香港浸會大學
HONG KONG BAPTIST UNIVERSITY
DEPARTMENT OF
COMPUTER SCIENCE
計算機科學系

COMP7015 Artificial Intelligence

Lecture 12: Genetic Algorithm & Course Review

Instructor: Dr. Kejing Yin

December 1, 2022

Course Project

- Submit early to avoid last-minute accidence!
- The submission box closes at due automatically. Late submissions are not allowed unless prior consent is obtained.
- Present your face in the video presentation.

> 8. In the **Presentation Video**, you are required to display the presenter's faces clearly to allow the lecturers to identify your identity.

Agenda

- Genetic Algorithm

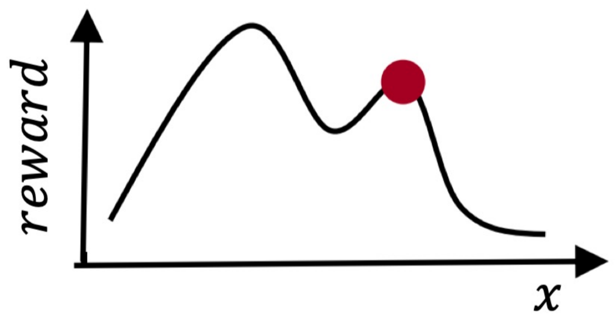
- Course Review
- Tips & Suggestions for preparing for the final

- Review of Assignments

Genetic Algorithm

Genetic Algorithm

- Why using genetic algorithm?



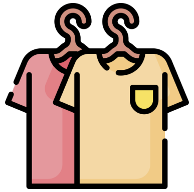
  - Avoid getting trapped in (bad) local optima
  - To handle discrete design variables $x \epsilon \{a, b, c, d, e\}$, $x \epsilon \{true, false\}$,
  - However, genetic algorithms do not guarantee a global optimum.

Motivating Problem: Discrete Optimization

- **The Knapsack Problem**

Goal: maximize the value of items but cannot exceed the weight limit.

Maximum weight: 3kg

Value:	750	1100	180	90	3	80	120
Weight:	0.4	1.2	0.3	0.6	1.0	0.2	0.6

Motivating Problem: Discrete Optimization

- **Solution 1: Brute-force** (List out all possible combinations)
  - For each item, use a binary variable to indicate its status.
  - Complexity: 2^N where N is the number of items.

Number of items	Number of combinations	Seconds
5	32	0.00000192
10	1,024	0.00006144
20	1,048,576	0.06291456
30	1,073,741,824	64.42450944
40	$\approx 10^{12}$	~65970.6976666
50	$\approx 10^{15}$	~67553994.4106
60	$\approx 10^{18}$	~69175290276.4

(2193 years)

Motivating Problem: Discrete Optimization

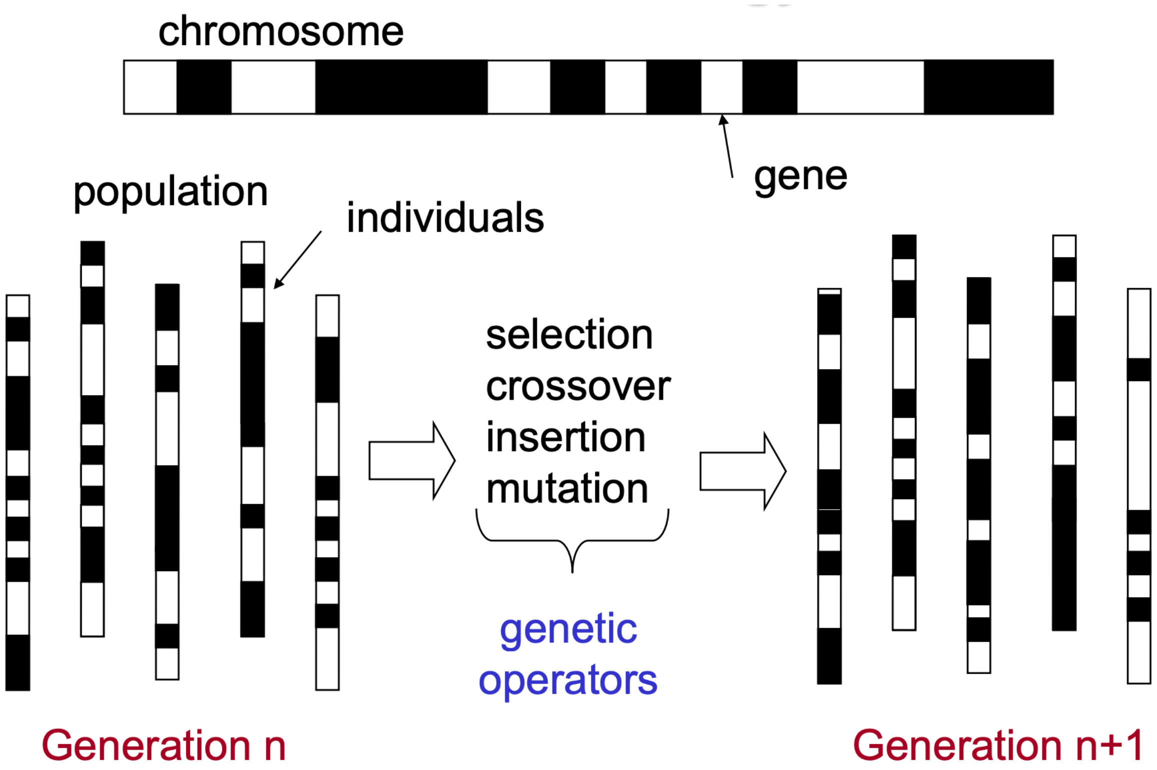
- **Solution 2: Genetic Algorithm**

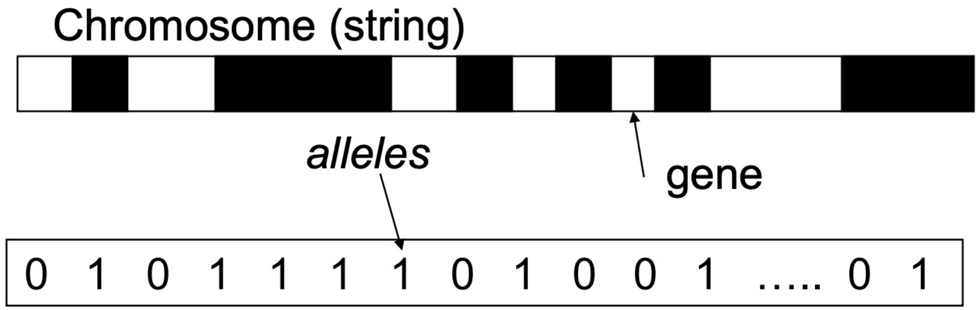
Number of items	Time using Brute-force	Time using Genetic Algorithm	Accuracy of Genetic Algorithm
5	0.00000192	0.000460147	100.00%
10	0.00006144	0.001574755	100.00%
20	0.06291456	0.003477812	100.00%
30	64.42450944	0.007086039	100.00%
40	~65970.6976666	0.0159061	100.00%
50	~67553994.4106	0.0206821	97.57%
60	~69175290276.4	0.0238941	94.32%

Idea of Genetic Algorithm

- Genetic algorithm (GA) belongs to a larger class of “Evolutionary algorithms”.
- It is inspired by the process of natural selection.
- Natural Selection is a very successful organizing principle for optimizing individuals and populations of individuals.
- It is argued that if we can mimic natural selection, then we will be able to optimize more successfully.
- Only the fittest survive – define a fitness function.

Terminologies

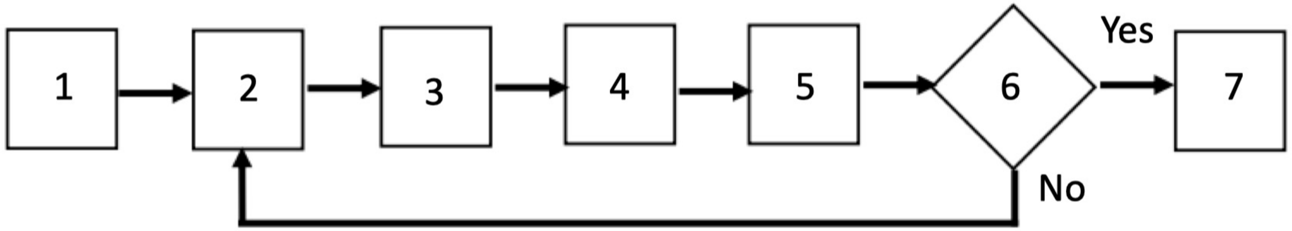




Each chromosome represents a solution, often using strings of 0's and 1's. Each bit typically corresponds to a gene. This is called binary encoding.

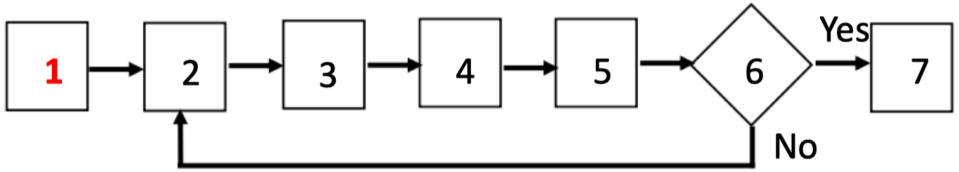
https://ocw.mit.edu/courses/ids-338j-multidisciplinary-system-design-optimization-spring-2010/resources/mitesd_77s10_lec11/

Framework of GA

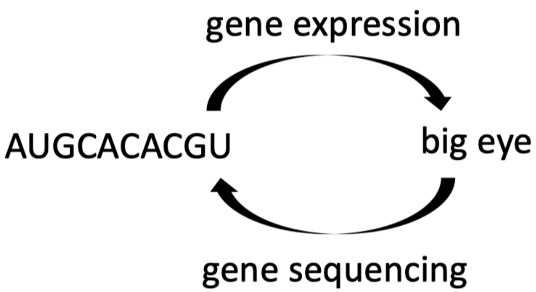


1. Initialize Population
2. Select individual for mating
3. Mate individuals and produce children
4. Mutate children
5. Insert children into population
6. Check whether termination conditions are met
7. Output

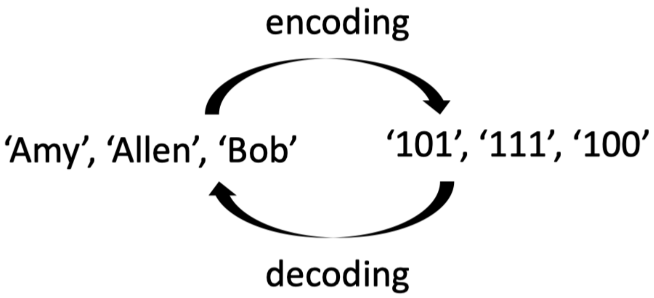
Framework of GA



1. Initialize Population



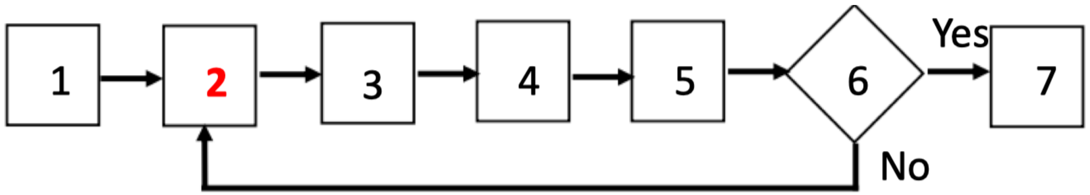
Biology



Design

(binary coding)

Framework of GA



2. Select individual for mating

a) Choosing the right fitness function

b) Evaluating genes with fitness functions

c) Selecting genes with higher value, e.g., top k

Selection by Ranking

- Goal is to select parents for crossover
- Should create a bias towards more fitness
- Must preserve diversity in the population

(1) Selection according to RANKING

Example: Let $D = \sum_{j \in P} 1/j$

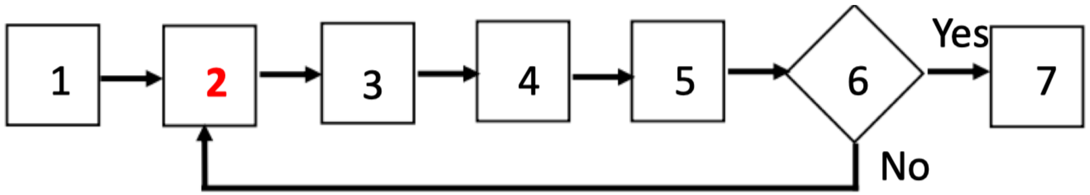
select the k[th] most fit member of a population

to be a parent with probability $P_k = \left(\frac{1}{k}\right) D^{-1}$

⇨ Better ranking has a higher probability of being chosen, e.g. 1st $\propto 1$, 2nd $\propto 1/2$, 3rd $\propto 1/3$ ...

https://ocw.mit.edu/courses/ids-338j-multidisciplinary-system-design-optimization-spring-2010/resources/mitesd_77s10_lec11/

Framework of GA



2. Select individual for mating

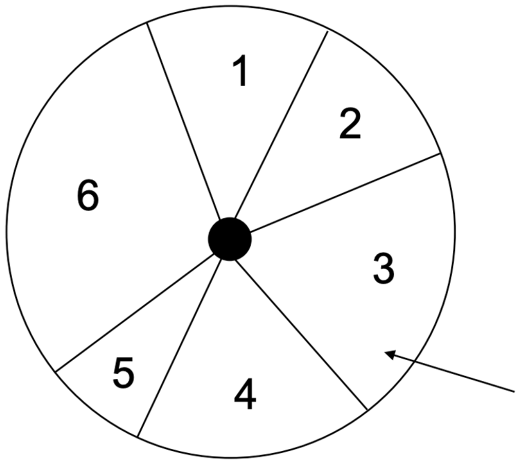
a) Choosing the right fitness function

b) Evaluating genes with fitness functions

c) Selecting genes with higher value, e.g., top k

Roulette Wheel Selection

Roulette Wheel Selection



Probabilistically select individuals based on some measure of their performance.

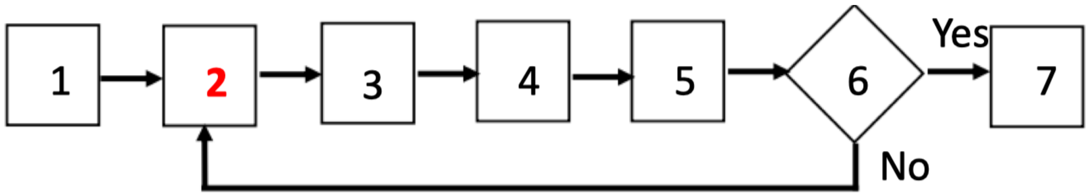
Sum Sum of individual's selection probabilities

3rd individual in current population mapped to interval [0,*Sum*]

Selection: generate random number in [0,*Sum*]
Repeat process until desired # of individuals selected
Basically: stochastic sampling with replacement (SSR)

https://ocw.mit.edu/courses/ids-338j-multidisciplinary-system-design-optimization-spring-2010/resources/mitesd_77s10_lec11/

Framework of GA



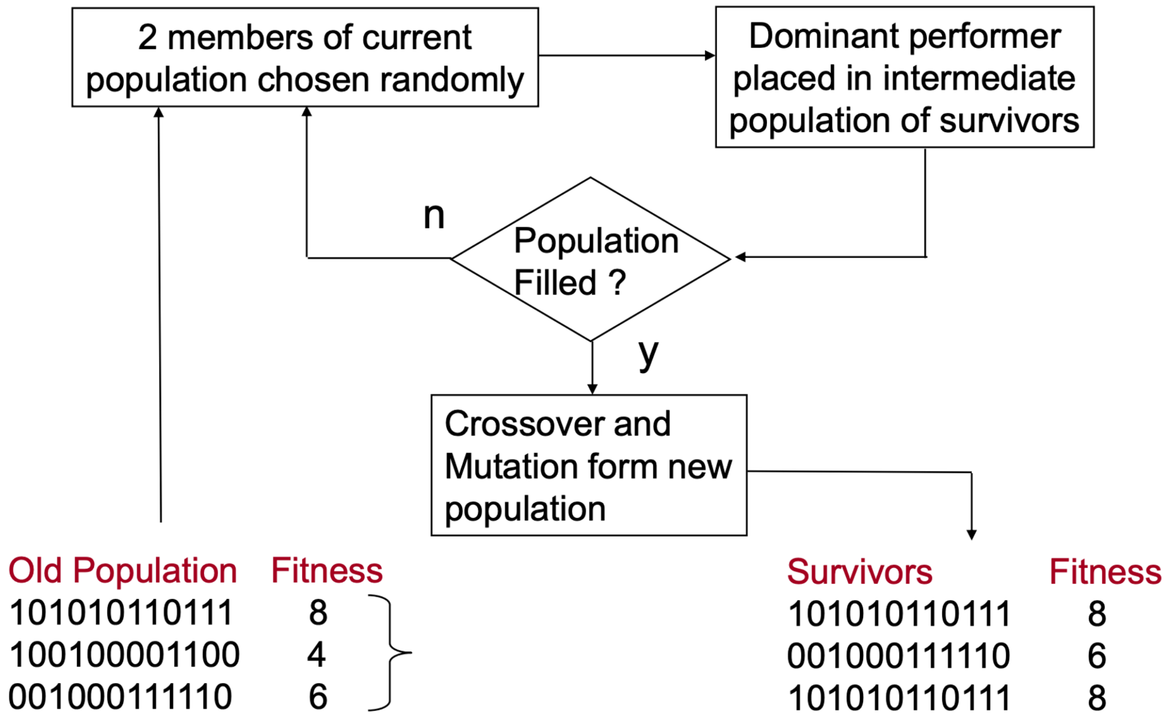
2. Select individual for mating

   a) Choosing the right fitness function

   b) Evaluating genes with fitness functions

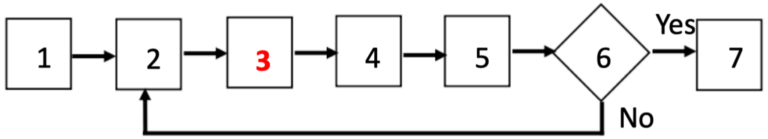
   c) Selecting genes with higher value, e.g., top k

Tournament Selection

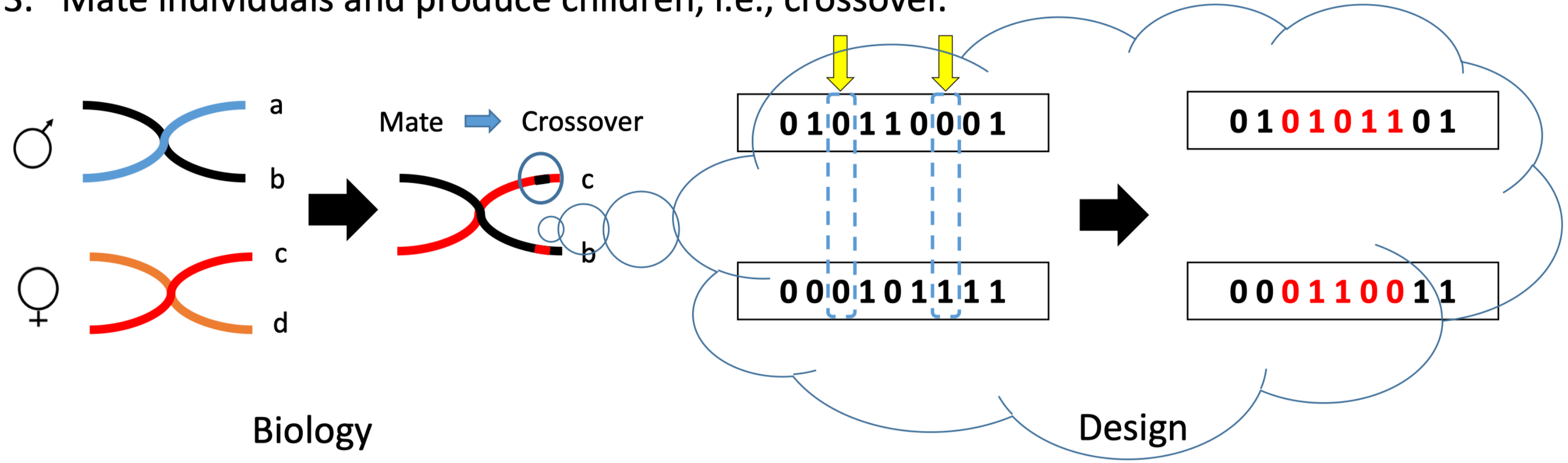


https://ocw.mit.edu/courses/ids-338j-multidisciplinary-system-design-optimization-spring-2010/resources/mitesd_77s10_lec11/

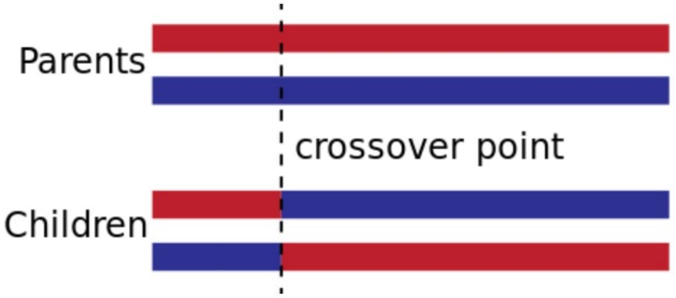
Framework of GA



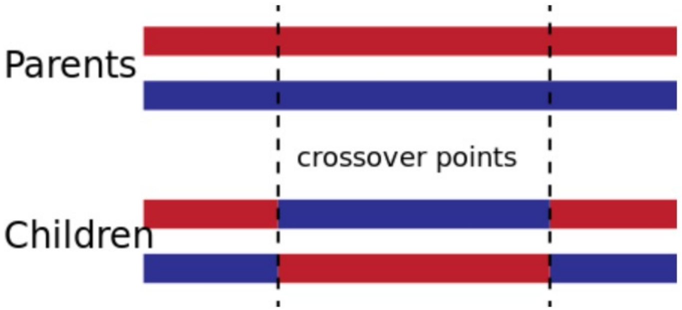
3. Mate individuals and produce children, i.e., crossover.



Two-point Crossover

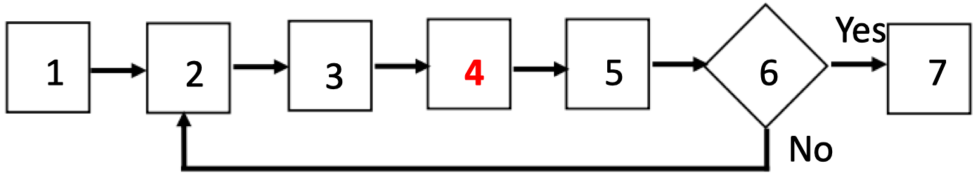


One-point crossover

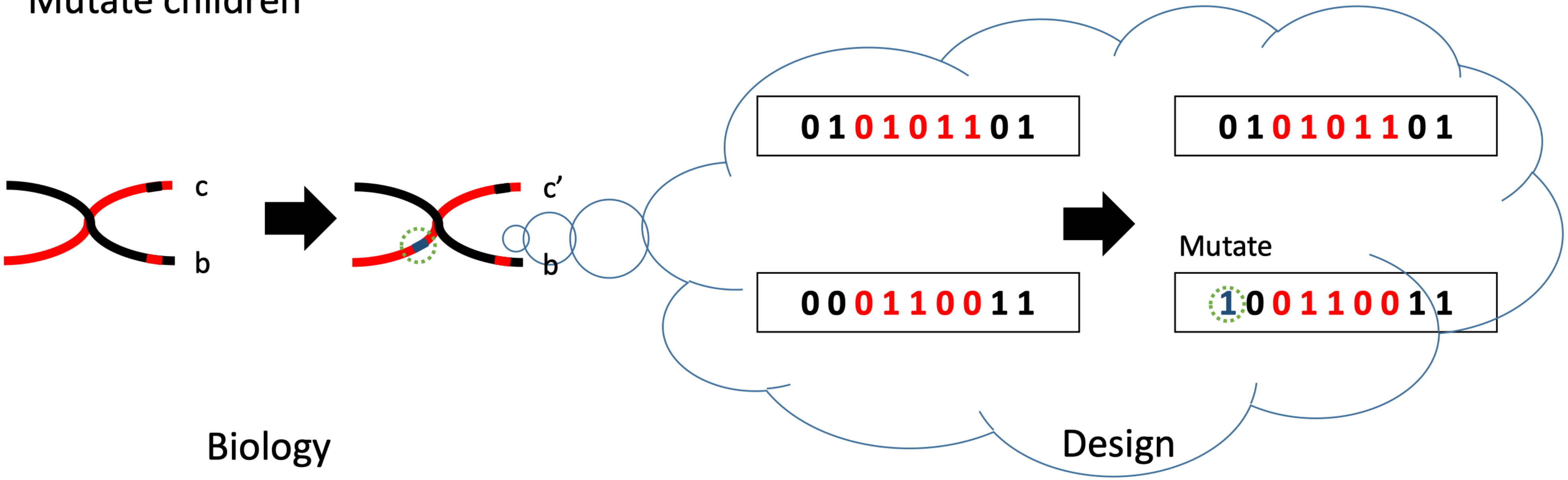


Two-point crossover

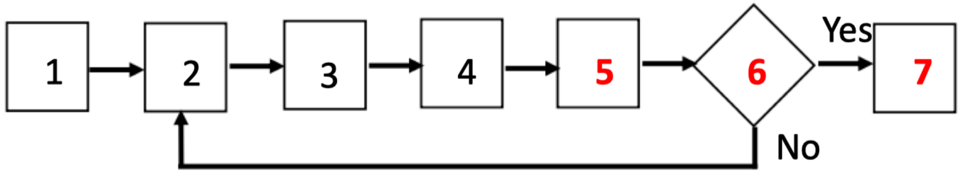
Framework of GA



4. Mutate children



Framework of GA



5. Insert children into population

6. Check whether termination conditions are met

   a) Iteration:

      The maximum number of generations completed.

   b) Design variable:

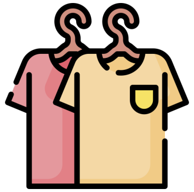
      All genes become almost the same.

   c) Fitness:

      The improvement of the fitness (reward) is marginal.

7. Output the design variables

An example of GA: The Knapsack Problem

							
Value:	750	1100	180	90	3	80	120
Weight:	0.4	1.2	0.3	0.6	1.0	0.2	0.6
binary coding	1	0	0	0	1	1	0

An example of GA: The Knapsack Problem

Step 1: Initialize population

Value:	750	1100	180	90	3	80	120
Weight:	0.4	1.2	0.3	0.6	1.0	0.2	0.6

Generation 0
(just random)

1000110 1000001 1100001 1111001 0110100

0101011 0100010 1010110 0010011 1101001

An example of GA: The Knapsack Problem

Step 2: Select Individuals for matting

Value:	750	1100	180	90	3	80	120
Weight:	0.4	1.2	0.3	0.6	1.0	0.2	0.6

Generation 0 (just random)

1000110 1000001 1100001 1111001 0110100 0101011 0100010 1010110 0010011 1101001

Fitness function: sum of the values

$$\text{Fitness} = \begin{cases} 750\ b_1 + 1100\ b_2 + 180\ b_3 + 90\ b_4 + 3\ b_5 + 80\ b_6 + 120\ b_7 & \text{if weight} \leq 3 \\ 0 & \text{if weight} > 3 \end{cases}$$

An example of GA: The Knapsack Problem

Step 2: Select Individuals for matting

Value:	750	1100	180	90	3	80	120
Weight:	0.4	1.2	0.3	0.6	1.0	0.2	0.6

Generation 0 (just random)

1000110	1000001	1100001	1111001	0110100	0101011	0100010	1010110	0010011	1101001

Fitness function: sum of the values

833	**870**	**1970**	**0**	**1283**	**1390**	**1180**	**1180**	**380**	**2060**

An example of GA: The Knapsack Problem

Step 2: Select Individuals for matting (e.g., Roulette Wheel selection method)

Suppose the two get selected

Individual	Fitness Function	Selection Probability
1000110	833	0.07
1000001	870	0.08
1100001	1970	0.18
1111001	0	0
0110100	1283	0.12
0101011	1390	0.12
0100010	1180	0.11
1010110	1180	0.11
0010011	380	0.03
1101001	2060	0.18
	Sum = 11146	Sum = 1

An example of GA: The Knapsack Problem

Step 3: Mate individuals and produce children (cross-over)

100 00 01	100 10 01
010 10 11	010 00 11
parents	*Children (offsprings)*

An example of GA: The Knapsack Problem

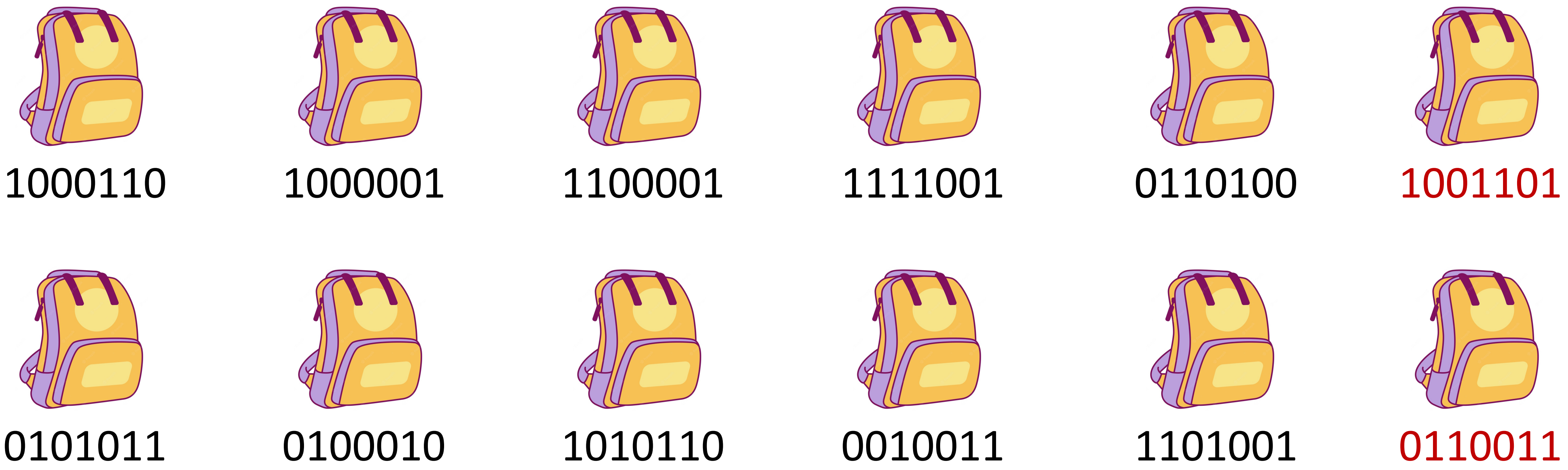
Step 4: Mutate children

100 10 01	→	100 11 01
010 00 11		011 00 11
children		*mutation*

An example of GA: The Knapsack Problem

Step 5: Insert children into population

Generation 1



An example of GA: The Knapsack Problem

Step 6: Check Termination

If one of the following conditions is met, terminate:

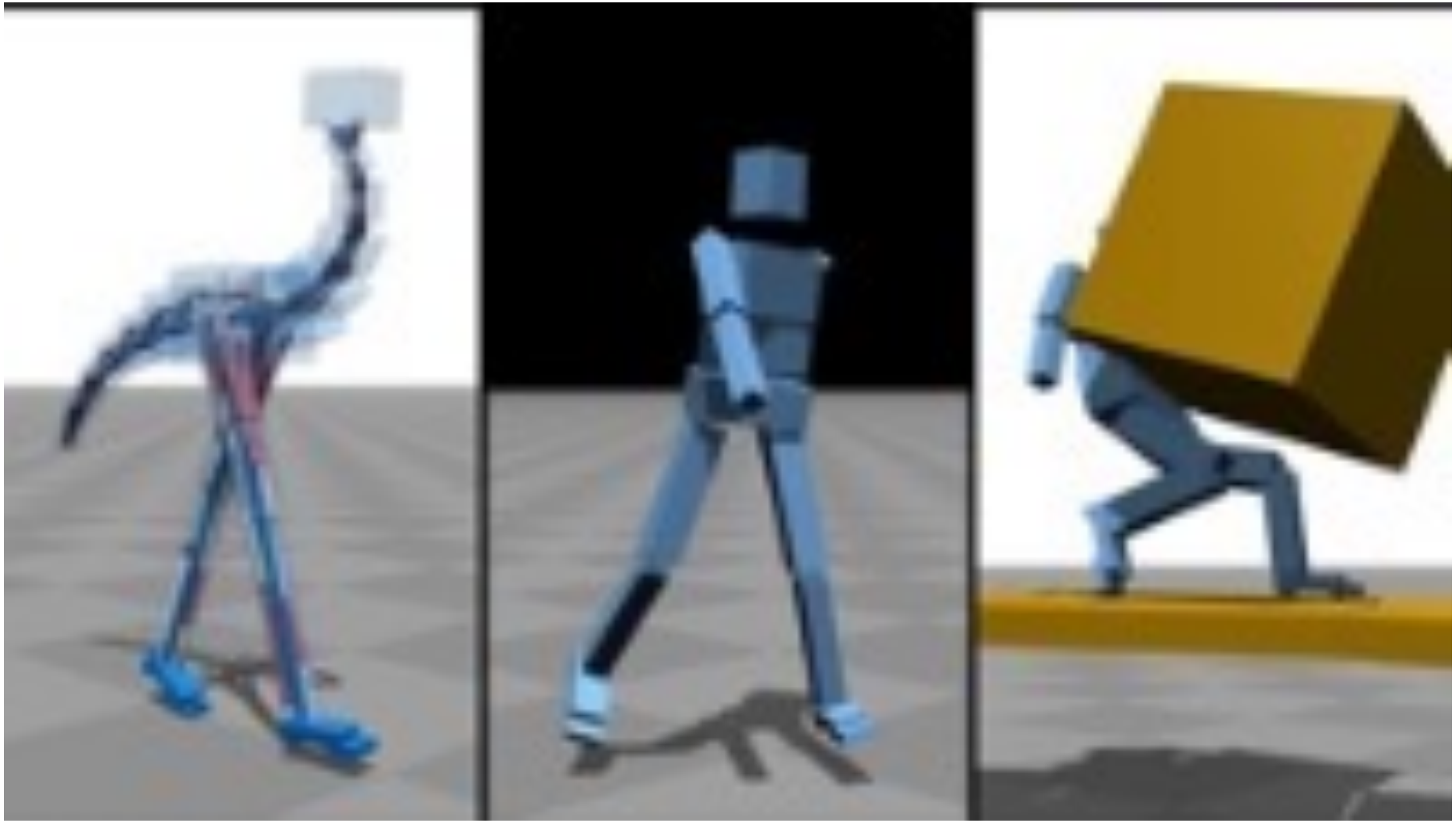
- The maximum number of generations completed.
- All genes become almost the same.
- The improvement of the fitness is marginal.

Otherwise, go back to step 2 and iterate.

The individual with maximum fitness function is the result.

https://www.youtube.com/watch?v=pgaEE27nsQw&t=1s

A Demo: Flexible Muscle-Based Locomotion for Bipedal Creatures



https://www.youtube.com/watch?v=pgaEE27nsQw

Suggested Readings

- David E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, 1989