Lecture 2: Genetic Algorithms Introduction

Sabah Sayed

Department of Computer Science

*Faculty of Computers and Artificial Intelligence*

Cairo University

Egypt

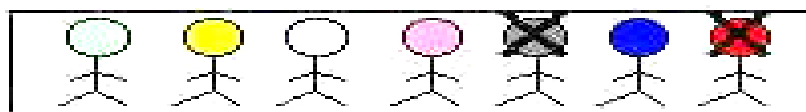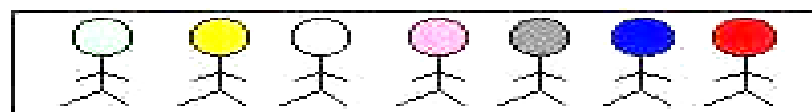# ***Biological Evolution Process***
"Natural Selection" + "Genetic Inheritance"

Basic Idea of Principle of Natural Selection
"Select The Best, Discard The Rest"

Remember

- An individual is characterized by a set of parameters: Genes
- The genes are joined into a string: Chromosome

- The chromosome forms the genotype
- The genotype contains all information to construct an organism: the phenotype

- Reproduction is a "dumb" process on the chromosome of the genotype
- Fitness is measured in the real world of the phenotype
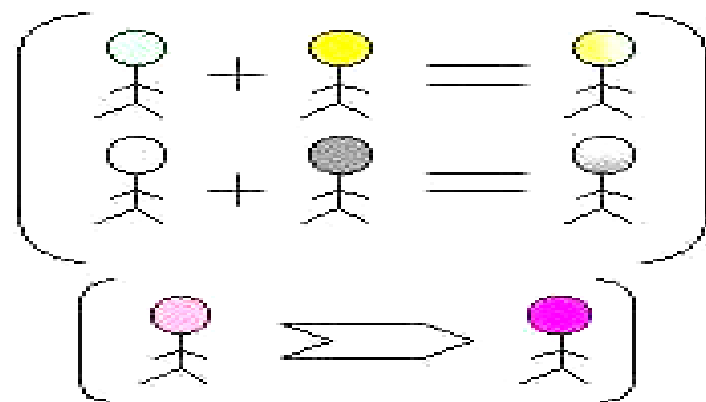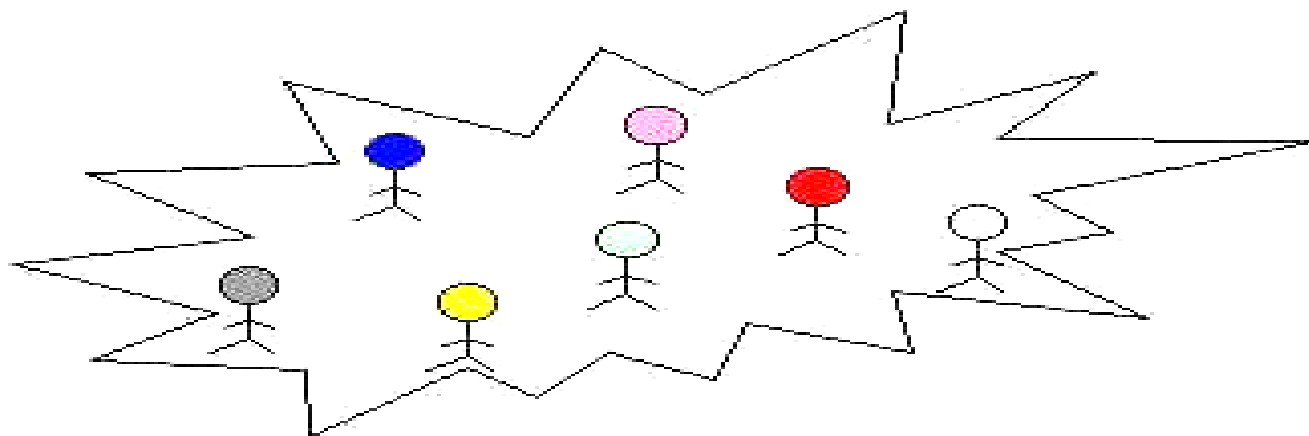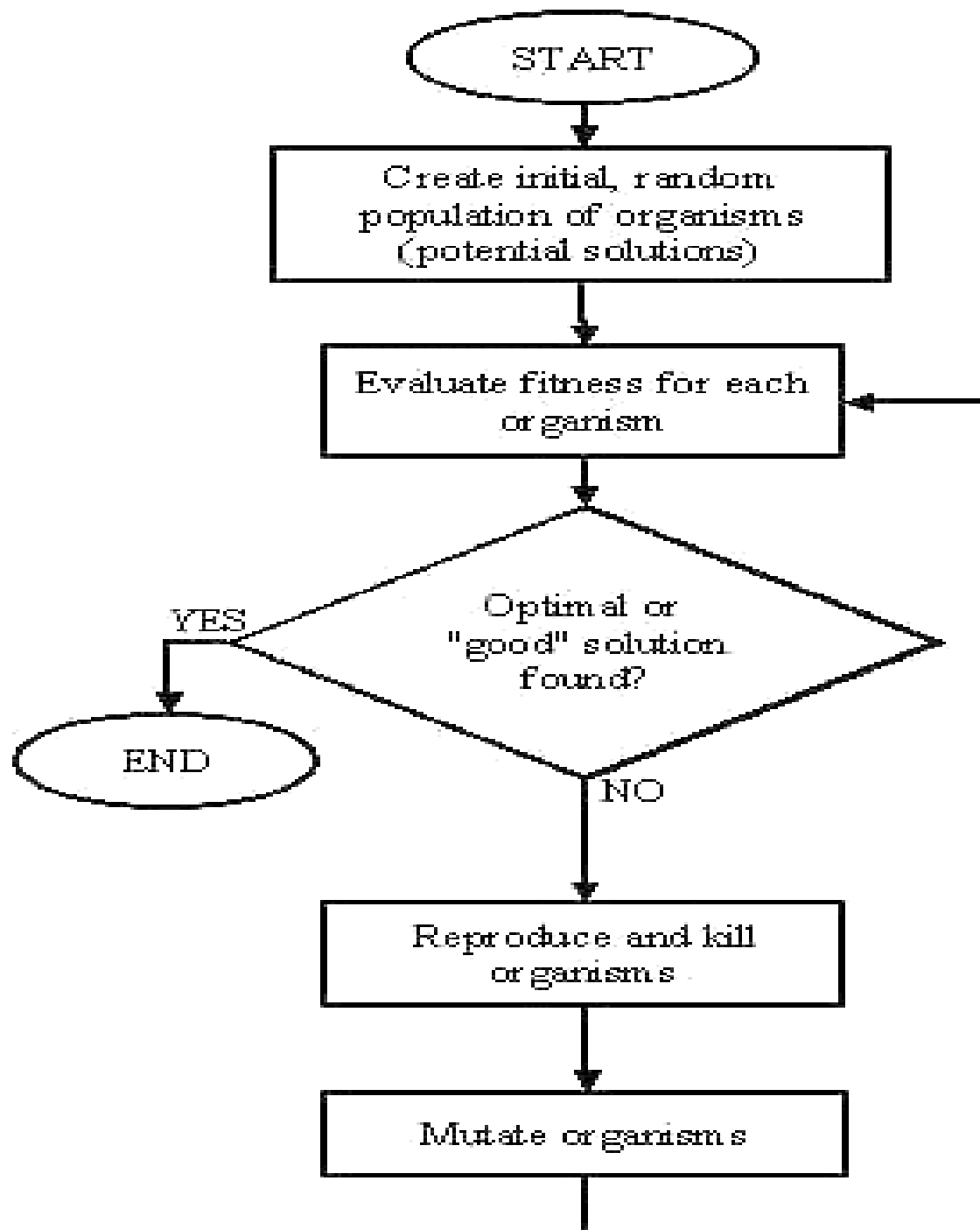
What Are Genetic Algorithms (GAs)?

- Genetic Algorithms are *search* and *optimization* techniques based on principle of *Biological Evolution Process*.

- Genetic Algorithms implement optimization strategies by simulating **evolution of species** through **Natural Selection**.

Working Mechanism Of GAs
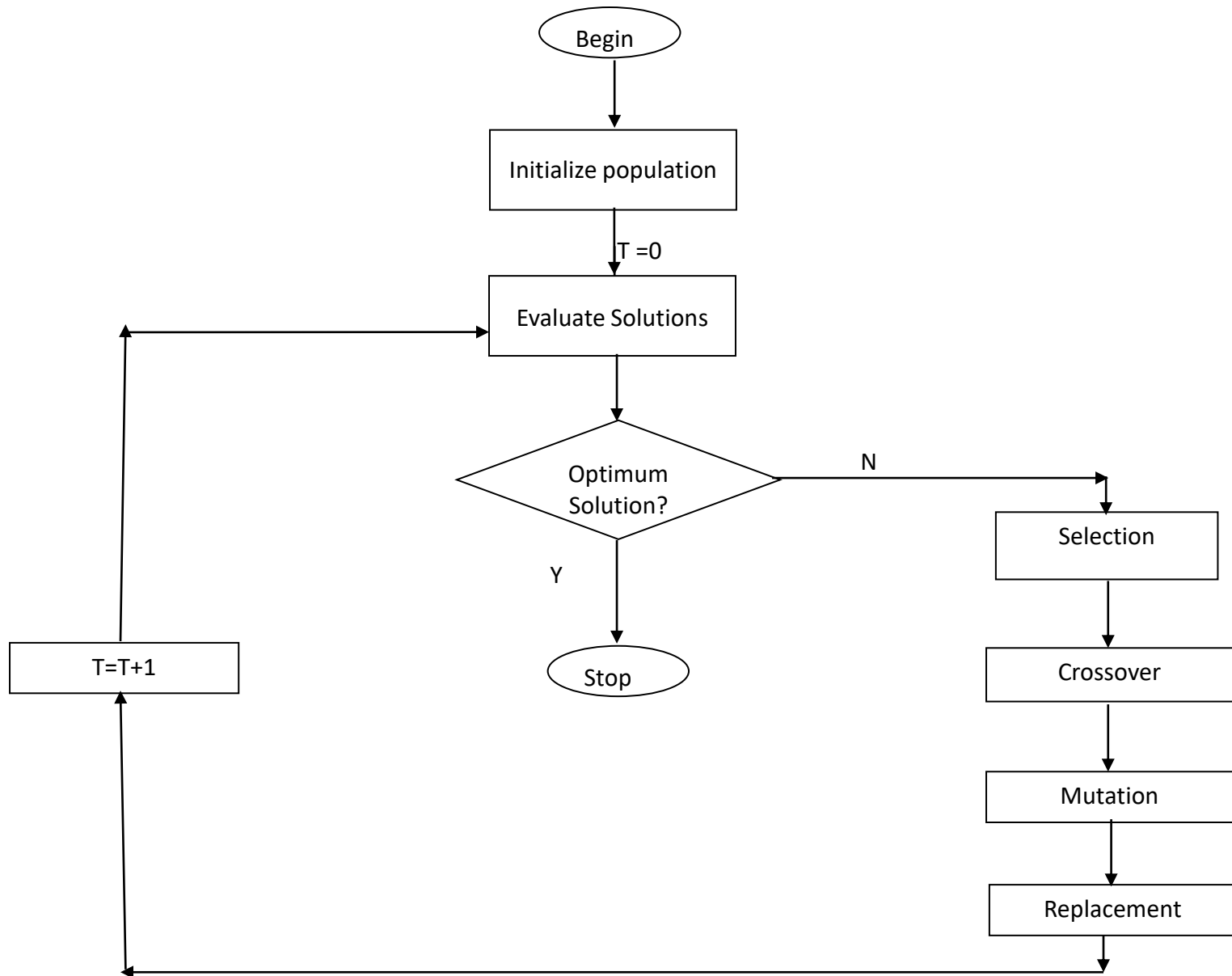
Simple Genetic Algorithm

```
Simple_Genetic_Algorithm()
{
   Initialize the Population;

   Calculate Fitness Function; // Evaluate Individuals …

   While(Fitness Value != Optimal Value) // Continue Evolution …
   {
        Selection; //Natural Selection, Survival Of Fittest …

        Crossover; //Reproduction, Propagate favorable characteristics …

        Mutation;   //Mutation …

        Replacement

        Calculate Fitness Function;
   }
}
```

Nature to Computer Mapping

Nature	Computer
Environment	Problem to be optimized.
Population	Set of candidate solutions.
Individual	One candidate solution to a problem.
Fitness/Survival Degree	Quality of a solution.
Chromosome	Encoding of a Solution to another format *(For example, bit string, floating point array)*.
Gene	Part of the encoded solution.
Reproduction	Crossover
Copying errors	Mutation

Components of a GA

A problem to solve, and …

- Encoding technique *(gene, chromosome)*
- Initialization procedure *(creation)*
- Evaluation function *(environment)*
- Selection *(reproduction, environment)*
- Genetic operators *(crossover, mutation)*
- Parameter settings *(practice and art)*

Many parameters to set

- Any GA implementation needs to decide on a number of parameters: Population size (N), mutation rate (m), crossover rate (c)

- Often these have to be "tuned" based on results obtained - no general theory to deduce good values

- Typical values might be: $N = 50$, $m = 0.05$, $c = 0.9$

Example 1: A Very Simple Example

- Generate a bit string for the decimal number 31.
- Traditional Technique:
  - Start by 00000 till reaching 11111
- GA Steps:
  1. Build an initial population of random individuals (chromosomes or initial solutions)
  2. Evaluate fitness of individuals
  3. Select some individuals to
     - Apply Crossover and Mutation
  4. Build next generation
  5. Go to step 2.

Example 1: GA Step 1

- Build an initial population of _random_ individuals

$$G_0 \quad \boxed{\begin{array}{c} G_{01} \\ 00100 \end{array}} \quad \boxed{\begin{array}{c} G_{02} \\ 10100 \end{array}} \quad \boxed{\begin{array}{c} G_{03} \\ 01011 \end{array}} \quad \boxed{\begin{array}{c} G_{04} \\ 00101 \end{array}}$$

- Generation 0

Example 1: GA Step 2

- Evaluate *fitness* of individuals

$$G_{01} \qquad G_{02} \qquad G_{03} \qquad G_{04}$$

G_0

| 00100 | 10100 | 01011 | 00101 |

$$4 \qquad\qquad 20 \qquad\qquad 11 \qquad\qquad 5$$

- Fitness can be calculated as the decimal value of each bit string

Example 1: GA Step 3
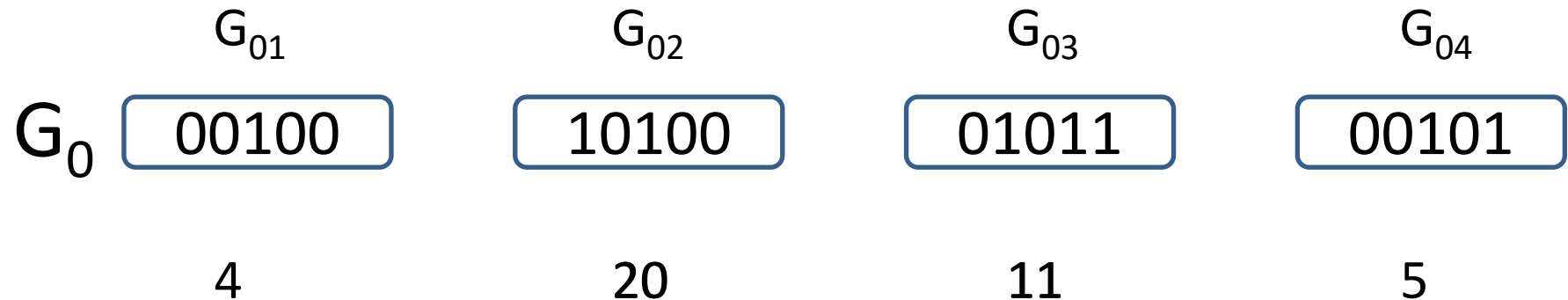
- Select some individuals to apply *crossover* and mutation

	G_{01}	G_{02}	G_{03}	G_{04}
G_0	00100	10100	01011	00101
	4	20	11	5

- Individual with good fitness should be selected for crossover and mutation.

Example 1: GA Step 3

- Select some individuals to apply *crossover* and mutation

G_{01}	G_{02}	G_{03}	G_{04}
00100	**10100**	**01011**	00101
4	20	11	5

G_0

- Individual with good fitness should be selected for crossover and mutation.

Example 1: GA Step 3

- Select some individuals to apply *crossover* and mutation

G_0

G_{01}	G_{02}	G_{03}	G_{04}
00100	**10100**	**01011**	00101
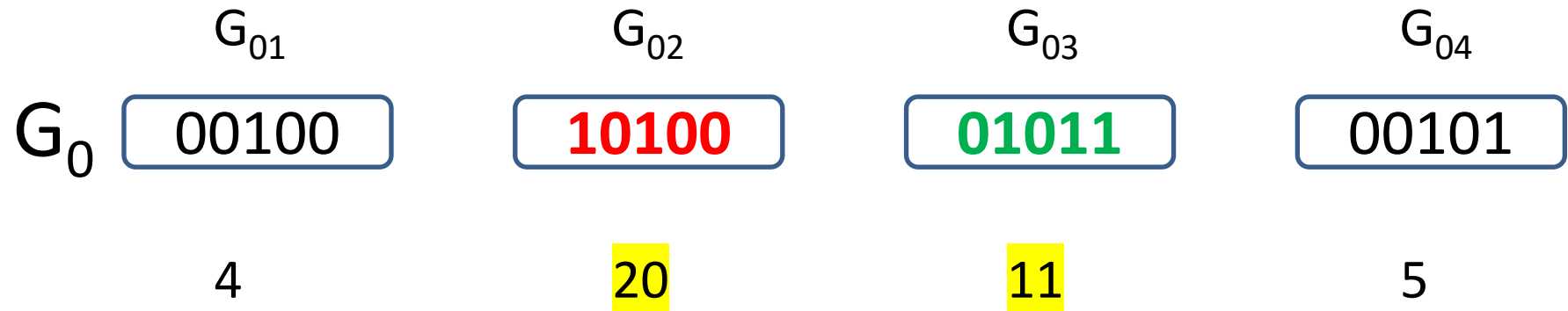
- How to apply crossover between these 2 parents?
  - Single point? Multiple point? Location?
  - It depends on your implementation parameters
  - Crossover may happen or not according to the probability of crossover

Example 1: GA Step 3 – Trial 1

- Select some individuals to apply *crossover* and mutation

G_0

G_{01} 00100

G_{02} **10100**

G_{03} **01011**

G_{04} 00101

G_{11} **10011**

G_{12} **01100**
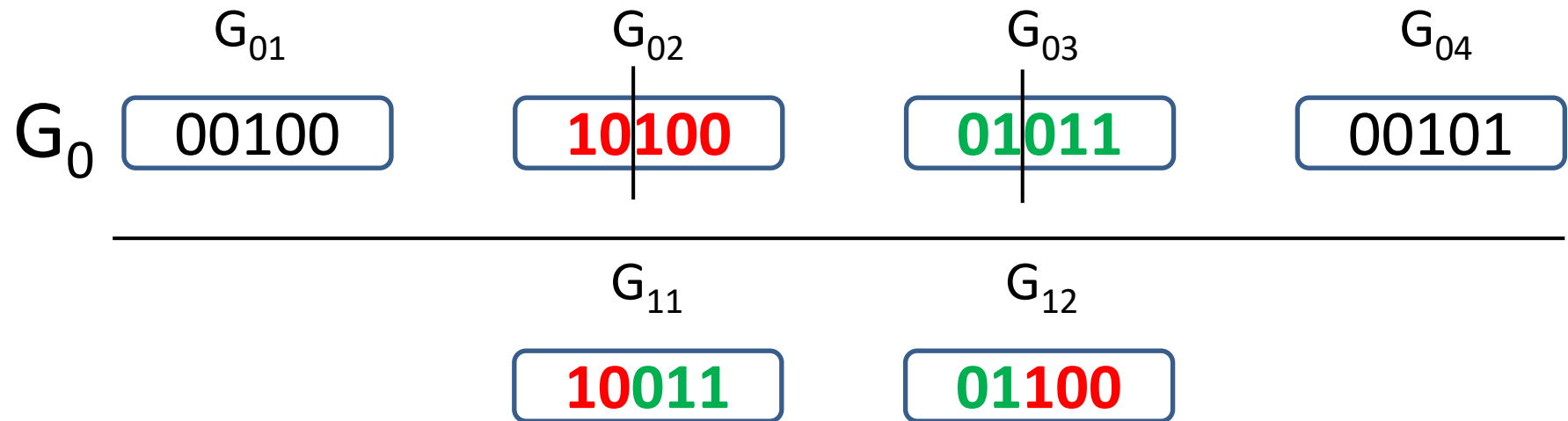
- Single point crossover after 2 bits (Genes)

Example 1: GA Step 3 – Trial 1
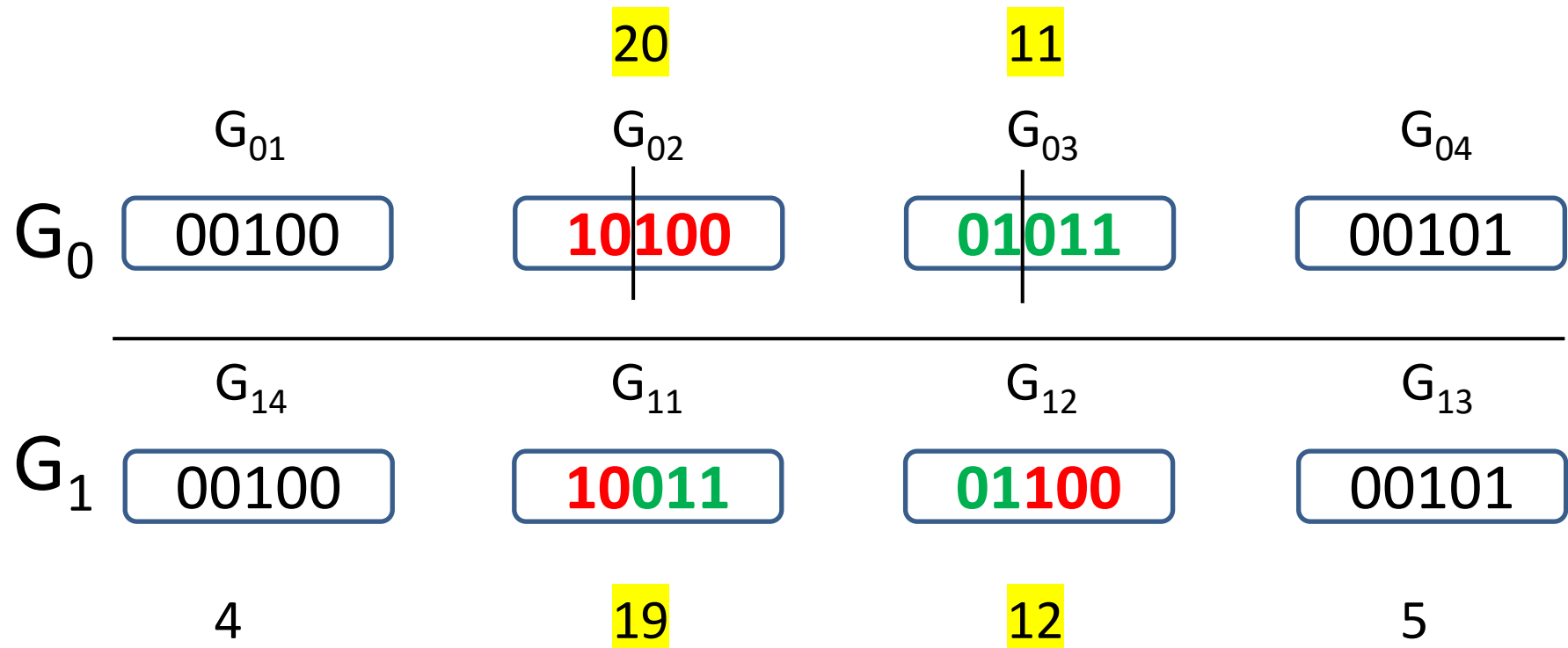
- Select some individuals to apply *crossover* and mutation

	G$_{01}$	20 G$_{02}$	11 G$_{03}$	G$_{04}$
G$_0$	00100	**10100**	**01011**	00101

	G$_{14}$	G$_{11}$	G$_{12}$	G$_{13}$
G$_1$	00100	**10011**	**01100**	00101
	4	19	12	5

Fitness?

Example 1: GA Step 3 – Trial 2

- Select some individuals to apply *crossover* and mutation

G_0

G_{01}	G_{02}	G_{03}	G_{04}
00100	**10100**	**01011**	00101

G_{11}	G_{12}
10111	**01000**
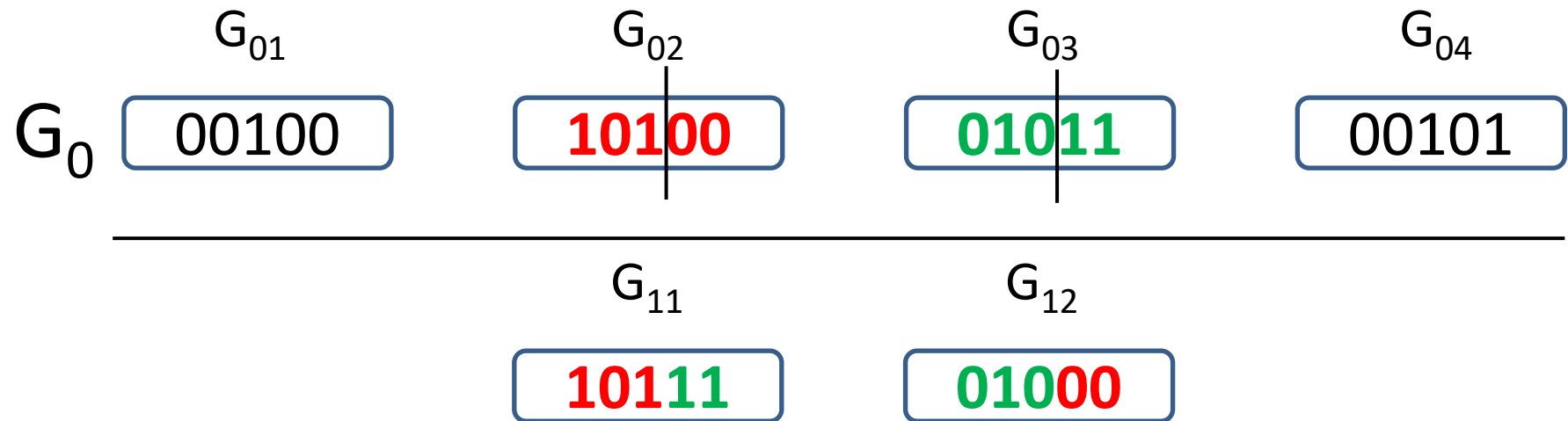
- Single point crossover after 3 bits

Example 1: GA Step 3 – Trial 2
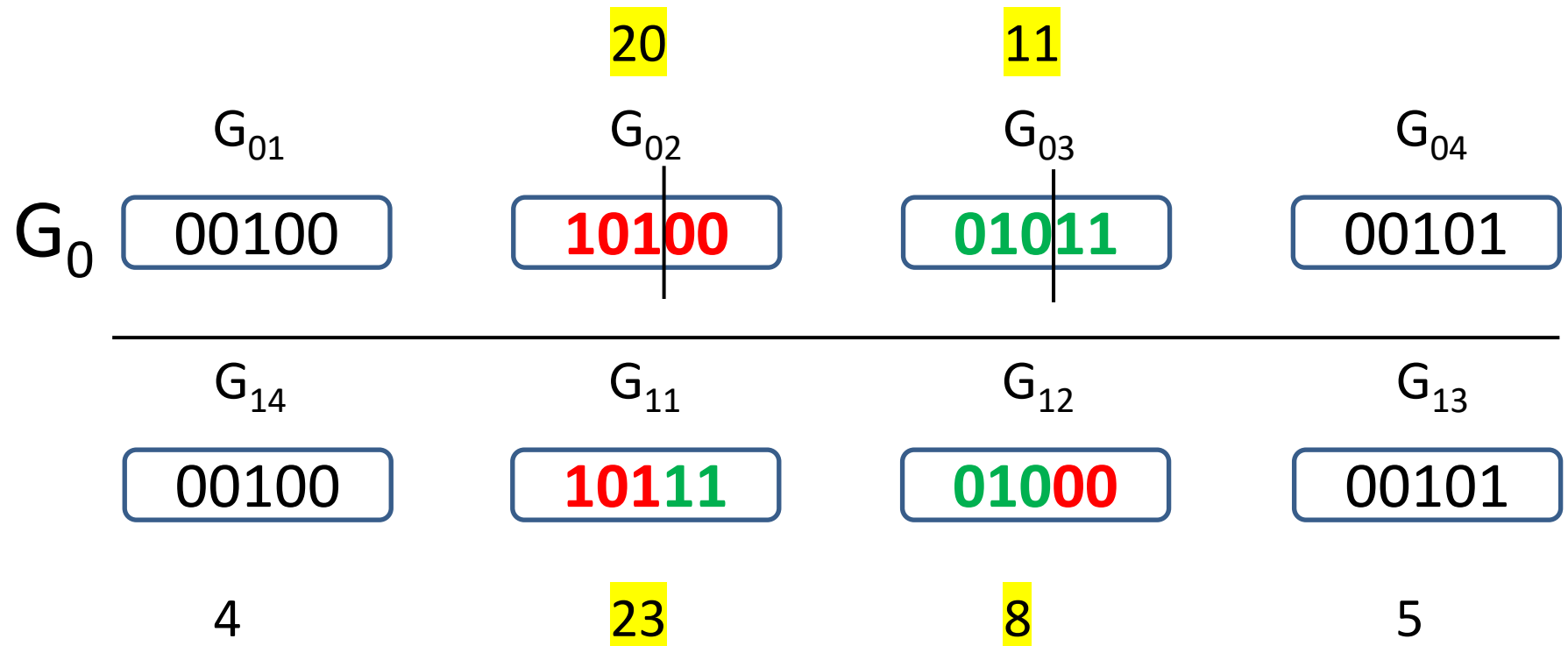
- Select some individuals to apply *crossover* and mutation

G_0

	20		11	
G_{01}	G_{02}		G_{03}	G_{04}
00100	**10100**		**01011**	00101

G_{14}	G_{11}	G_{12}	G_{13}
00100	**10111**	**01000**	00101
4	23	8	5

Fitness?

Example 1: GA Step 3 – Trial 2

- Select some individuals to apply crossover and *mutation*

G_0

20		11	
G_{01}	G_{02}	G_{03}	G_{04}
00100	**10100**	**01011**	00101

G_{14}	G_{11}	G_{12}	G_{13}
00100	**10111**	**01000**	00101
4	23	8	5
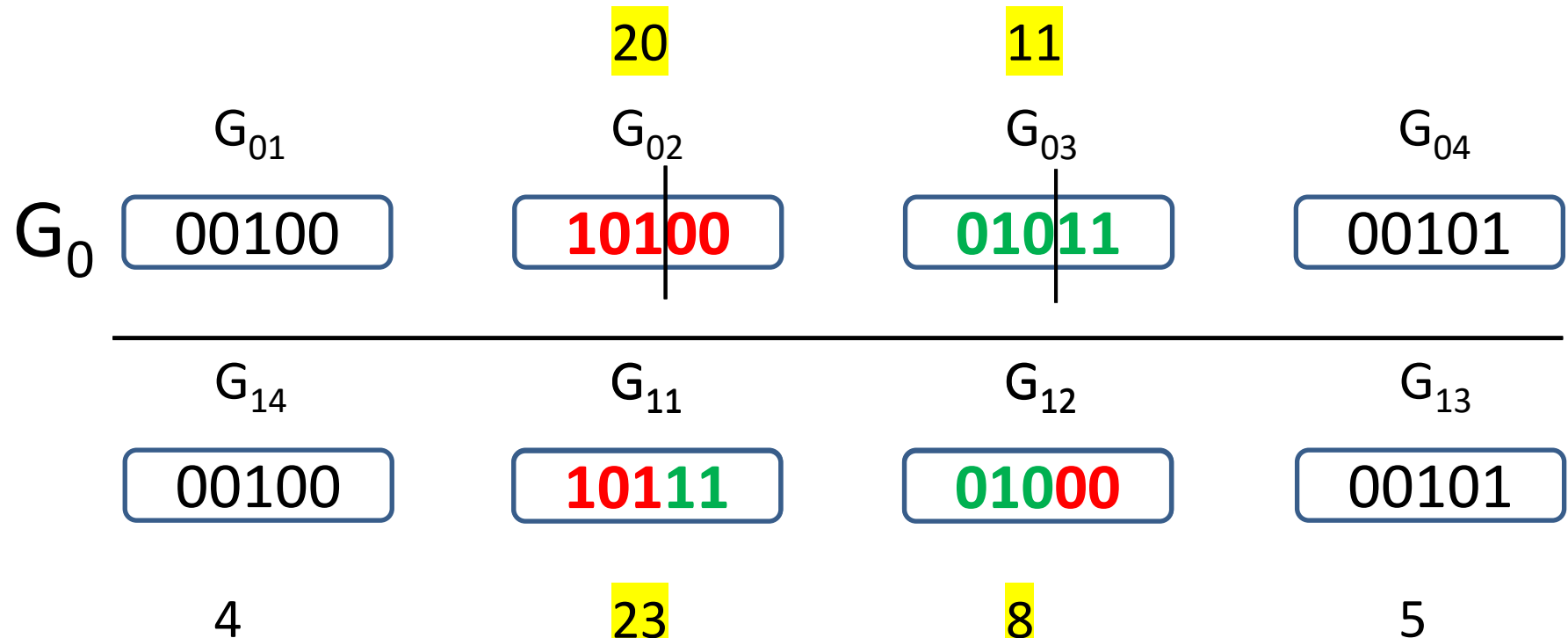
- How to apply a mutation over some individual?
  - For bit strings, and according to the mutation probability of your implementation, one bit might be flipped!

Example 1: GA Step 3 – Trial 2

- Select some individuals to apply crossover and *mutation*

20   11

G_{01} G_{02} G_{03} G_{04}

G_0 00100   **10100**   **01011**   00101

G_{13} G_{11} G_{12} G_{14}

G_1 00100   **11111**   **01000**   00101

4   **31**   8   5
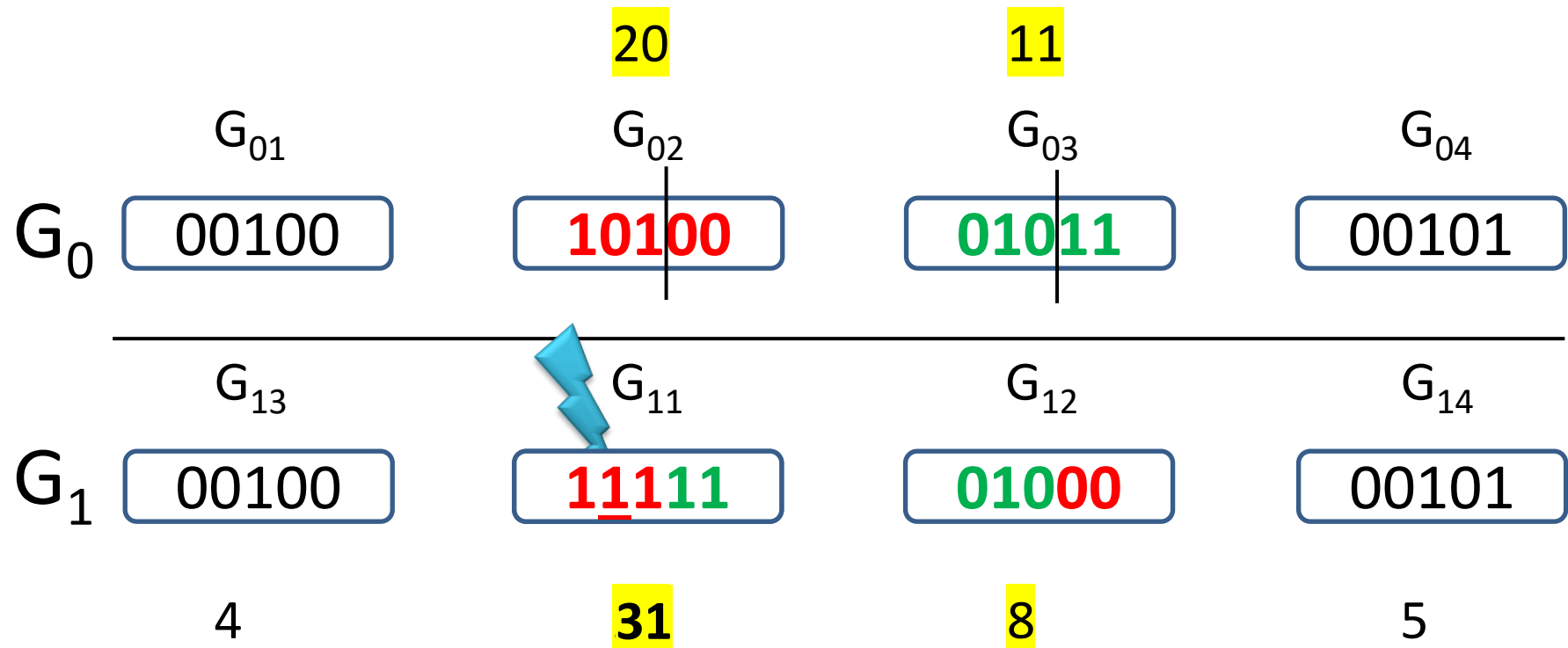
OUR OBJECTIVE!

Example 1: GA Step 3 – Trial 3

- Select some individuals to apply crossover and mutation

G_0

G_{01}	G_{02}	G_{03}	G_{04}
00100	**10100**	**01011**	00101

- Applying multiple point crossover!
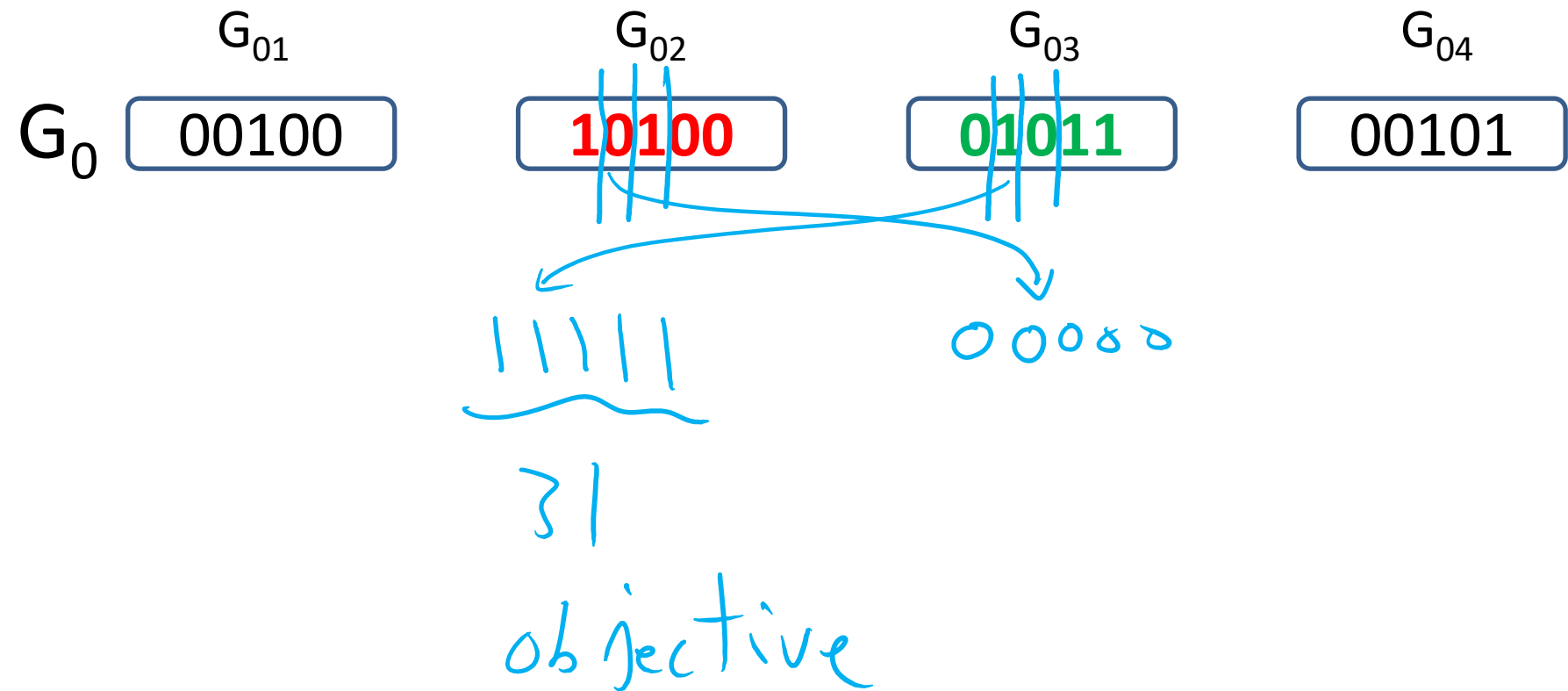
G_0

G_{01} 00100 G_{02} **10100** G_{03} **01011** G_{04} 00101

G_{01} G_{02} G_{03} G_{04}

G_0 00100   **10100**   **01011**   00101

||||||

$\underbrace{}$

31

objective

00000

Example 1: GA Step 3 – Trial 4

- Select **<u>ALL</u>** individuals to apply *crossover* and mutation

G_0

G_{01}
00100
4

G_{02}
10100
20

G_{03}
01011
11

G_{04}
00101
5

Crossover

G_{11}
00011
3

G_{12}
01100
12

Example 1: GA Step 3 – Trial 4

- Select **ALL** individuals to apply *crossover* and mutation

G_0

G_{01}	G_{02}	G_{03}	G_{04}
00100	10100	01011	00101
4	20	11	5

Crossover

G_{11}	G_{13}	G_{12}	G_{14}
00011	10101	01100	00100
3	21	12	4

Mutation

G_{11}	G_{13}	G_{12}	G_{14}
00011	11101	01100	00110
3	21→29	12	4→6

Example 1: GA Step 4

- Build next generation .. Full Replacement or Mix?

G_0

G_{01}	G_{02}	G_{03}	G_{04}
00100	10100	01011	00101
4	20	11	5

Crossover

G_{11}	G_{13}	G_{12}	G_{14}
00011	10101	01100	00100
3	21	12	4

Mutation

G_{11}	G_{13}	G_{12}	G_{14}
00011	11101	01100	00110
3	29	12	6
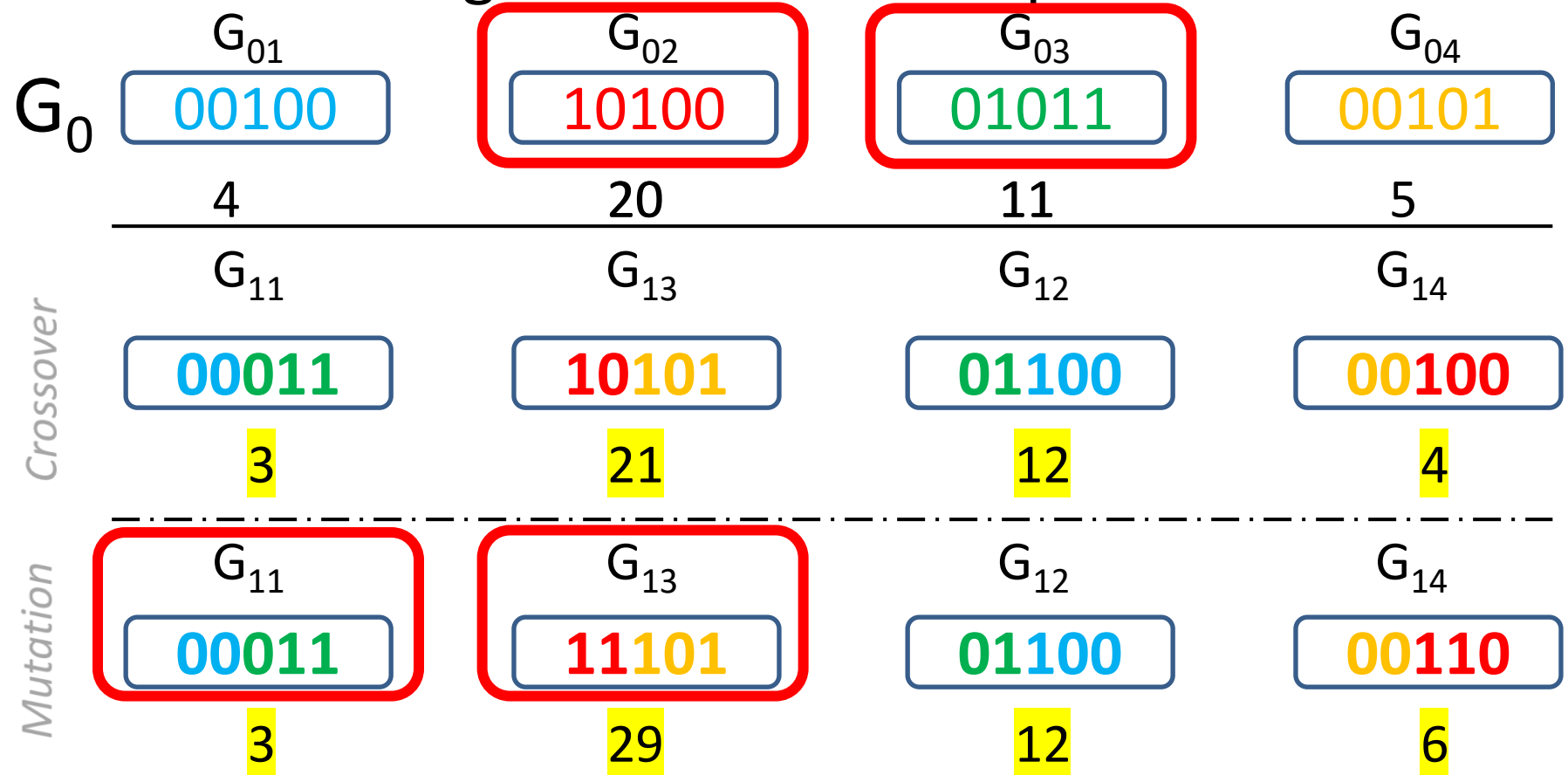
Example 1: GA Step 4

- Build next generation .. Mixing

G_1

G_{02}	G_{03}	G_{11}	G_{13}
10100	01011	00011	11101
20	11	3	29

Crossover

G_{02}	G_{03}	G_{21}	G_{22}
10100	01011	00001	11111
20	11	1	31

Mutation

G_{02}	G_{03}	G_{21}	G_{22}
10100	11011	00000	11111
20	27	0	31

OUR OBJECTIVE!
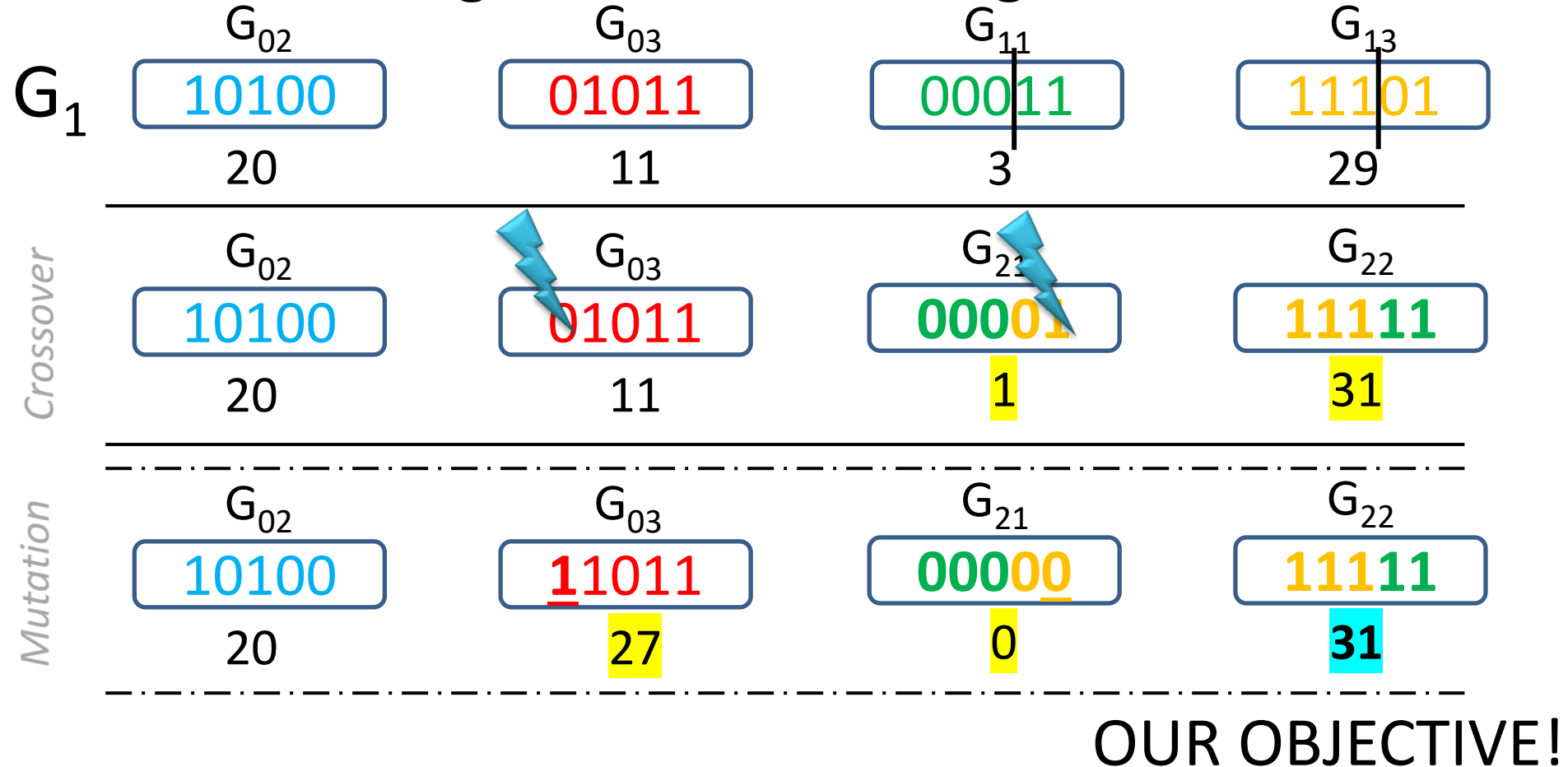
Does it really cost one iteration?

- Real world problems are not that easy.
- You could have:
  - Hundreds of individuals (chromosomes)
  - Huge search space
- You might perform thousands of iterations (have thousands of generations) before reaching your objective!

How do you encode a solution?

- Obviously this depends on the **problem**!

- GA's *often* encode solutions as fixed length "*bitstrings*" (e.g. 101110, 111111, 000101)

- Sometimes, your solutions (chromosomes) will be used in their original format (*floating point numbers* or *alphabetical strings*)

Why does crossover work?

- The idea is that crossover preserves "good bits" from different parents, combining them to produce better solutions

- The fitness may be improved or get worst after mutation

- A good scheme would therefore try to preserve "good bits" during crossover and mutation

Which problems can GAs solve?

- The solution can be encoded.

- Each gene represents some aspect of the proposed solution to the problem (the solution can be decomposed into smaller parts)

- The ability to "test" any solution and get a "score" indicating how "good" that solution is (**Fitness**).