Genetic Algorithms
Various Operators

Sabah Sayed

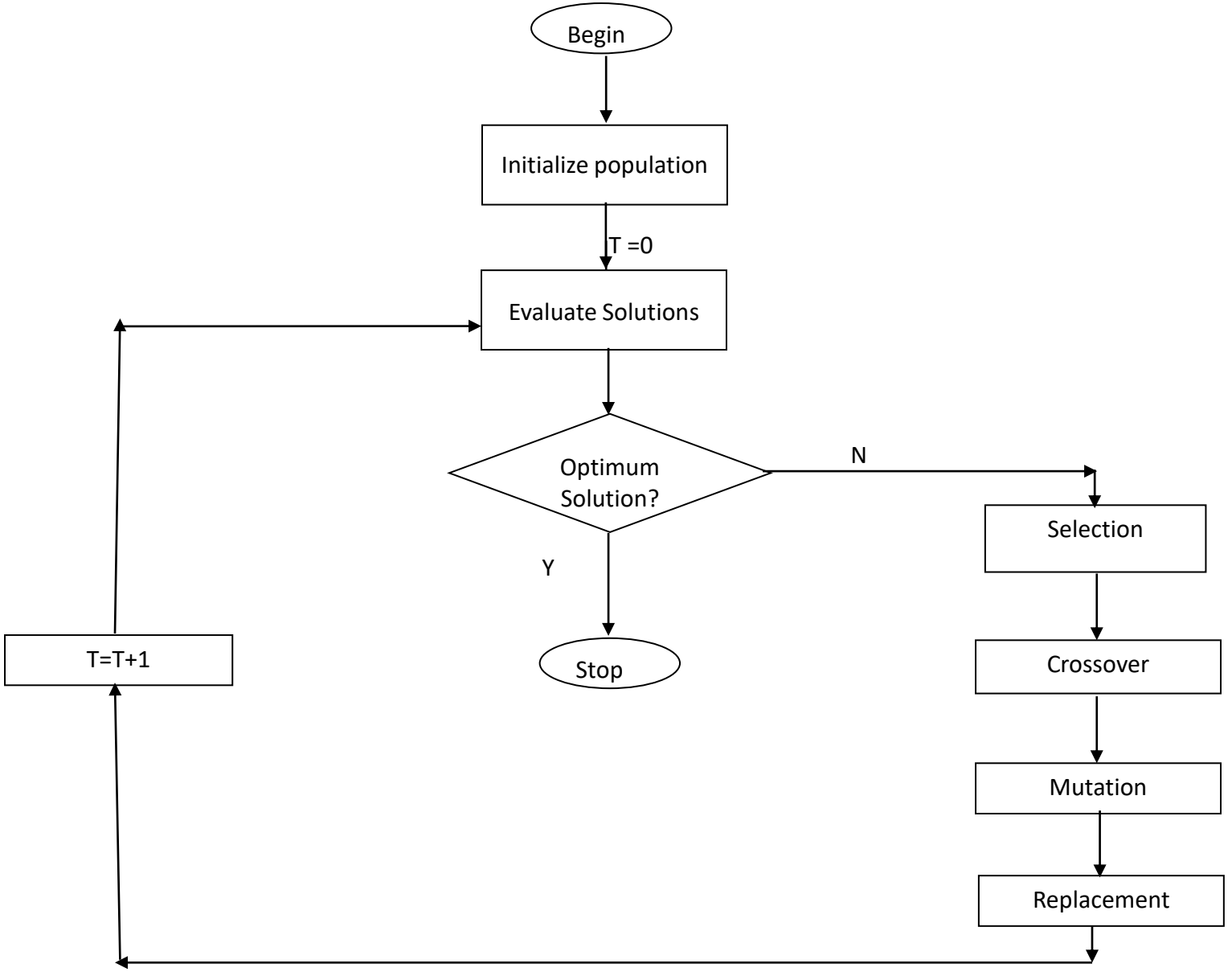
Department of Computer Science

Faculty of Computers and Artificial Intelligence

Cairo University

Egypt

Remember: Mechanism Of GAs



Various Strategies for the Genetic Operators

- Different GAs use different …… strategies.
  - Representation (encoding/decoding)
  - Crossover
  - Mutation
  - Selection
  - Replacement

Various Representations

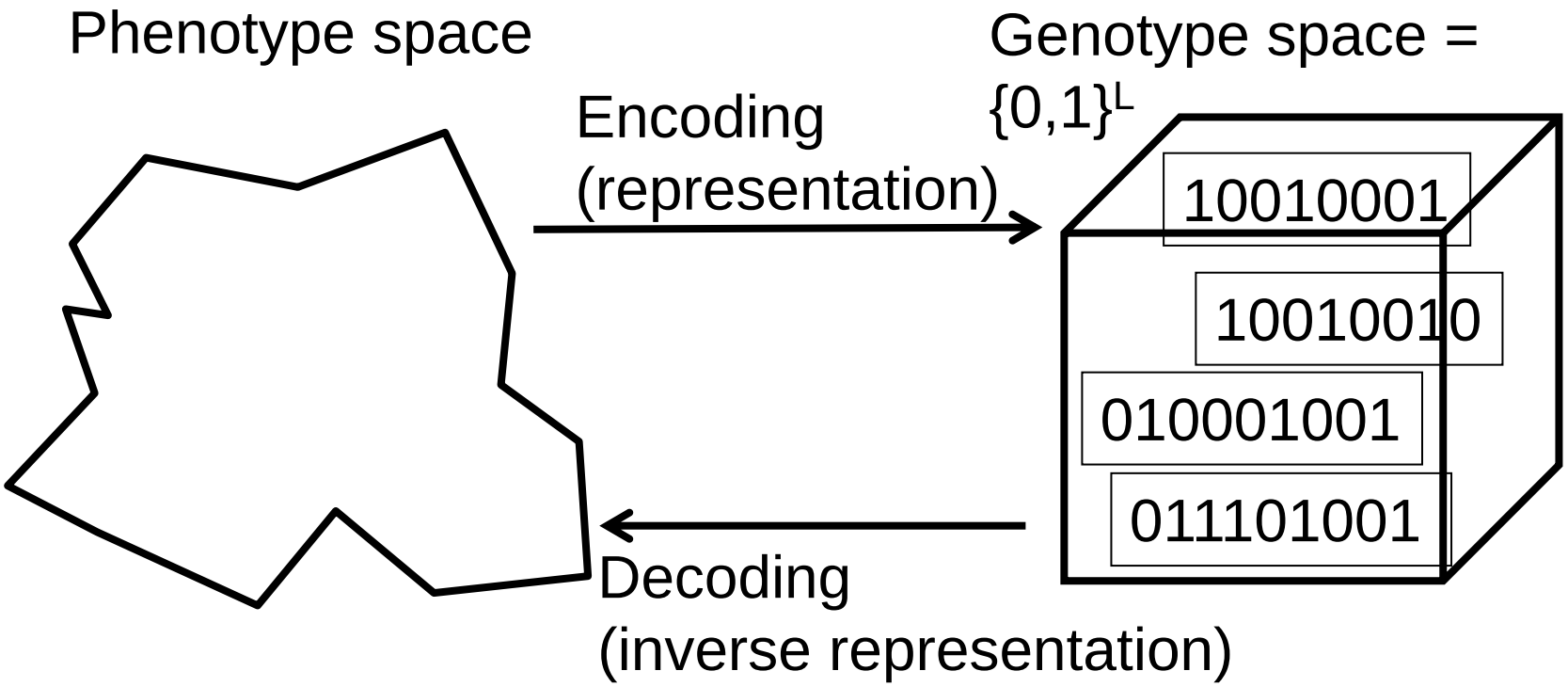
String Array?

Character Array?

Floating Point?

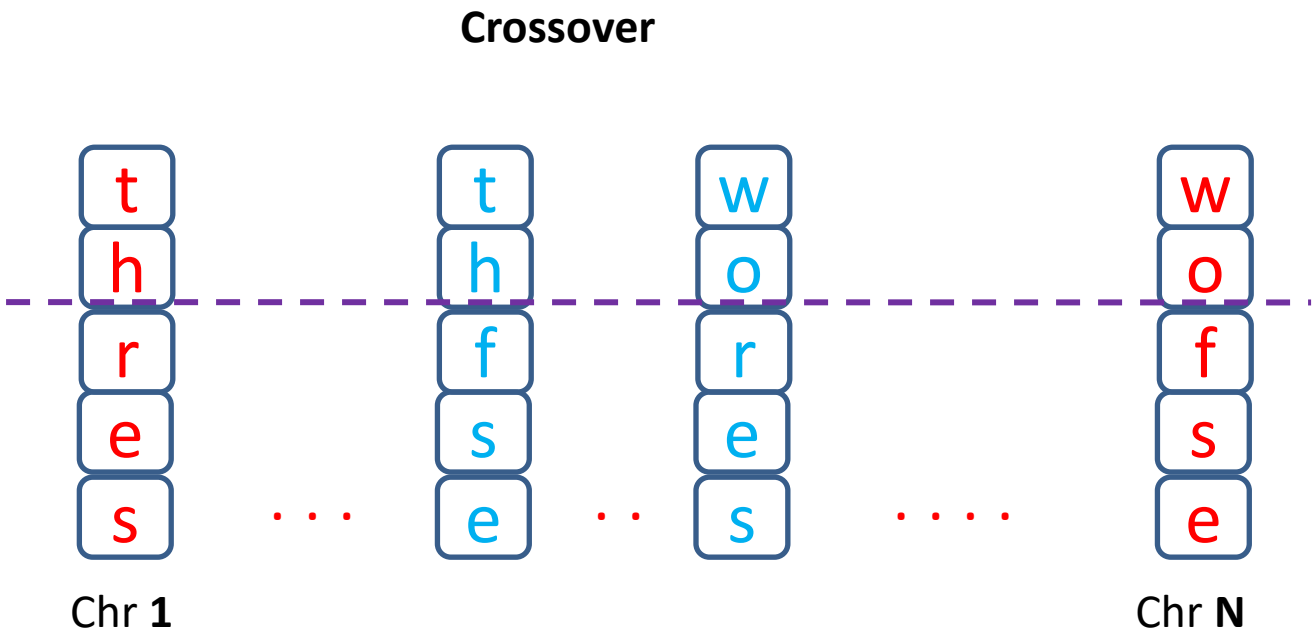
Permutation?

Integer?



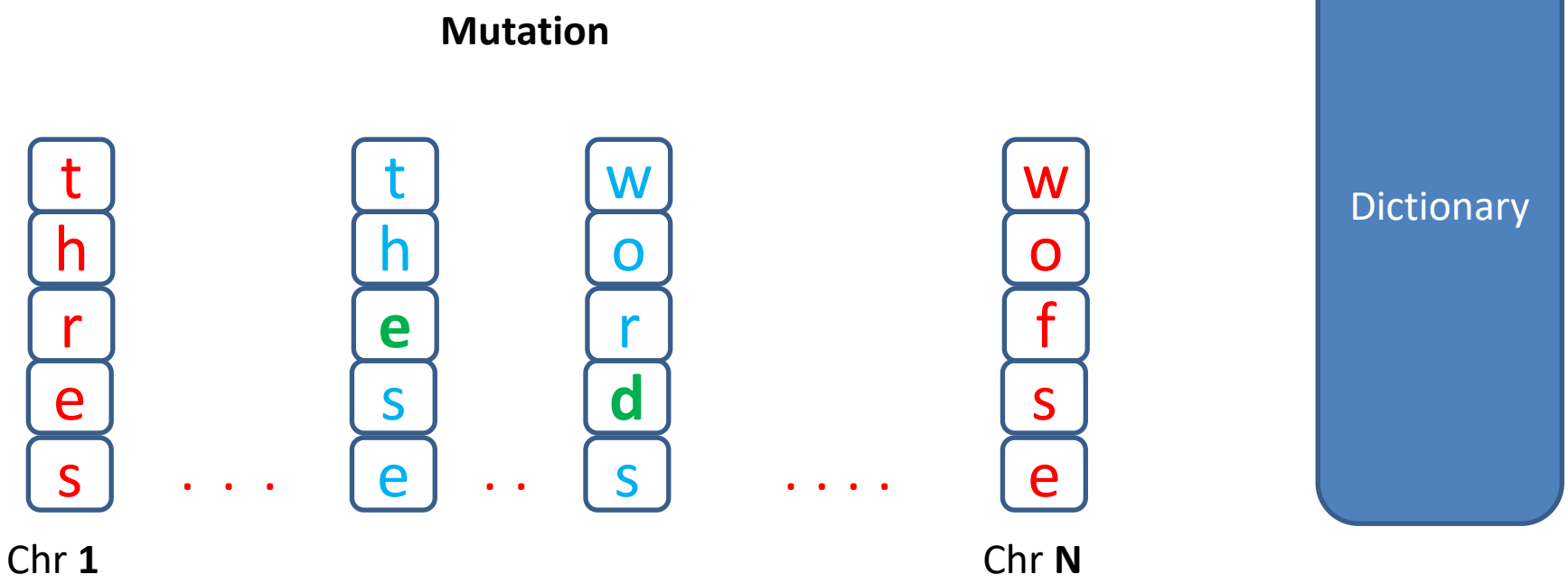
English Word Generation Example

Problem: Generate valid English word of length 5 characters



English Word Generation Example

Problem: Generate valid English word of length 5 characters



Integer Representation

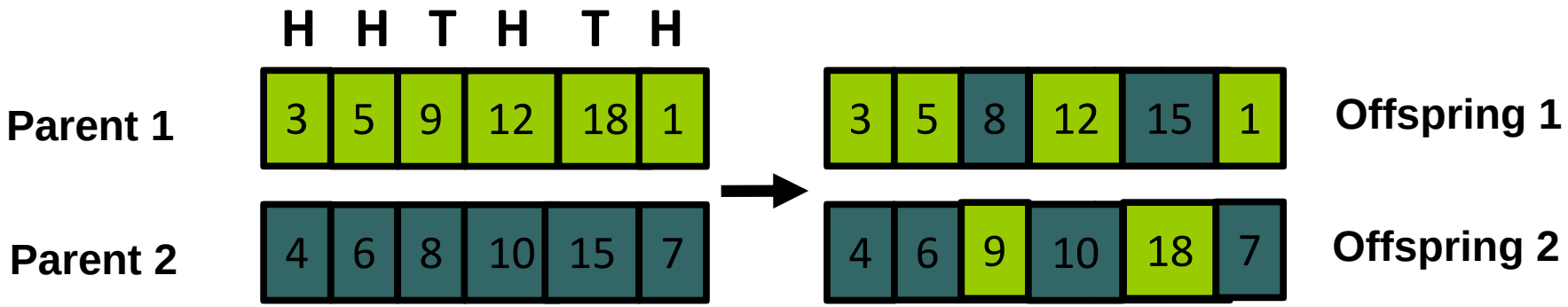
- Some problems naturally have integer variables, e.g. image processing parameters
- Others take *categorical* values from a fixed set e.g. {blue, green, yellow, pink}
- N-point / uniform crossover operators work
- Extend bit-flipping mutation to make
  - "**creep**" i.e. more likely to move to similar value, For ordinal problems it is hard to know correct range for creep
  - **Random choice** (categorical variables)

Uniform Crossover for Integers

- Assign 'heads' to one parent, 'tails' to the other
- Flip a coin for each gene of the first child
- Make an inverse copy of the gene for the second child

- Example:

Suppose H for Parent1 and T for Parent 2



Uniform Crossover for Integers

- Inheritance is independent of position
- Applicable for binary representation
- How to implement it (programming)
- N-point crossover ???

Subset: **BAABBAABBB** **(Randomly generated)**

Parents: **1010001110** **0011010010**

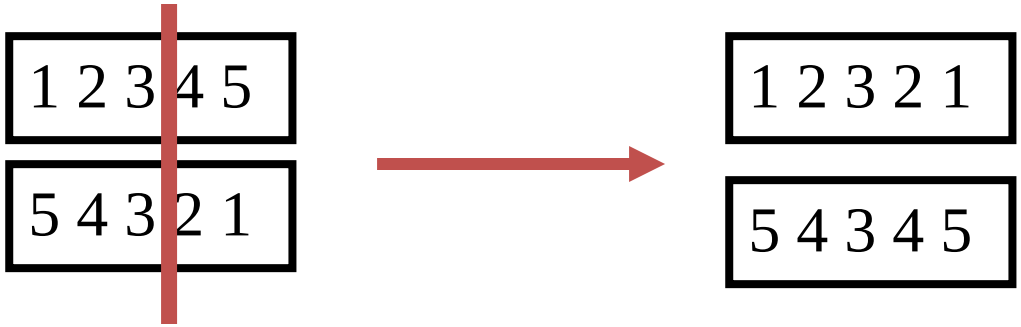
Offspring: **0011001010** **1010010110**

Permutation Representations

- Ordering/sequencing problems form a special type
- Task is (or can be solved by) arranging some objects in a certain order
  - Example: sort algorithm: important thing is which elements occur before others (order)
  - Example: Travelling Salesman Problem (TSP) : important thing is which elements occur next to each other (adjacency)
- These problems are generally expressed as a permutation:
  - if there are *n* variables then the representation is as a list of *n* integers, each of which occurs exactly once

Crossover for Permutations

- Normal crossover operators will often lead to inadmissible solutions



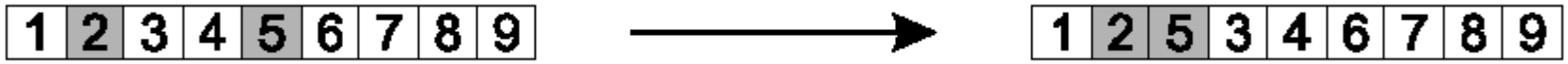
- Many specialised operators have been devised which focus on combining order or adjacency information from the two parents

Mutation operators for permutations

- Normal mutation operators lead to inadmissible solutions
  - e.g. bit-wise mutation : let gene *i* have value *j*
  - changing to some other value *k* would mean that *k* occurred twice and *j* no longer occurred
- Therefore must change at least two values
- Mutation parameter now reflects the probability that some operator is applied once to the whole string, rather than individually in each position

Insert Mutation for permutations

- Pick two allele (gene) values at random
- Move the second to follow the first, shifting the rest along to accommodate
- Note that this preserves most of the order and the adjacency information



Swap mutation for permutations

- Pick two alleles (genes) at random and swap their positions
- Preserves most of adjacency information (4 links broken)
- disrupts order more

Inversion mutation for permutations

- Pick two alleles at random and then invert the substring between them.
- Preserves most adjacency information (only breaks two links) but disruptive of order information

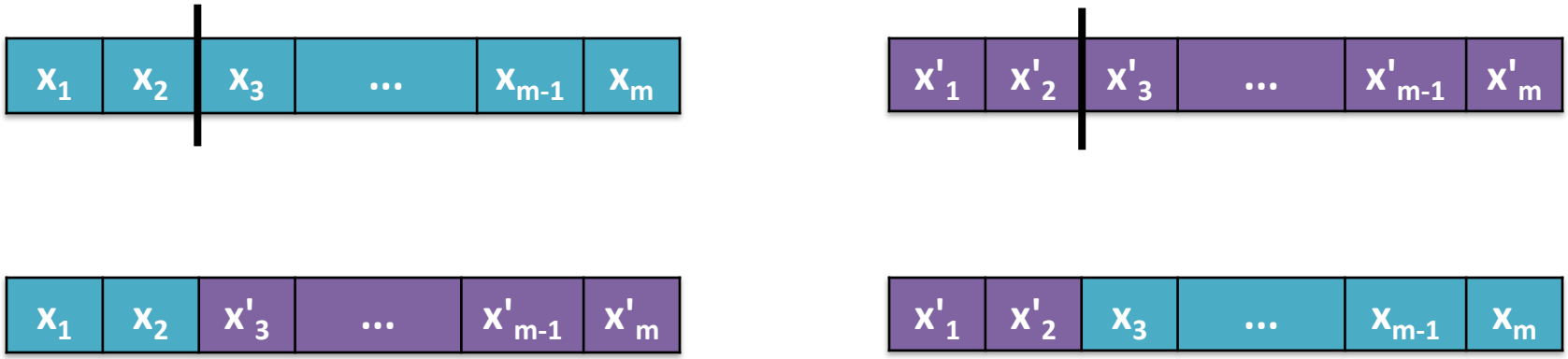
Floating-point (real values) Representation

- Many problems occur as real valued problems, e.g. continuous parameter optimization $f : \mathcal{R}^n \rightarrow \mathcal{R}$
- The chromosome will be an array of floating point variables

0.5	0.2	0.6	0.8	0.7	0.4	0.3	0.2	0.1	0.9

- This can serve in optimization of a multi-variate function $y=f(x_1, x_2, \ldots x_m)$
- Crossover is the same as in bit-string chromosomes.
- Mutation is different

Crossover over FP Chromosomes



Mutation over FP Chromosomes

- General scheme of floating point mutations

$$\bar{x} = \langle x_1, \ ..., \ x_l \rangle \rightarrow \bar{x}' = \langle x_1', ..., \ x_l' \rangle$$
$$x_i, x_i' \in [LB_i, UB_i]$$

- Two kinds of FP mutations:
  - Uniform
  - Non-uniform

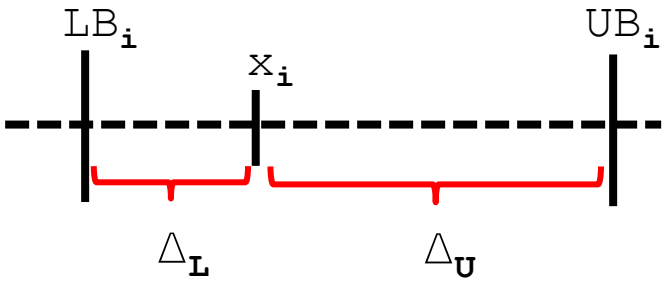
Uniform FP Mutation

$$x_i' \text{ drawn randomly (uniform) from } [LB_i, UB_i]$$

x_1	x_2	...	x_i	...	x_{m-1}	x_m

Given the above chromosome **X** in a particular generation **G**:

- Each gene (variable) has a range
- Gene x_i is a FP value inside chromosome **X** at generation **G**
- To mutate gene x_i
  1. Generate random number $r_{i1} \in [0, 1]$
     - $\Delta = \Delta_L$ if $r_{i1} \leq 0.5$
     - $\Delta = \Delta_U$ if $r_{i1} > 0.5$
     - This means equal chance to go left or right
  2. Generate random number $r_{i2} \in [0, \Delta]$
     - if $\Delta = \Delta_L$ then $x_{i\text{-new}} = x_i - r_{i2}$
     - if $\Delta = \Delta_U$ then $x_{i\text{-new}} = x_i + r_{i2}$



$$\Delta_L = x_i - LB_i$$

$$\Delta_U = UB_i - x_i$$

Non-uniform FP Mutation

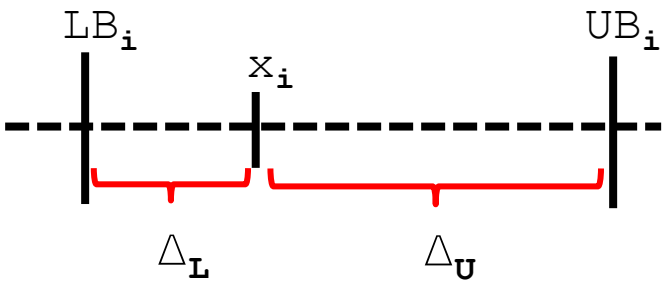
Some methods proposed such as time-varying range of change

x_1	x_2	...	x_i	...	x_{m-1}	x_m

Given the above chromosome **X** in a particular generation **G**:

- Each gene (variable) has a range
- Gene x_i is a FP value inside chromosome **X** at generation **G**
- To mutate gene x_i
  1. Generate random number $r_{i1} \in [0, 1]$
     - $\mathbf{y} = \Delta_L$ if $r_{i1} \leq 0.5$
     - $\mathbf{y} = \Delta_U$ if $r_{i1} > 0.5$
  2. Let $\Delta(\mathbf{t}, \mathbf{y})$
     $= $ value of mutation at generation **t**
     $= \mathbf{y}(1 - r^{(1-t/T)^b})$

     where:
     - r = random number $\in$ [0, 1]
     - t = current generation
     - T = maximum number of generations
     - b = dependency factor $\approx$ 1…5



$$\Delta_L = x_i - LB_i$$

$$\Delta_U = UB_i - x_i$$

Non-uniform FP Mutation

- Analysis of Equation:

  $\Delta(t,y)$ = value of mutation at generation t

  $= y(1-r^{(1-t/T)^b})$

  where:

  - r = random number $\in [0, 1]$
  - t = current generation
  - T = maximum number of generations
  - b = dependency factor $\approx 1 \ldots 5$

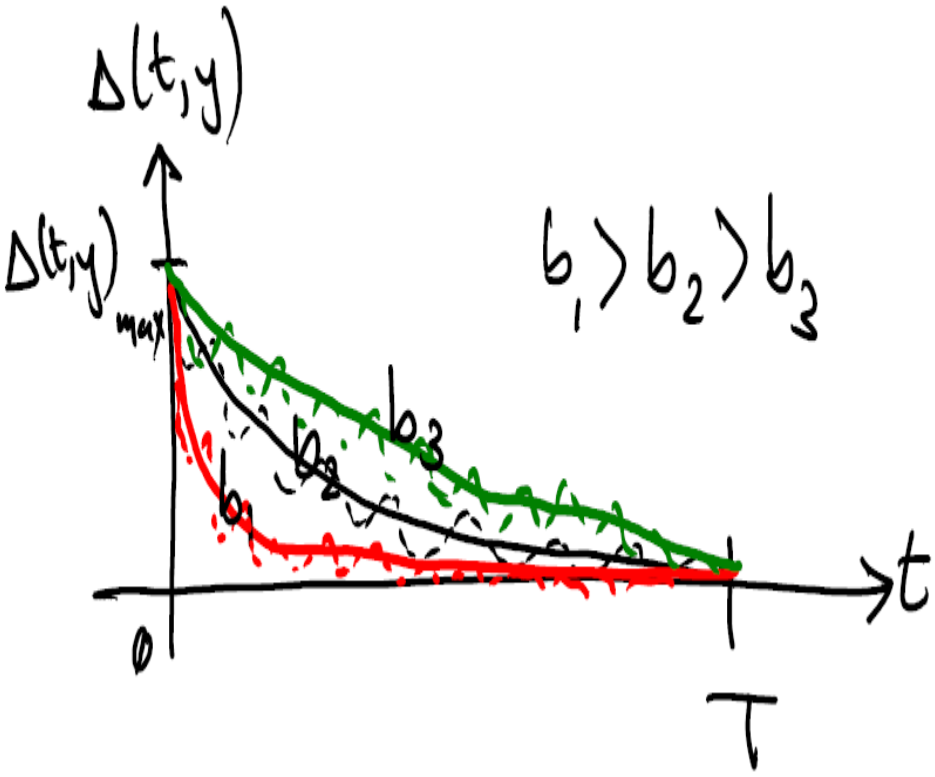
  (Controls the curve of mutation)

  - At $t=0$:

  $\Delta(t,y)$ = Maximum value of mutation

  - At $t=T$:

  $\Delta(t,y) = 0$ (No mutation)



Crossover or Mutation?

- Decades of long debate: **which one is better or necessary?**

- Answer (at least, rather wide agreement):
  - it depends on the problem, but
  - in general, it is good to have both
  - both have different roles
  - mutation-only-GA is **possible**, crossover-only-GA **would not** work. Why??

Mutation

- Causes movement in the search space (local or global)
- Restores lost information to the population
- Mutation is necessary because some important genes might be missing from all the initial population.

Crossover

- It greatly accelerates search early in evolution of a population
- It leads to effective combination of schemata (subsolutions on different chromosomes)

Crossover or Mutation?

- ✓ **Exploration:** Discovering promising areas in the search space, i.e. *gaining information* on the problem
- ✓ **Exploitation:** Optimising within a promising area, i.e. *using information*

There is co-operation AND competition between them

- **Crossover is explorative**: it makes a *big* jump to an area somewhere "in between" two (parent) areas
- **Mutation is exploitative:** it creates random *small* diversions, thereby staying near (in the area of) the parent

Crossover OR mutation?

- Only crossover can combine information from two parents
- Only mutation can introduce new information (alleles)
- Crossover does not change the allele frequencies of the population (thought experiment: 50% 0's on first bit in the population, ?% after performing *n* crossovers)
- To hit the optimum you often need a 'lucky' mutation

Selection Operator

- Purpose: to focus the search in promising regions of the space
- Bias the mating pool (those who can pass on their traits to the next generation) with fitter individuals

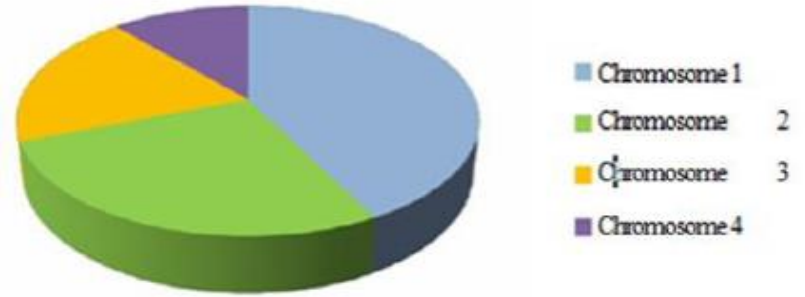
- Selection can occur in two places:
  - Selection from *current generation* to take part in mating (parent selection)
  - Selection from *parents + offspring* to go into next generation (Replacement Strategy)

Fitness Based Competition

- Selection operators work on whole individual
  - i.e. they are representation-independent
- Distinction between selection
  - operators: define selection probabilities
  - algorithms: define how probabilities are implemented
- Chance to be selected as parent proportional to fitness:
  - Roulette wheel selection
  - Rank selection
- To avoid problems with fitness function
  - Tournament selection algorithm

Rank Selection Technique

- Roulette wheel suffers from premature convergence.
- An alternative is **Rank Selection**. Attempt to remove problems of FPS by basing selection probabilities on ***relative*** rather than ***absolute*** fitness
- Based on sorting of individuals by decreasing fitness



Steps:

1. Rank selection first ranks the population and then every chromosome receives fitness from this ranking.
2. The worst will have fitness 1, second worst 2 etc. and the best will have fitness N (number of chromosomes in population).
3. Then calculate cumulative Fitness.
4. The next steps is same as roulette wheel.

Rank Selection Technique

Example:

Assuming these are the individual fitnesses: 10, 9, 3, 15, 85, 7.

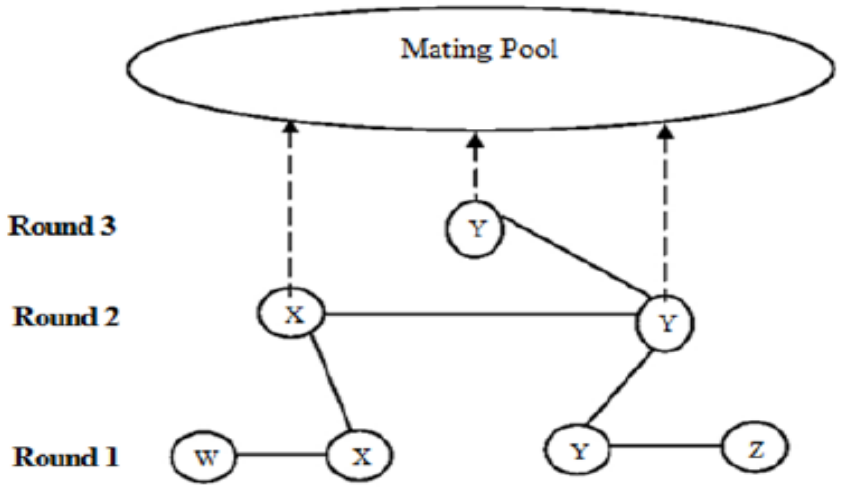
- Sort the individuals according to fitness 3,7,9,10,15,85
- Assign the ranks in **ascending order**: 1: **3**, 2: **7**, 3: **9**, 4: **10**, 5: **15**, 6: **85**
- Sum of all ranks is 1+2+3+4+5+6 = 21 (or using the **gauss formula** : N*(N+1)/2) ➔ (6+1)*6/2 = **21**.
- Compute the probabilities as: 1/21, 2/21, 3/21, 4/21, 5/21, 6/21 ➔ 0.047, 0.095, 0.143, 0.19, 0.24, 0.29
- Apply roulette wheel on those probabilities.

- What are the problems that could arise?
  - ➢ Computationally expensive because it sorts the populations based on fitness value.
- Can lead to slower convergence, because the best chromosomes do not differ so much from other ones.
- It preserves diversity hence leads to a successful search.

Tournament Selection Technique

Algorithm:

- Choose ***n*** individuals randomly
- Pick the one with highest fitness
- Place ***n*** copies of these individual in the mating pool
- Repeat the process till all in the original population have been chosen

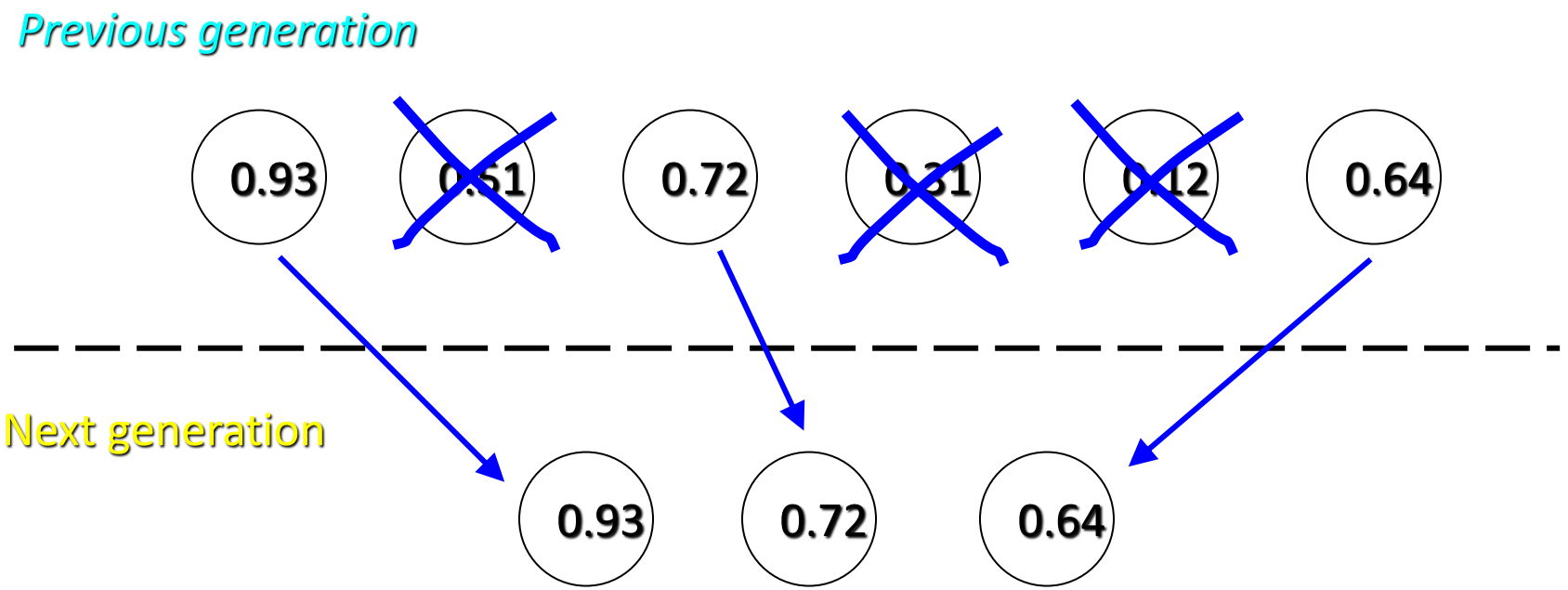


Replacement - Survival of The Strongest

- Cutoff selection:

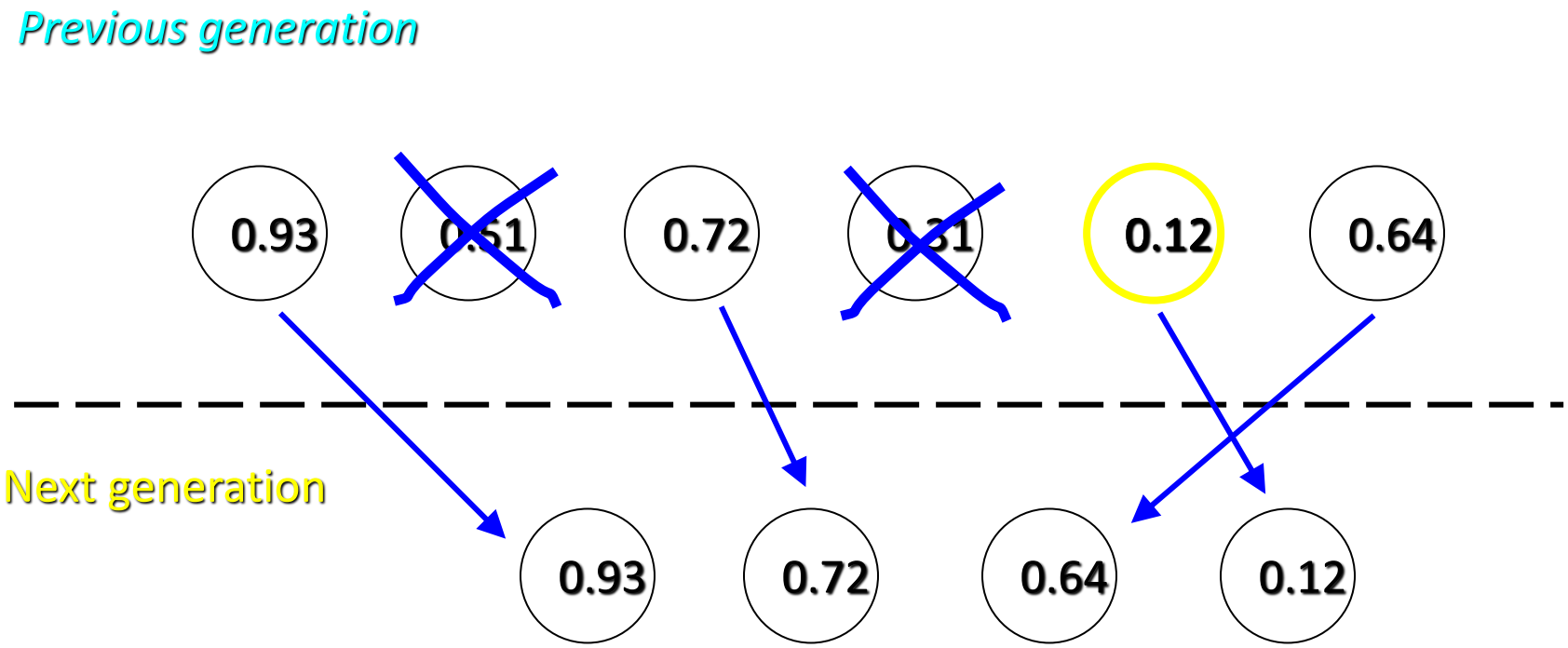
Select only those that are above a certain cutoff for the target function.

- Throw away the weak half of the population.



Replacement - Some Weak Solutions Survive

Mixing Strategy:
Select the strongest solutions with some weak ones.



Why ??

Replacement Strategies

- Three replacement schemas:
  - Generational Replacement (GGA):
    - Mate enough individuals to generate `pop_size` offspring
    - each individual survives for **exactly one generation**
    - the entire set of parents is replaced by the offspring
  - Steady-state Replacement (SSGA):
    - Specific number (K) of individuals are selected for reproduction, and offspring replace their parents in the next generation
  - Elitist Strategy (Elitism):
    - It is steady-state replacement, but keep best-so-far individuals

Elitism

- A fitness proportional selection doesn't guarantee survival of the fittest.

- Although it is beneficial for some algorithms to throw away the best so far to allow for a balance between **exploration (new solutions)** and **exploitation(find the best within the available solutions)**.

- Exploitation allows the algorithm to converge faster but without enough exploration, it might converge to a local optimum solution.

- To keep the best solution so far from being thrown away by crossover or mutation, the elitism option is used to keep one or more of the best solutions discovered so far and copy them to the next generation.

Population Models

- Generation Gap
  - The proportion of the population replaced from one Generation to the next one
  - Equals `1.0` for GGA
  - Equals `K/pop_size` for SSGA

- Generation Overlap
  - The amount of overlap for the population individuals between the current and new generations
  - Relationship between **Generation Gap** and **Generation Overlap ??**