

Computational Intelligence

(Part I, Fall 2021)

Instructor: Dr. Yanan Sun

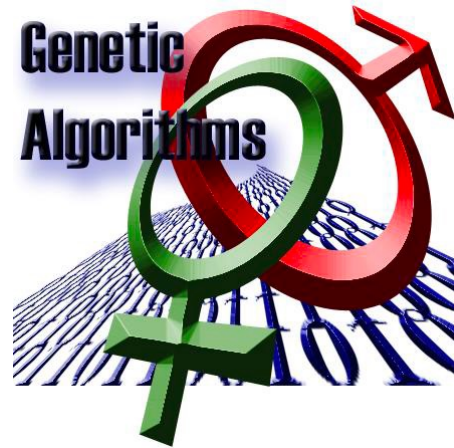
College of Computer Science

ysun@scu.edu.cn

<https://yn-sun.github.io>

GENETIC ALGORITHM

.....



Introduction

- Genetic algorithms are inspired by Darwin's theory of evolution (i.e., survival of the fittest);

“The new breeds of classes of living things come into existence through the process of reproduction, crossover, and mutation among existing organisms.”

- The above concept in the theory of evolution has been translated into an algorithm to search for solutions to problems in a more ‘natural’ way.

History

- Genetic algorithms are originated from the studies of cellular automata, conducted by John Holland and his colleagues at the University of Michigan.
- Until the early 1980s, the research in genetic algorithms was mainly theoretical, with few real applications.
- From the early 1980s the community of genetic algorithms has experienced an abundance of applications which spread across a large range of disciplines. Each and every additional application gave a new perspective to the theory.

- In the process of improving performance as much as possible via tuning and specializing the genetic algorithm operators, new and important findings regarding the generality, robustness, and applicability of genetic algorithms became available.
- Following the last couple of years of furious development of genetic algorithms in the sciences, engineering and the business world, these algorithms in various guises have now been successfully applied to optimization problems, scheduling, data fitting and clustering, trend spotting and path finding.

Biological Background

- **Chromosome**: are strings of DNA (genes) and serve as a model for the whole organism
- Each **gene** encodes a particular protein. Basically, it can be said that each gene encodes a trait, for example color of eyes.
- Possible settings for a trait (e.g. blue, brown) are called **alleles**.
- Each gene has its own position in the chromosome. This position is called **locus**.

- Complete set of genetic material (all chromosomes) is called **genome**.
- Particular set of genes in genome is called **genotype**.
- The genotype is the after birth base for the organism's **phenotype**, its physical and mental characteristics, such as eye color, intelligence, etc.

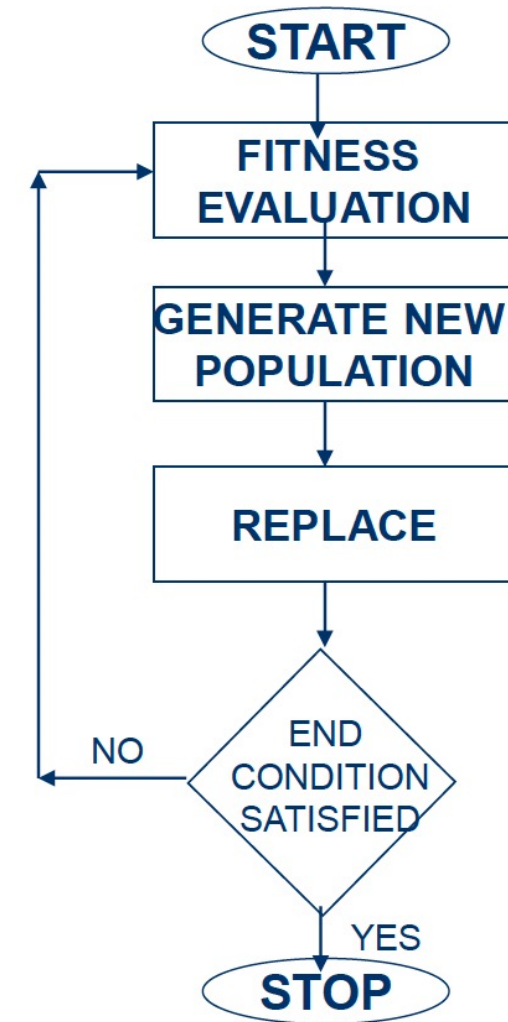
- During reproduction, **recombination**(or **crossover**) first occurs. Genes from parents combine to form a whole new chromosome.
- The newly created offspring can then be mutated. **Mutation** means that the elements of DNA are a bit changed. This changes are mainly caused by errors in copying genes from parents.
- The **fitness** of an organism is measured by success of the organism in its life (survival).

Biological Metaphor

1. Algorithm begins with a set of solutions (represented by **chromosomes**) called **population**.
2. Solutions from one population are evaluated for their goodness (fitness) and used to form a new population, (**reproduction**). This is motivated by a hope, that the new population will be better than the old one.
3. Solutions which are then selected to form new solutions. (**Offspring**) are selected according to their fitness -the more suitable they are the more chances they have to reproduce.
4. This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

Flowchart

1. Generate random population of N chromosomes (feasible solutions for the problem)
2. Evaluate the fitness $f(x)$ of each chromosome x in the population
3. Create a new population by repeating following steps until the new population is complete by means of selection and crossover or mutation
4. Replace unfit individuals in old population by new off springs
5. If the end condition is satisfied, stop, and return the best solution in current population
6. else Go to step 2



An Illustration Example

- How a line may be fit through a given data set using a genetic algorithm?
- Consider a known model structure, a linear fit $y = C_1x + C_2$
 - Encode C_1 and C_2 in 6-bit string each ($L = 6$)
 - E.g., 0 0 0 1 1 1 0 1 0 1 0 0 (genotype), $C_1 = 7$ $C_2 = 20$, $y = 7x + 20$ (phenotype)
 - Initial population = 4
 - Assume the minimum of C_1 and C_2 are $C_{\min} = -2$; and the maximum of C_1 and C_2 are $C_{\max} = 5$.
 - Evaluate the performance from the given data set
 - Cut-off value = 0.8 (relative fitness)

- Mapping from genotype to phenotype

$$- C_i = C_{\min,i} + b / (2^L - 1) (C_{\max,i} - C_{\min,i})$$

- Selection: strings with better fitness values receive corresponding more copies in the new generation (i.e., strings with lower fitness values are eliminated) (i.e., fitness proportionate selection)
- Crossover: strings are able to mix and match their desirable qualities in a random fashion (i.e., one-point crossover)
- Mutation: helps to increase the searching space, allowing a vital bit of information to vary (from 1 to 0 or from 0 to 1) (i.e., bit-flip mutation). Mutation takes place very rarely (e.g., 0.005/bit/generation)

Design Parameters

- Design a representation
- Decide how to initialize a population
- Design a mechanism to map the phenotype to genotype and vice versa
- Design a way of evaluating an individual
- Design suitable mutation operators
- Design suitable crossover operators
- Decide how to manage the population
- Decide how to select parents for crossover
- Decide how to select individuals to be replaced
- Decide when to stop the algorithm

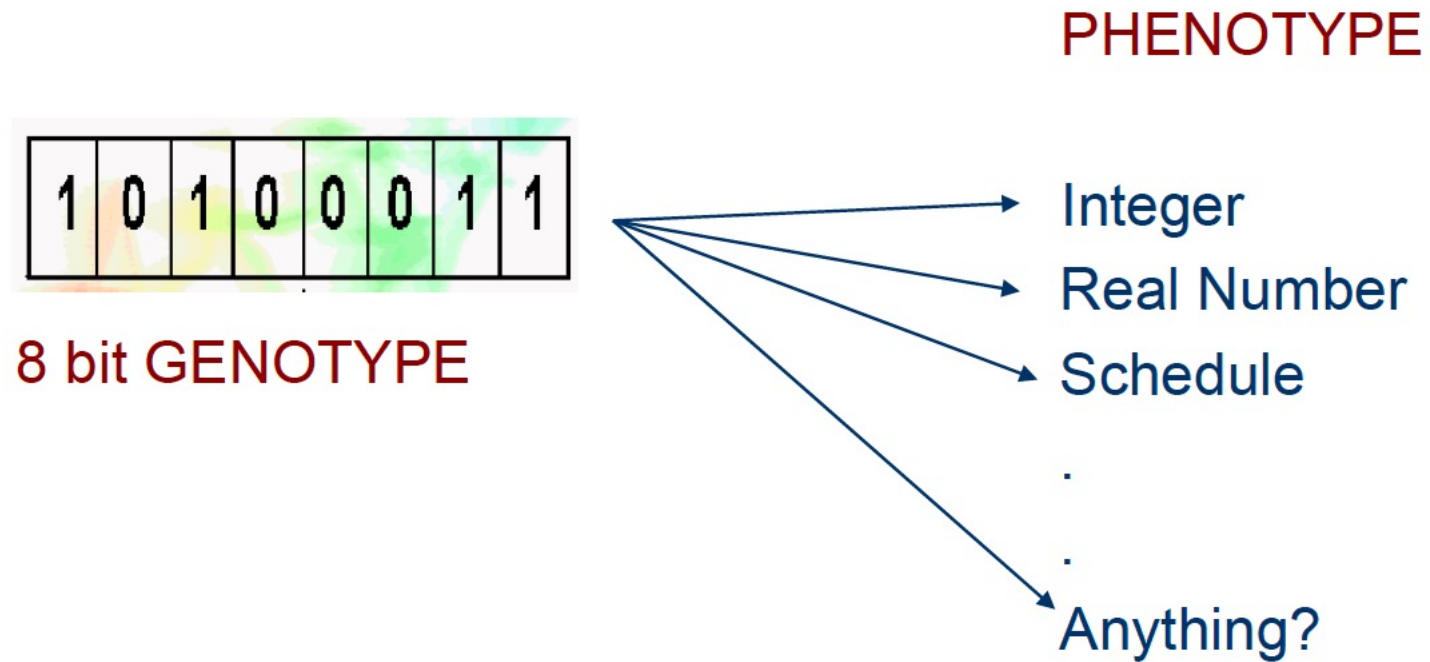
Designing a Representation

- We have to come up with a method of representing an individual as a genotype.
- The way we choose to do it must be relevant to the problem that we are trying to solve.
- When choosing a representation, we have to bear in mind how the genotypes will be evaluated and what the genetic operators might be.

Binary Representation

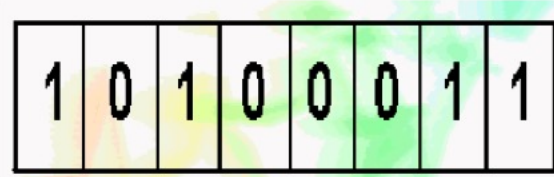
- A chromosome should in some way contain information about solution that it represents. The most used way of encoding is a binary string. A chromosome then could look like this:
 - Chromosome1 1101 1001 0011 0110
 - Chromosome2 1101 1110 0001 1110
- Each chromosome is represented by a binary string. Each bit in the string can represent some characteristics of the solution. Another possibility is that the whole string can represent a number

Example



WHAT CAN IT REPRESENT ???

- Phenotype can be an integer number



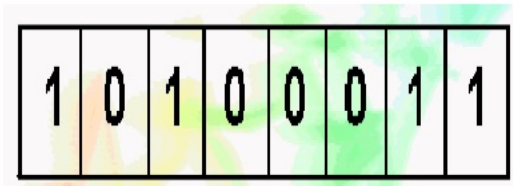
$$= 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$

$$= 128 + 32 + 2 + 1 = 163$$

- Phenotype can be a real number
e.g. a number between 2.5 and 20.5 using 8 binary digits



$$X = 2.5 + (163/256) (20.5 - 2.5) = 13.9609$$



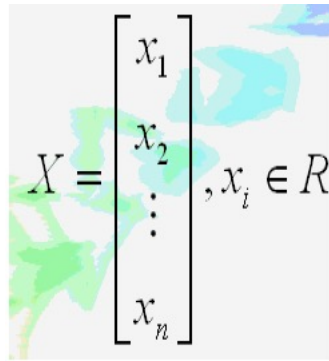
Job	Time step
1	2
2	1
3	2
4	1
5	1
6	1
7	2
8	2

- Phenotype can be a schedule
- e.g. 8 jobs , 2 time steps

Real Valued Representation

- A very natural encoding if the solution we are looking for is a list of real-valued numbers, then encode as a list of real-valued numbers (not as a string of 1's and 0's)
- Lots of applications , e.g. parameter optimization

- Individuals are represented as a tuple of n real valued numbers:


$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, x_i \in R$$

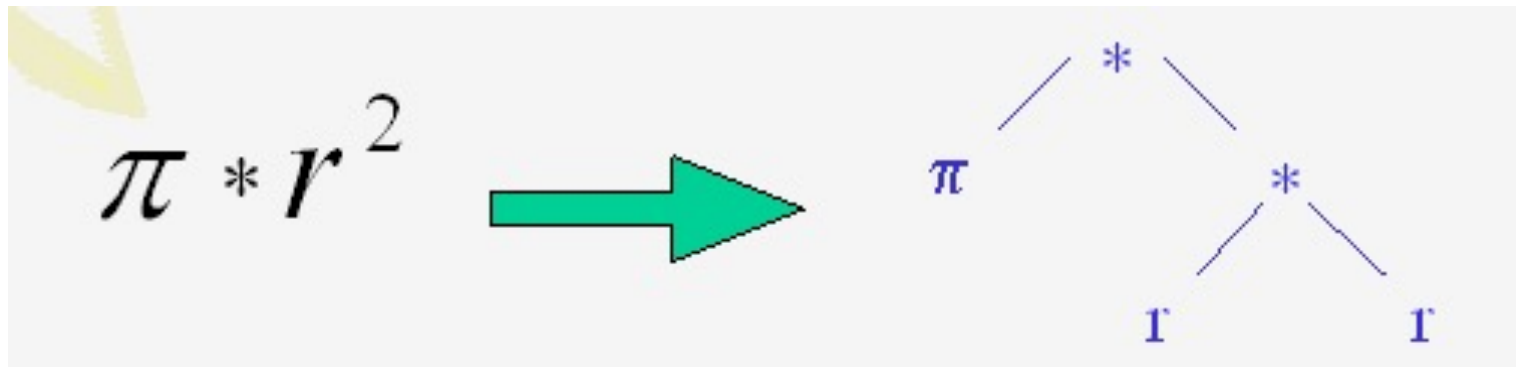
- The fitness function maps tuples of real numbers to a single real number: $f: R^n \rightarrow R$

Order Based Representation

- Individuals are represented as permutations
- Used for ordering, sequencing problems
- Famous example: traveling salesman problem where every city gets assigned a unique number from 1 to n . A solution could be (5, 4, 2, 1, 3).
- Famous example: N-queen problem (or N^2 -queen)
- Needs special operators to make sure the individuals stay in **valid** permutations.

Tree Based Representation

- Individuals in the population are trees
- Any expression can be drawn as a tree of function and terminals
 - Functions : sine, cosine, add, sub, if-then-else
 - Terminals : X, Y, 0.456, true, false, sensor 1,



Tree Based: Closure and Sufficiency

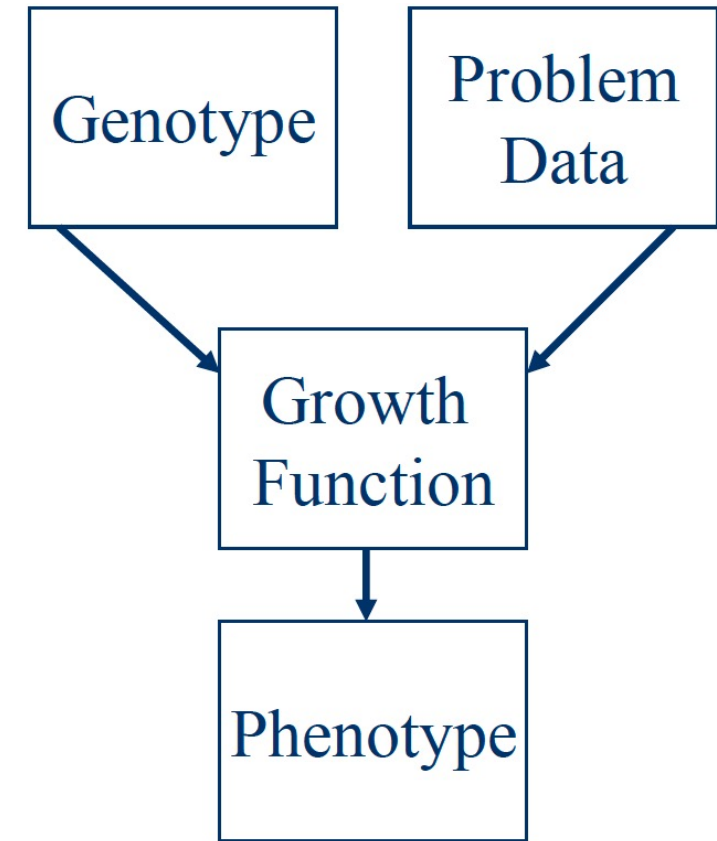
- In the tree based representation we need to specify a function set and a terminal set. It is very desirable that these sets both satisfy closure and sufficiency.
- By closure we mean that each of the functions in the function set is able to accept as its arguments any value and data type that may possibly be returned by some other function or terminal.
- By sufficiency we mean that there should be a solution in the space of all possible programs constructed from the specified function and terminal sets.

Initialization of a Population

- Usually, at random
- Uniformly on the search space If possible
 - Binary representation: 0 and 1 with probability of 0.5
 - Real-valued representation: uniformly on a given interval (OK for bounded values only)
- Seed the population with previously known values or those from heuristics. With care:
 - Possible loss of genetic diversity
 - Possible unrecoverable bias

Mapping a Phenotype from a Genotype

- Sometimes producing the phenotype from the genotype is a obvious process.
- Other times the genotype might be a set of parameters to some algorithm, which work on the problem data to produce the phenotype.



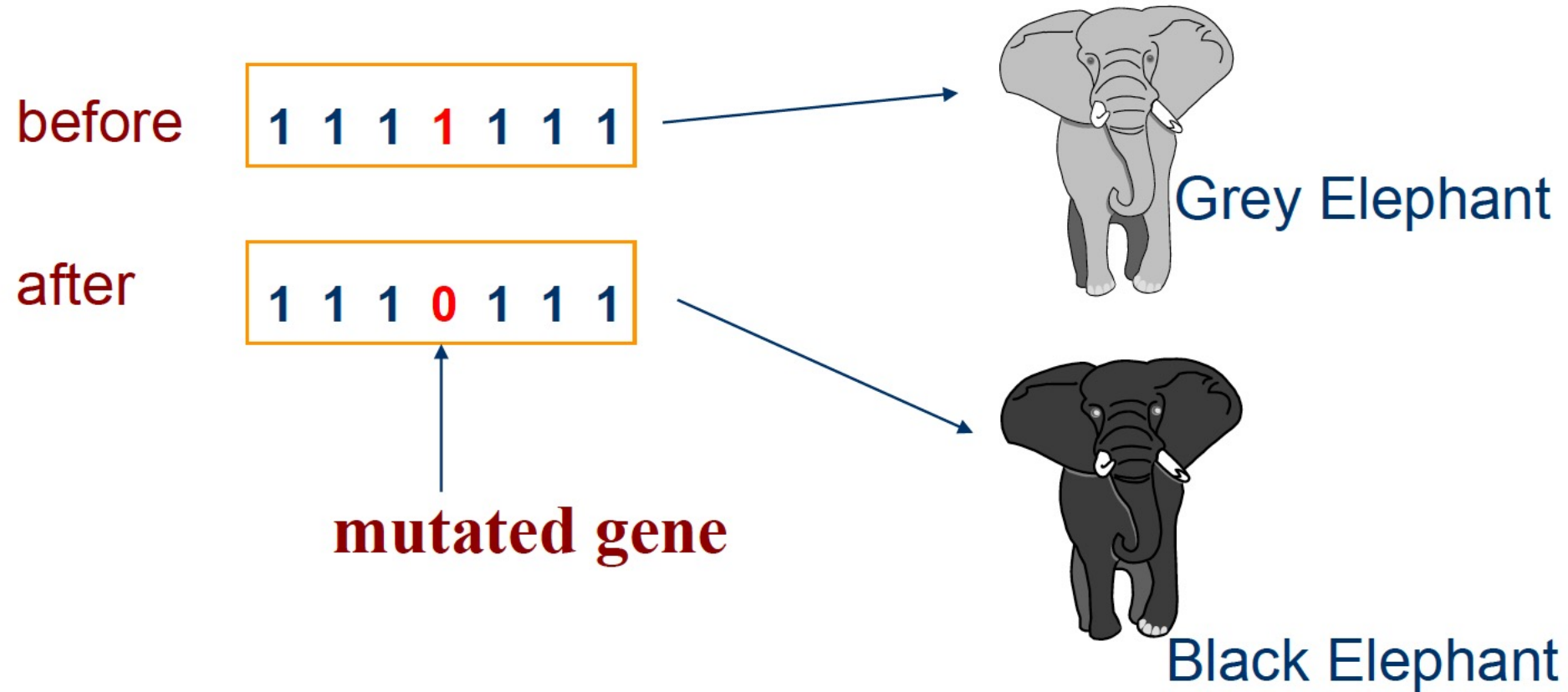
Evaluating an Individual

- By far the most costly step for real applications
 - Do not re-evaluate unmodified individuals
- It might be a subroutine, a black-box simulator, or any external process (e.g., robot experiment).
- The effectiveness of the process depends on the choice of the fitness function.
- You could use approximate fitness, but not for too long (fitness inheritance or fitness approximation).
- Constraint handling—what if the phenotype breaks some constraint of the problem:
 - Penalize the fitness
 - Specific evolutionary method
- Multi-objective evolutionary optimization gives a set of compromise solutions

Mutation Operators

- The mutation operator should allow every part of the search space to be reached.
- The size of mutation is important and should be controllable.
- Mutation should produce valid chromosomes.

Binary Valued Representation



Mutation probability p_m , for each gene

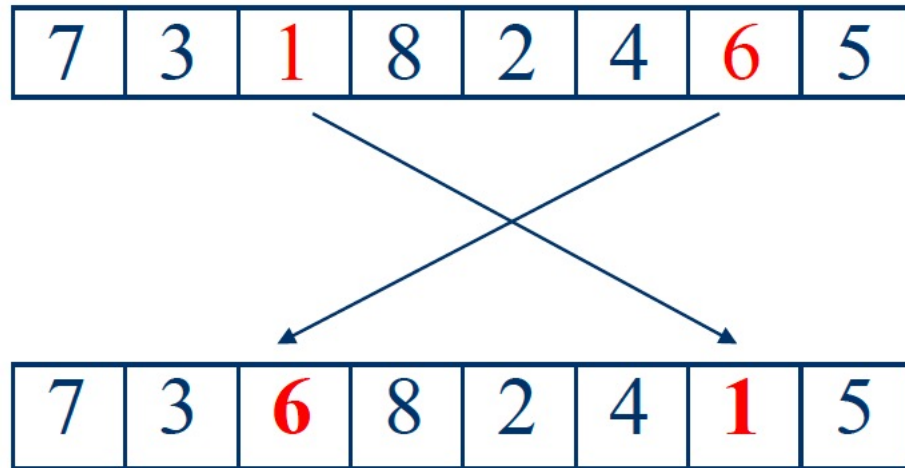
Real Valued Representation

- Perturb values by adding some random noise. Often, a Gaussian/normal distribution $N(0, \theta)$ is used, where
 - 0 is the mean value
 - θ is the standard deviation
- and $x'_i = x_i + N(0, \theta_i)$ for each parameter

Order Based Representation

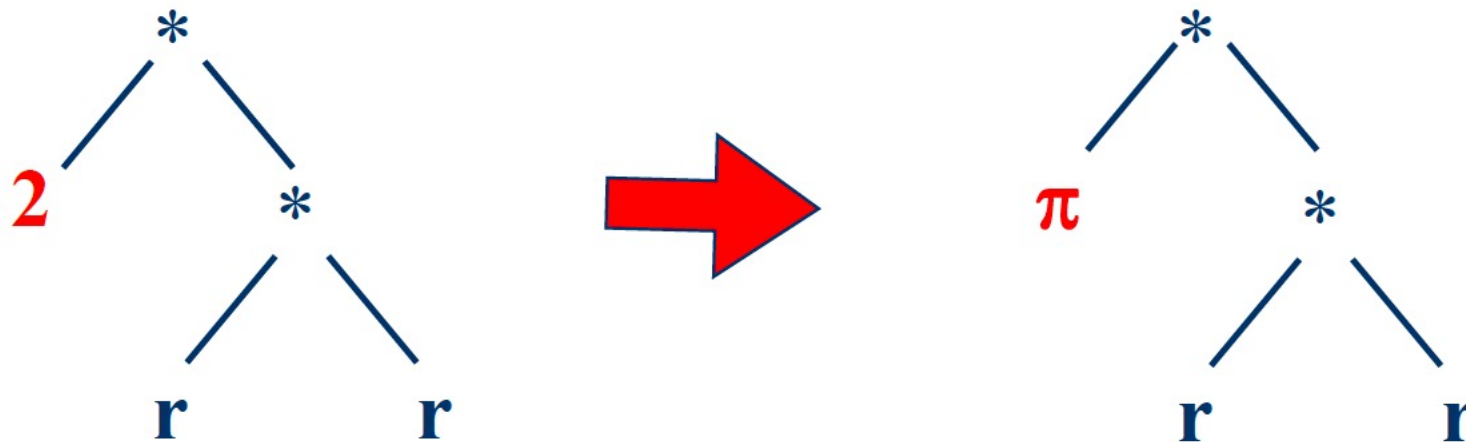
- Randomly select two different genes and swap them.

-



Tree Based Representation

- Single point mutation selects one node and replaces it with a similar one.



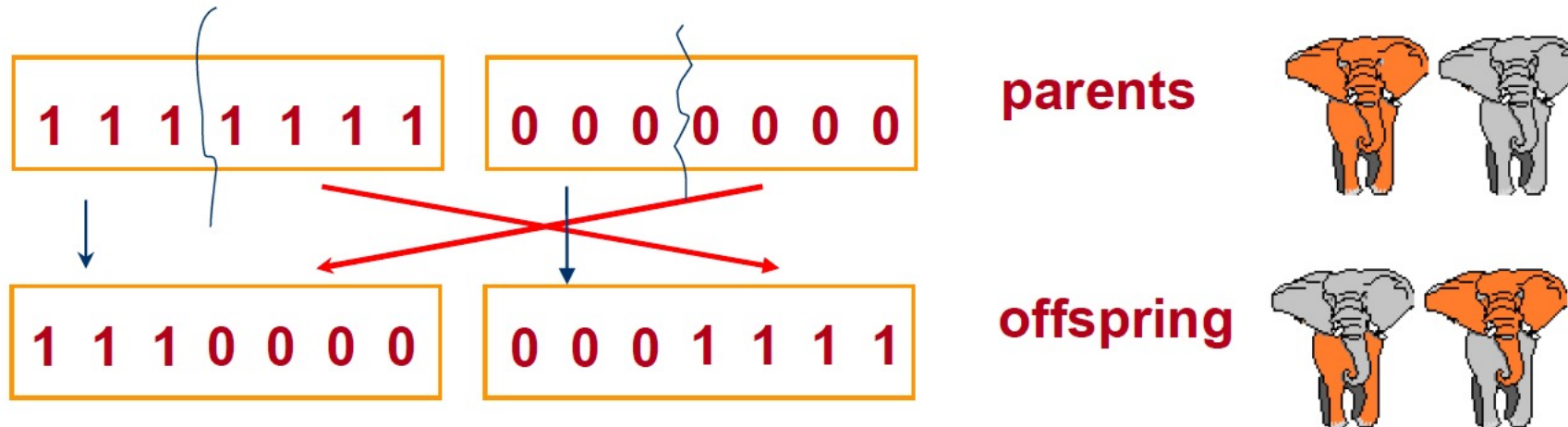
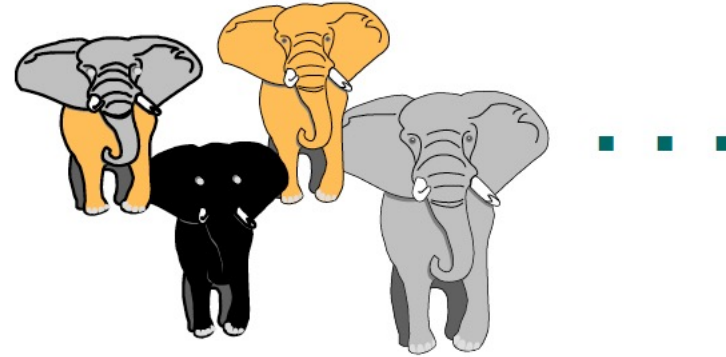
Polynomial Mutation

Recombination/Crossover Operators

- The child should inherit something from **both** parents. If this is not the case then the operator is a mutation operator.
- The recombination/crossover operator should be designed in conjunction with the representation so that recombination is not always catastrophic.
- Recombination should produce valid chromosomes.

Binary Valued Representation

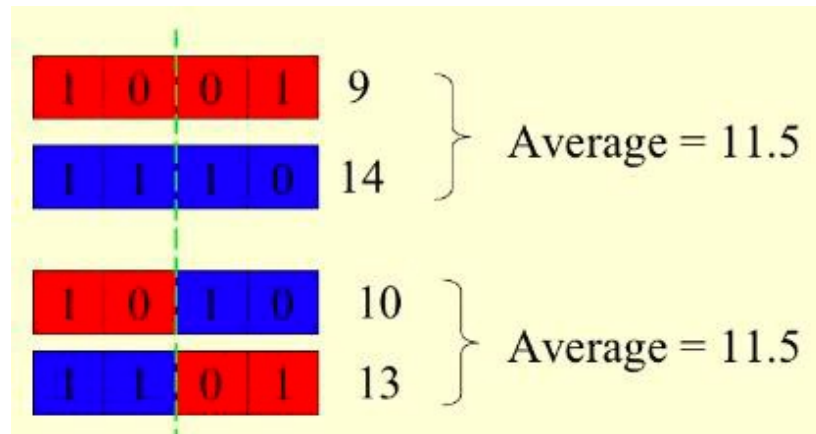
Whole Population:



1-point crossover

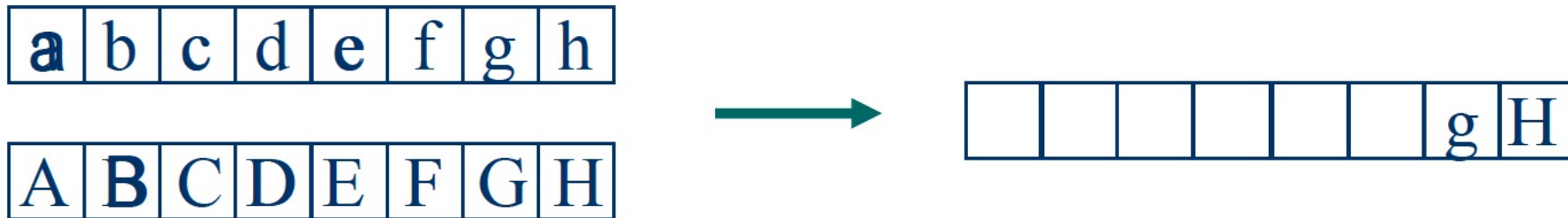
- Important properties of 1-point crossover

- The average of the decoded parameter values is the same before and after the crossover

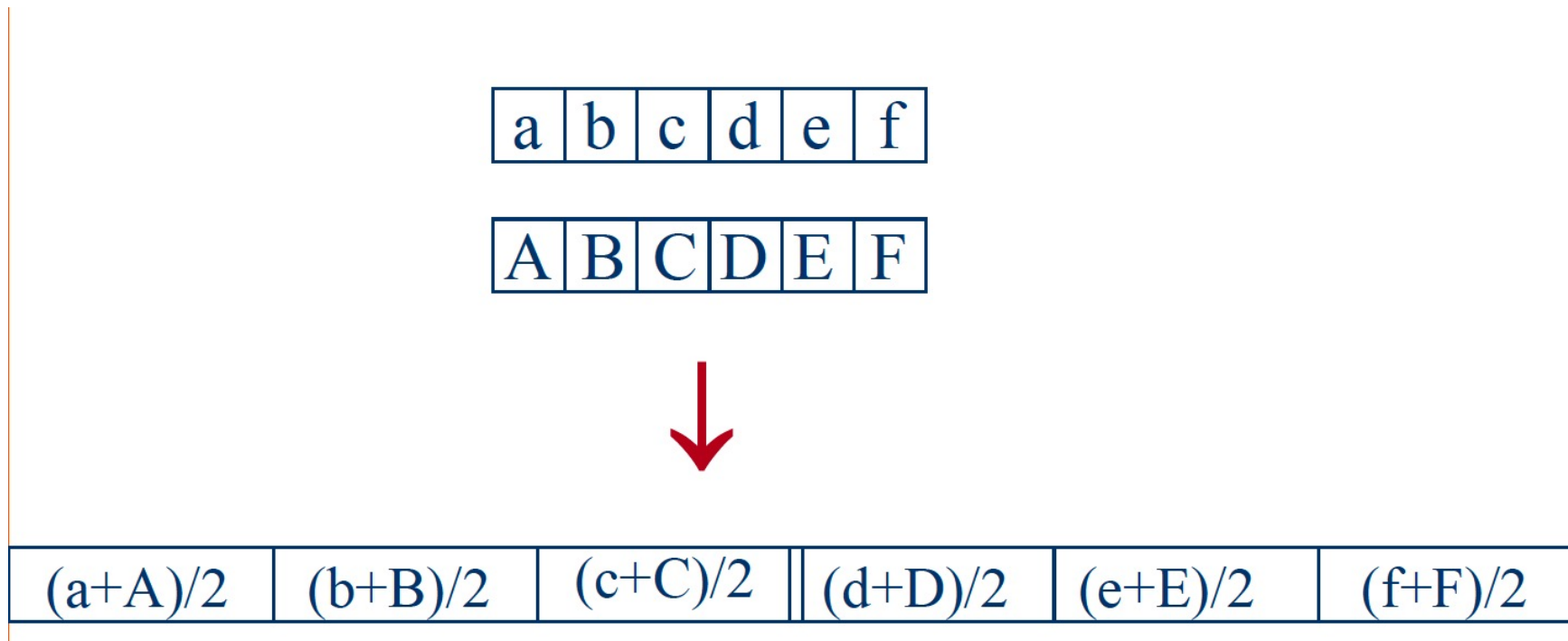


Real Valued Representation

- Uniform crossover: given two parents one child is created as follows



- Intermediate recombination (arithmetic crossover): given two parents one child is created as follows

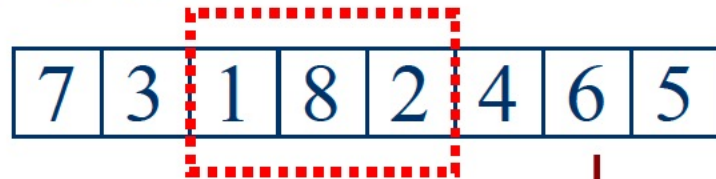


Simulated Binary Crossover (SBX)

Order Based Representation

- Choose an arbitrary part from the first parent and copy this to the first child.
- Copy the remaining genes that are not in the copied part to the first child:
 - starting right from the cut point of the copied part
 - using the order of genes from the second parent
 - wrapping around at the end of the chromosome
- Repeat this process with the parent roles reversed.

Parent 1



7, 3, 4, 6, 5

Parent 2



4, 3, 6, 7, 5

order



Child 1



Tree Based Representation

