

Overview of GA

- GA = Genetic algorithm
- GA is a category of algorithms for optimization, mainly inspired from biological evolution procedure such as “natural selection”
- GA is suitable for large or complex optimization problem that other deterministic algorithms need too much time.
- GA contains many heuristic operations.
- Outputs of GA strongly depends on its initial state.

Schematic view of GA (1)

①Coding: define “genes” that represent candidates of a solution

1011010 1111010 1011011

1011011 1111011

0011010

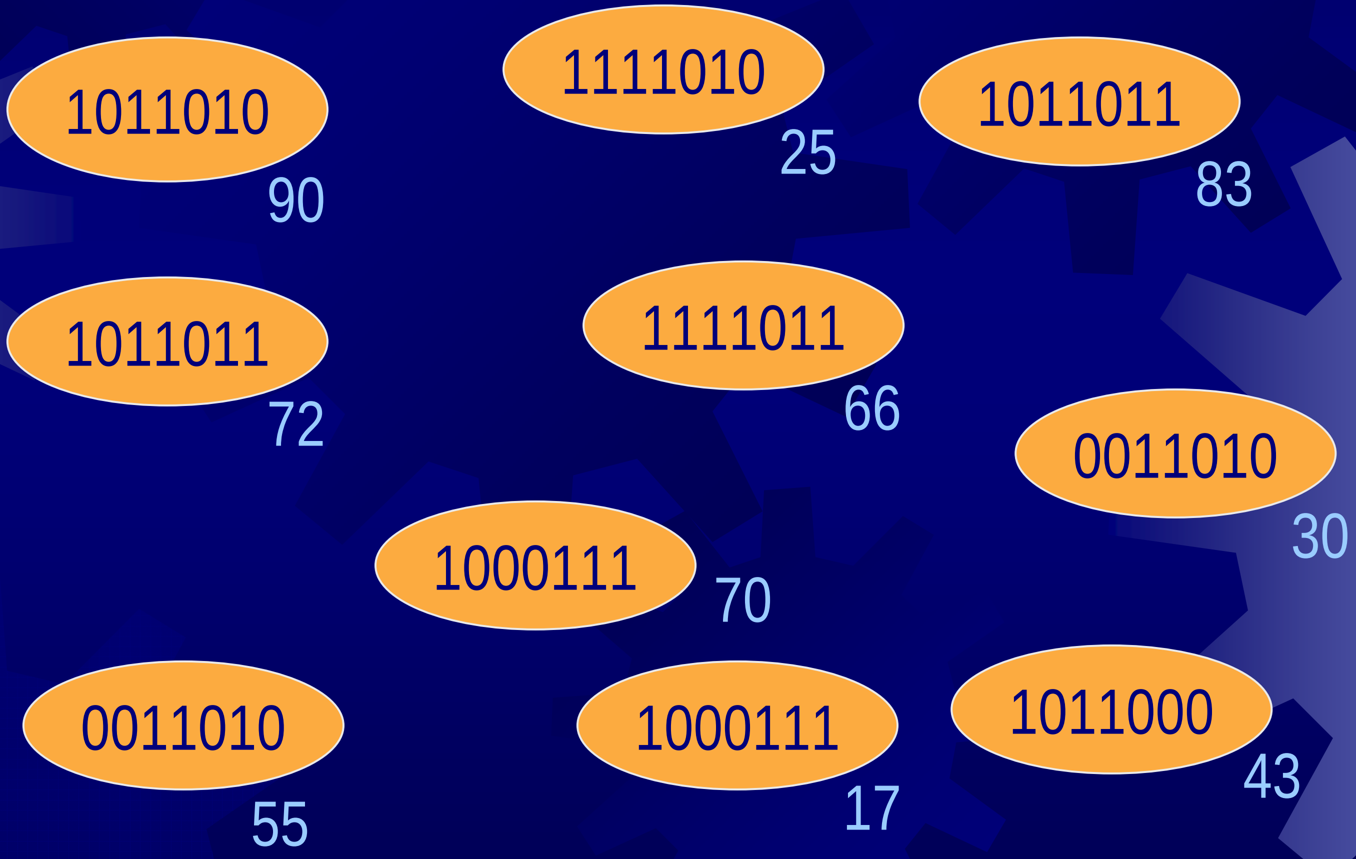
1000111

0011010 1000111 1011000

Population (max. number of genes) must be defined.

Schematic view of GA (2)

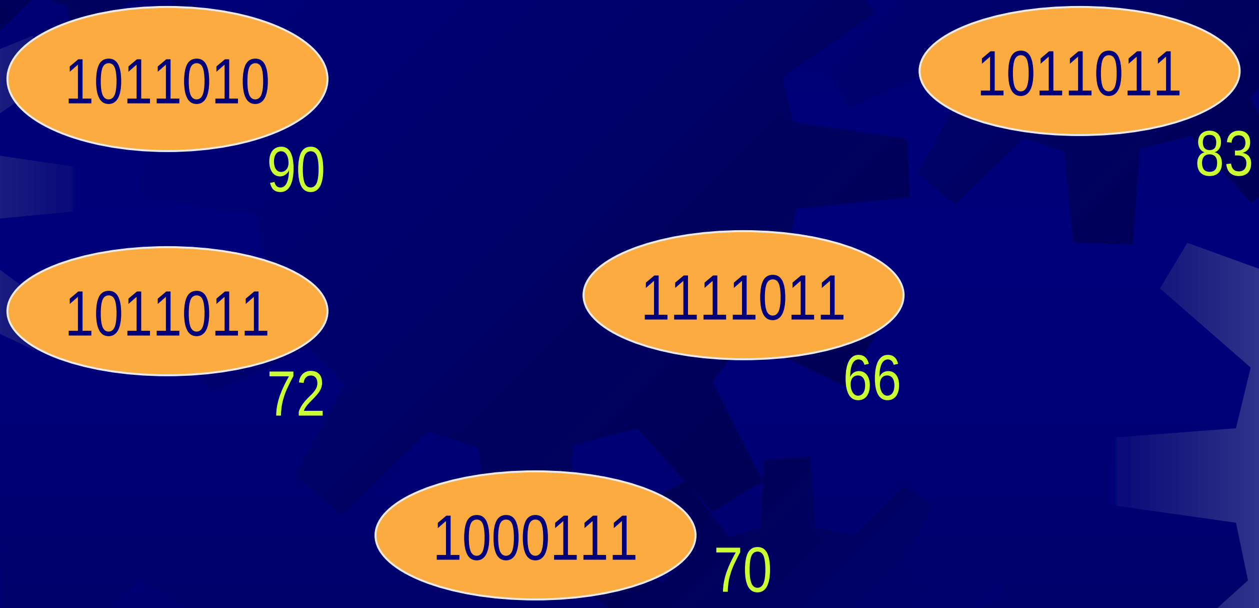
② Selection and reproduction based on evaluations.



A fitness function (defined by a user) evaluates genes.

Schematic view of GA (2)

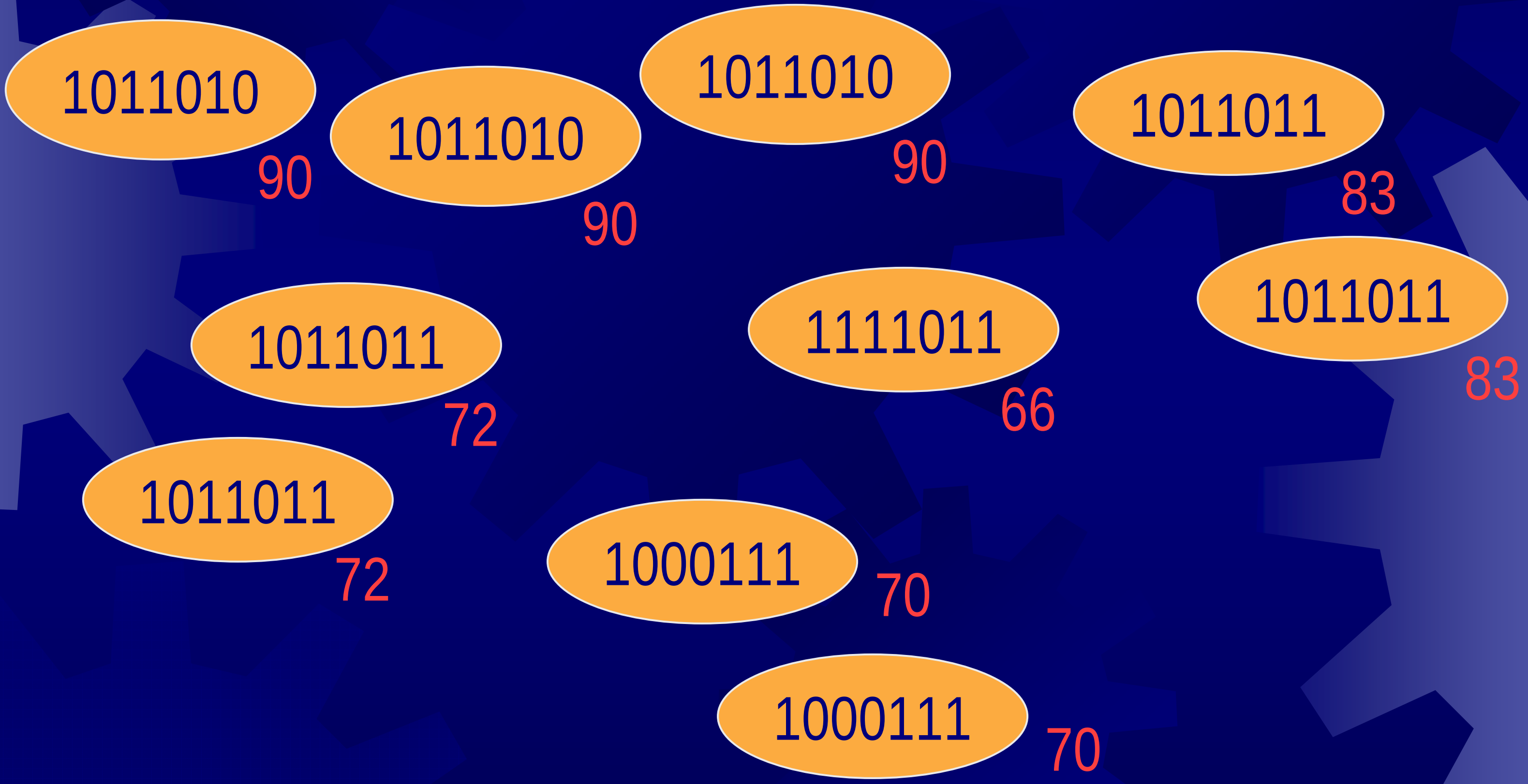
② Selection and reproduction based on evaluations.



A fitness function (defined by a user) evaluates genes.

Schematic view of GA (2)

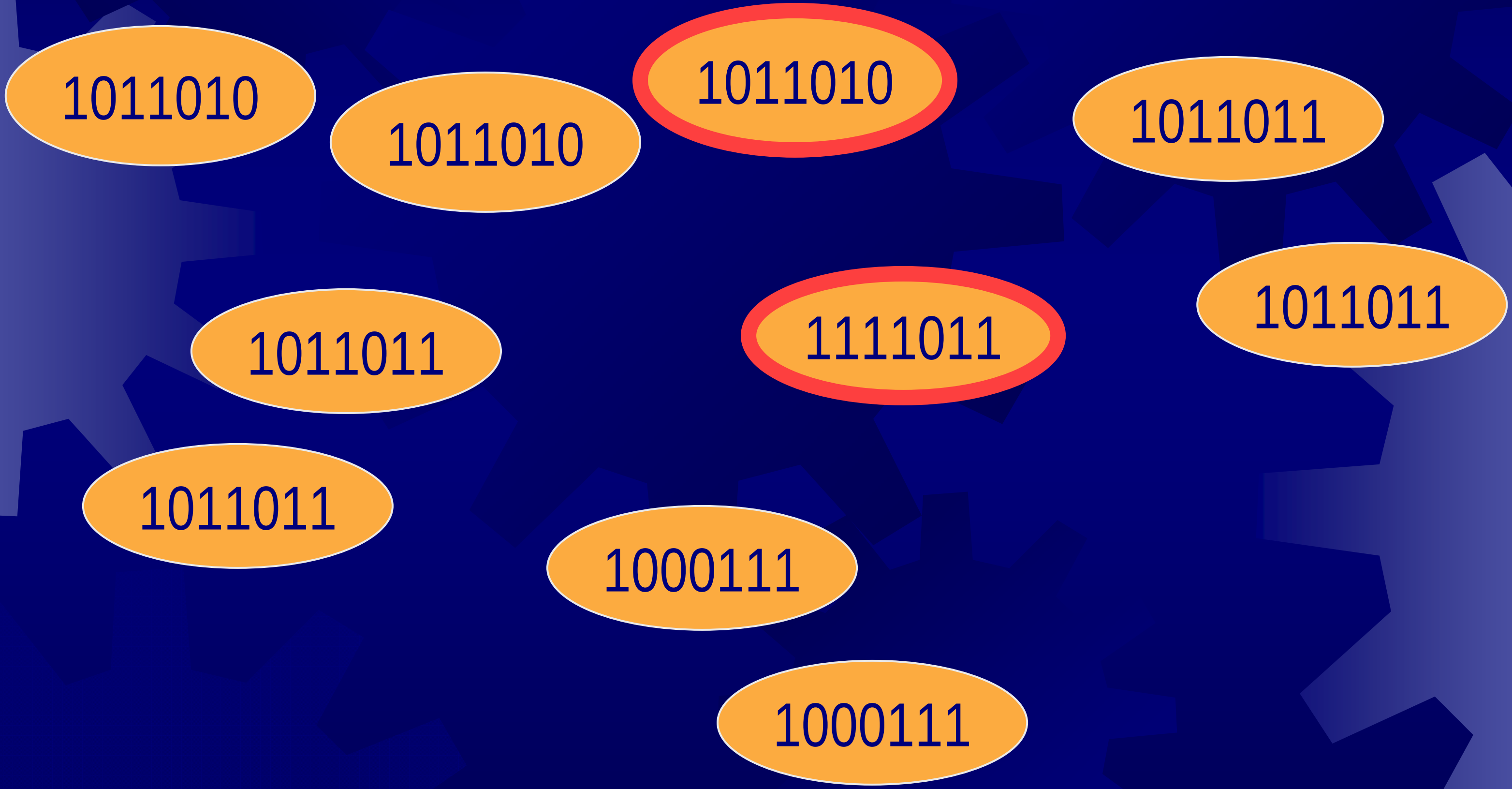
② Selection and reproduction based on evaluations.



A fitness function (defined by a user) evaluates genes.

Schematic view of GA (3)

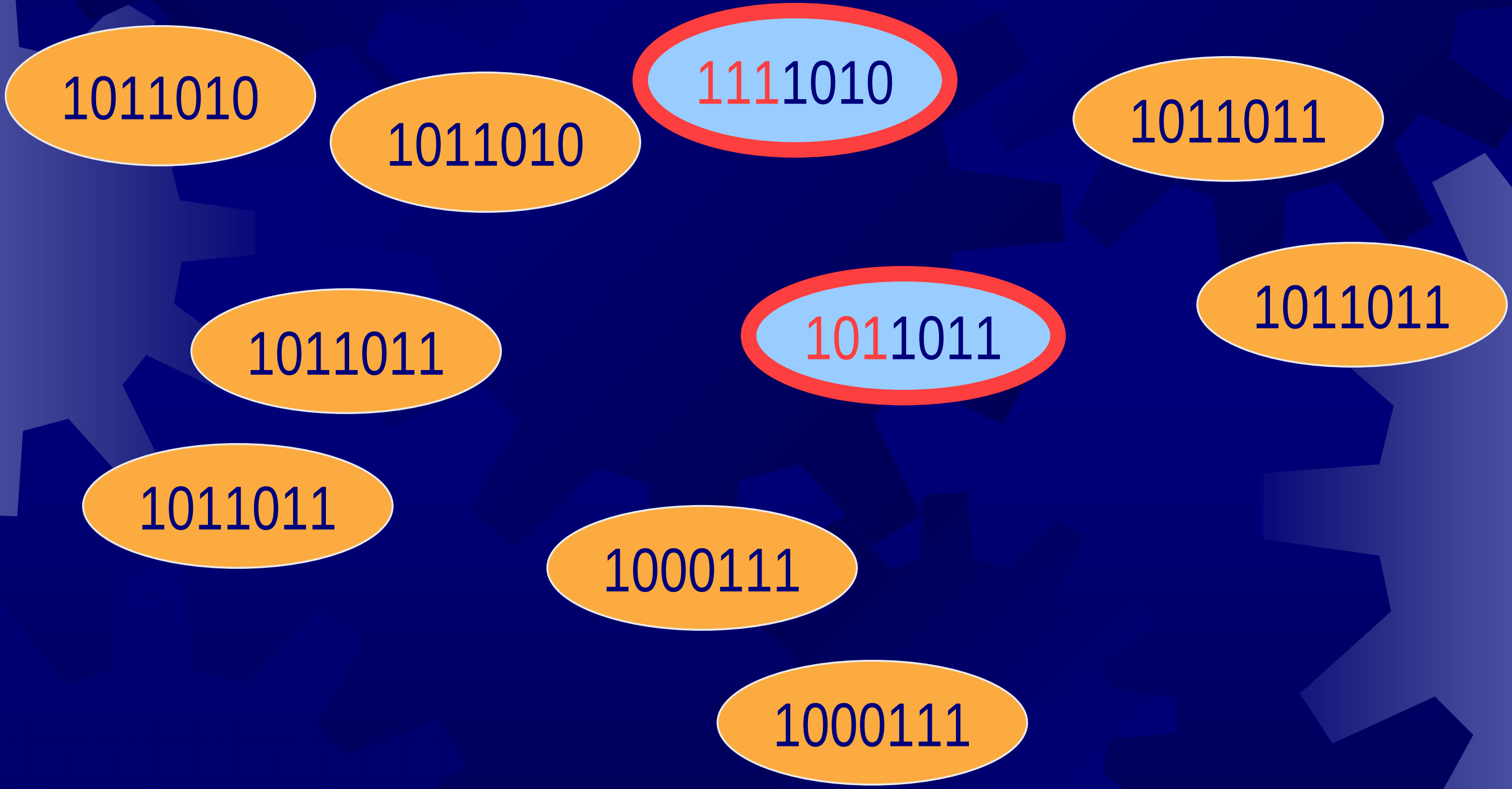
③ Crossover: generate “children”.



Select a pair of “parents”...

Schematic view of GA (3)

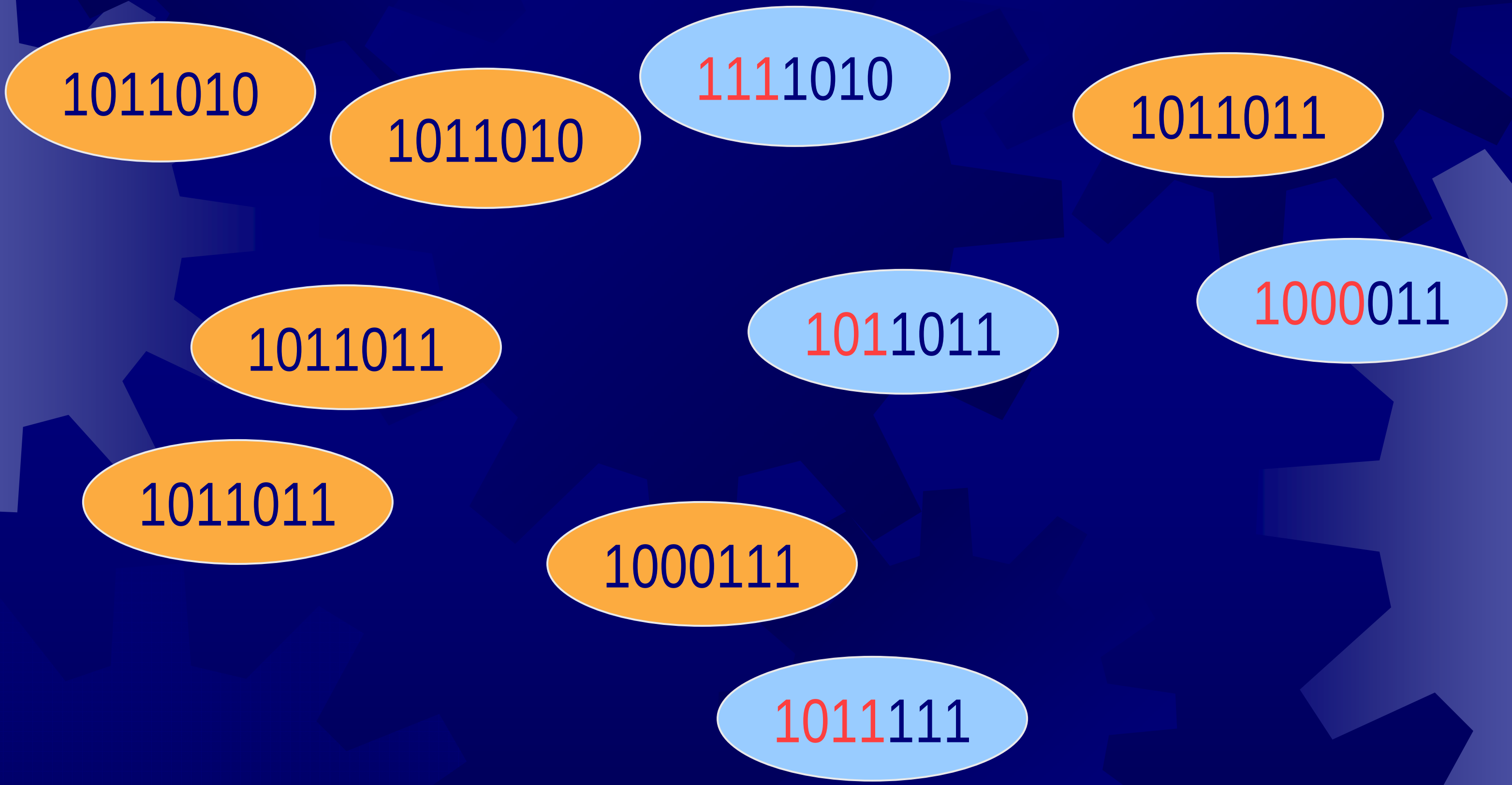
③ Crossover: generate “children”.



Select a pair of “parents”..., and exchange parts of their genes.

Schematic view of GA (3)

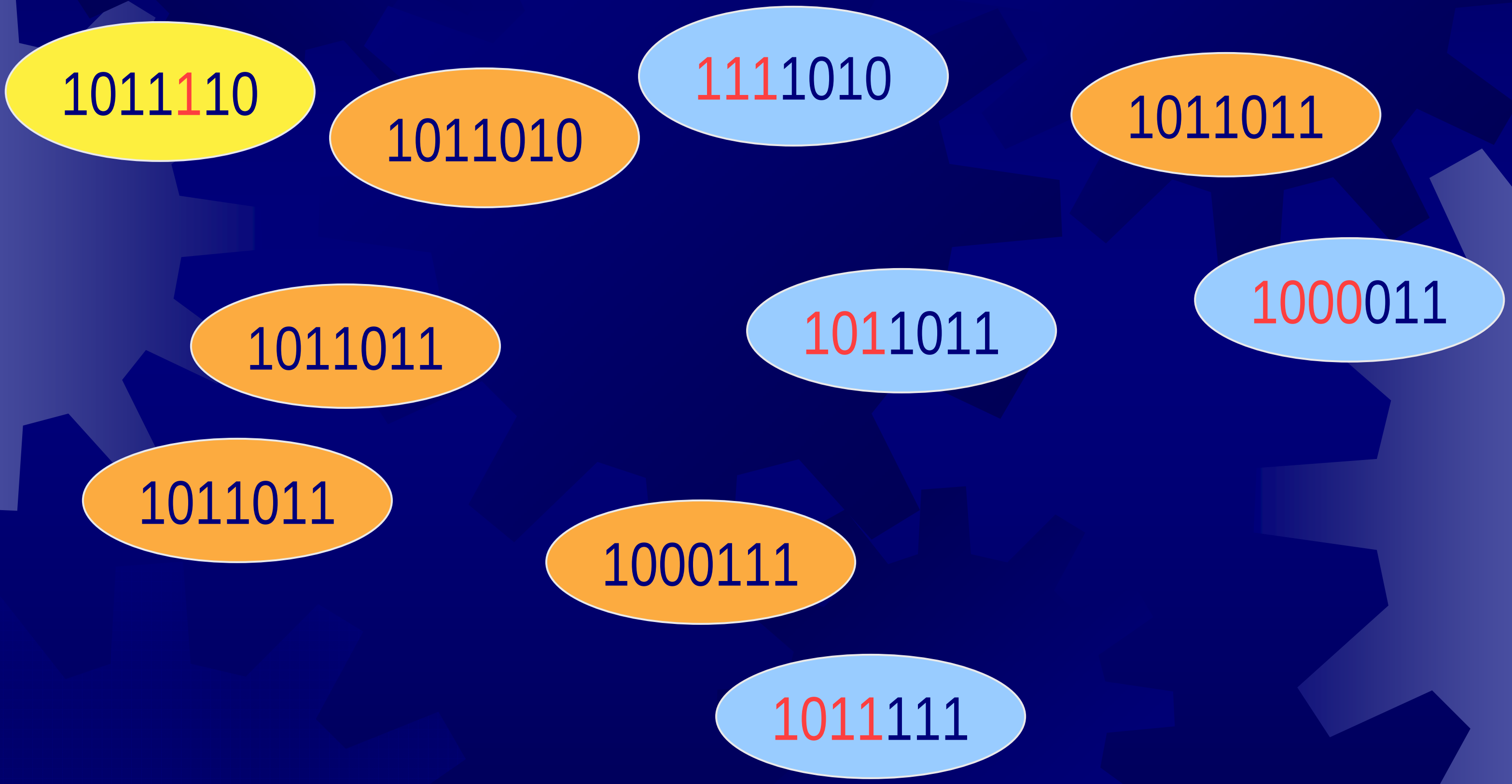
③ Crossover: generate “children”.



Select a pair of “parents”..., and exchange parts of their genes.

Schematic view of GA (4)

④ Mutation: change a part of a gene at random.



Continue ②→③→④

One generation in GA

①Defining a coding, the population, and a fitness function

②Selection and reproduction

③Crossover

④Mutation

Continue

… that is all about GA.

From now on, let us consider algorithms of selection, reproduction, crossover, and mutation, respectively.

Selection and reproduction

- Procedure to keep the population and to select (possibly) good genes.
- A fitness function evaluates genes.
- “Selection” and “reproduction” relates each other.

There are two ideologies;

- A gene have possibility to live according to its fitness.
- Genes that have low fitness must die.

Roulette selection

A gene that has high fitness has high possibility to duplicate it.
A gene that has low fitness may live.

- Selection by “a roulette”.
- A gene that has high fitness occupies large region.
- Iterate selections *population* times.
- Suitable for large population cases.

Probability to be reproduce

Fitness of gene *i*

Sum of fitness values

Expected-value selection

Genes that has low fitness must die.

- An expected value = fitness / population
- Determine number of reproductions according to the value
- Suitable for small population cases.

Fitness
Expected
Reproduction

Example: population = 10

Ranking selection

Genes that has low fitness must die.

- Determine a ranking according to the fitness.
- Reproduce a gene based on its rank.

Rank		Number of reproduction	
1	Gene 5	10	10
2	Gene 2	9	6
3	Gene 8	8	4
4	Gene 1	7	3
.	.	.	.
.	.	.	.
.	.	.	.
		(Linear)	(Non-linear)

Crossover

- A crossover exchanges parts of parent genes.
- “Children” hopefully succeed “good characteristics” of parents.
- Crossover procedures must consider the coding in order to avoid mortal genes.
- Mortal gene = inadmissible answer.
- Here I would like to introduce general methods for crossover.

Simple crossover

(1) Simple crossover (One-point crossover)

Parents

Children

A crossover point is determined at random.

(2) Multipoint Crossover

Parents

Children

Two-points crossover

Three-points crossover

(3) Uniform Crossover

Parents

Using a mask pattern

Children

Mutation

- Change a locus of a gene at random.
- So often mutation results a random search.
- A mutation also consider the coding in order to avoid mortal genes.

```
<1011101000> → <1001101000>
   ↑              ↑
```

Example

Find x that maximize $f(x)$

$$f(x) = x \sin(10\pi x) + 2.0$$

$$(-1.0 \leqq x \leqq 2.0)$$

Error must be smaller than 10^{-5}

Coding

In order to keep the condition (error < 10^{-5}),
we express x by 22bit code.

$$s_1 = <1000101110110101000111>$$

Boundary condition:

$$<0000000000000000000000> = -1.0$$
$$<1111111111111111111111> = 2.0.$$

- In this case, no mortal gene exists.
- At the beginning, we generate genes at random.

Fitness function

In this case, we apply *f(x)* itself as a fitness function.

s_1=<1000101110110101000111> → f(s_1) = 2.586345

s_2=<0000001110000000010000> → f(s_2) = 1.078878

s_3=<1110000000111111000101> → f(s_3) = 3.250650 **BEST**

A result of optimization

TRUE

1.850542

Population = 50, Prob. mutation = 0.01
Simple crossover, Prob. crossover = 0.25
Roulette selection

Solve a TSP by GA

TSP: Traveling Salesman Problem

- Let us assume a salesman who starting from his home city, is to visit exactly once each city on a given list and then return home.

- A TSP problem is a problem such that he selects the order in which he visits the cities so that the total of the distances traveled in his tour is minimum.

- Assume that he knows, for each pair of cities, the distance from one to the other. Then he has all the data necessary to find the minimum, but it is by no means obvious how to use these data in order to get the answer.

- So, TPS is difficult problem.

Calculation cost

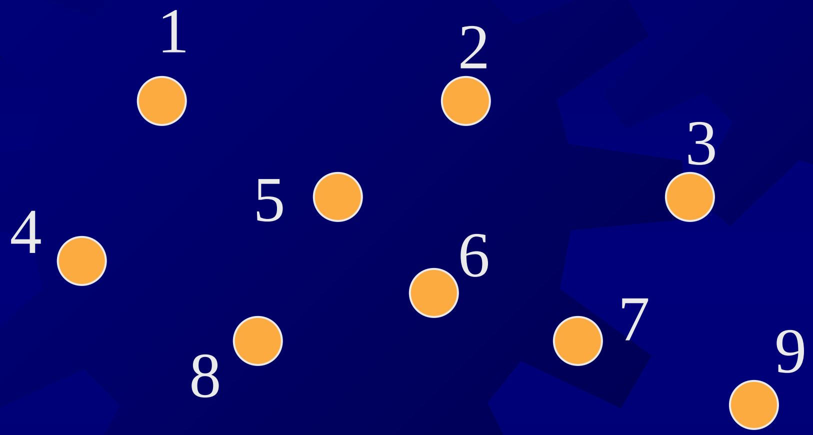
$$\text{number of routes} = \frac{(\text{number of cities} - 1)!}{2}$$

#city	#route	#city	#route

Too many candidates to search

Coding and crossover

Let us use a list of visiting cities as a gene ....



s_1=<12345|6789> → s'_1=<12345|7465>

s_2=<19283|7465> → s'_2=<19283|6789>

(Mortal gene!)

We cannot apply simple crossover to this coding.

We have to change the crossover procedure

Crossovers for TSP

Researchers on the field of GA often use TSP as a benchmark. So, there are many proposals about crossover procedures for TSP.

- Partially Matched Crossover, PMX
- Ordered Crossover, OX
- Cycle crossover ,CX

(1) Partially Matched Crossover (PMX)

(i) Parents

s_1=<123|4567|89>
s_2=<452|1876|93>

(ii) Exchanging

s'_1=<***|**1876**|**>
s'_2=<***|**4567**|**>

Corresponding pairs
1-4, 8-5, 7-6, 6-7

(iii) Insertion

s'_1=<***23**|1876|***9**>
s'_2=<****2**|4567|**93**>

Additional pairs
3-2, 9-3

(iv) Completion

s'_1=<**4**23|1876|**5**9>
s'_2=<**18**2|4567|93>

This crossover loses orders of visiting cities in parent genes.

(2) Ordered Crossover (OX)

(i) Parents

```
s1=<123|4567|89>
s2=<452|1876|93>
```

Order after 2nd crossover point

(ii) Copy

s′$_1$=<***|**4567**|**>
s′$_2$=<***|**1876**|**>

93**45**218**76**

93218

(iii) Insertion of remained genes according to their original orders.

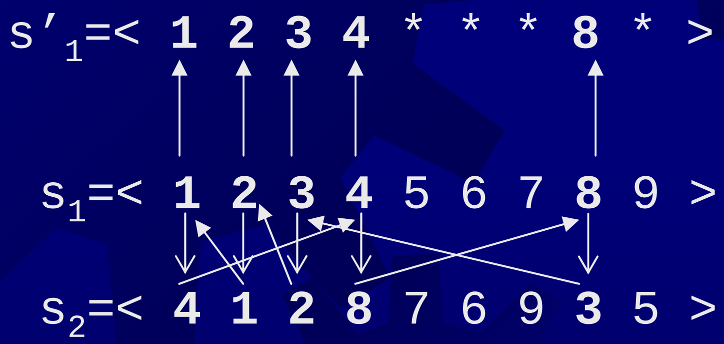
s′$_1$=<**218**|4567|**93**>
s′$_2$=<**345**|1876|**92**>

This crossover loses correspondence between locus and a city.

(3) Cycle crossover (CX)

(i) Parents s_1=<123456789>
s_2=<412876935>

(ii) Find a cycle



s′₁=< **1 2 3 4** * * * **8** * >
s₁=< **1 2 3 4** 5 6 7 **8** 9 >
s₂=< **4 1 2 8** 7 6 9 **3** 5 >

(iii) Exchange remained genes

s'_1=< 1 2 3 4 **7 6 9** 8 **5** >

(s'_2 is applied the same completion)

Demonstration

- Cities：10
- Population：10
- Cycle crossover and ranking selection
- Ratio of mutation: 10%
- Fast, but not global optimum.
- Variation of genes will be lost.

Conclusion

- GA is a category of optimization algorithms that are inspired from natural selection.
- Fast, but no guarantee of global optimum.
- We have to consider a procedure of crossover depends of the coding.