02 GENETIC ALGORITHMS

Spring 2023

CS6431 Natural Language Processing

Credits

1. B1: *Machine learning: an algorithmic perspective*. 2nd Edition, Marsland, Stephen. CRC press, 2015
2. B2: *Principles of Soft Computing*. 3rd Edition. S. N. Sivanandam, S. N. Deepa. Wiley, 2018.
3. https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_parent_selection.htm

Assignment

Read:

B1: Chapter 10 (Till 10.3)

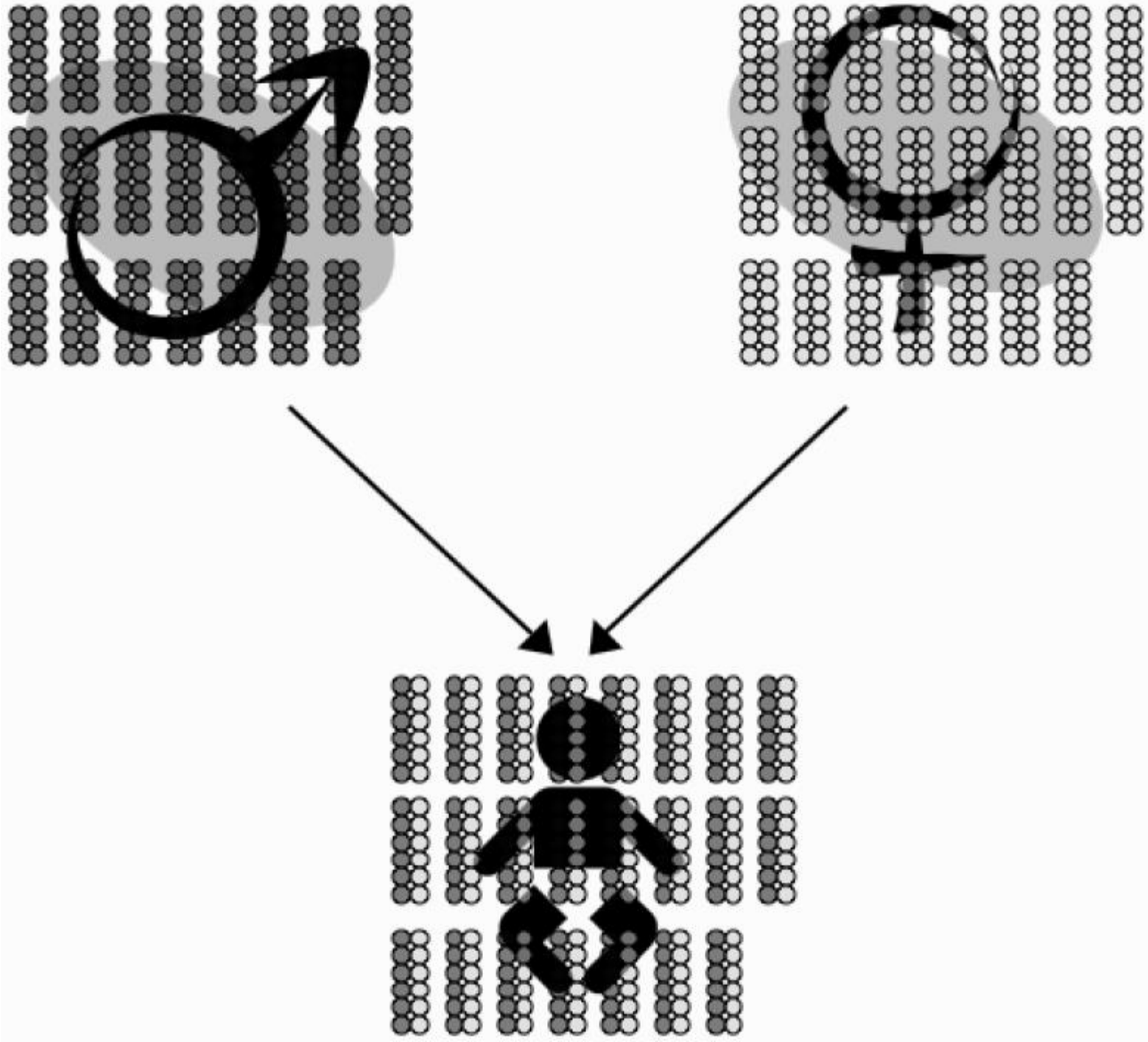
Problems:

B1:

Evolution as a search problem

- Competing animals and “Survival of the fittest”
  - “Fittest” animals
    - Live longer
    - Stronger
    - More attractive
  - Hence, they get more mates and produce more and “healthier” off springs

- Nature is biased towards "fitter" animals for reproduction
  - Basis for Genetic Algorithms
- Parent chromosomes are copied randomly to the child
  - But copy errors can happen: Mutation



Genetic Algorithm (GA)

Modelling a problem as a GA

- A method for representing solutions as chromosomes (or string of characters)
- A way to calculate the fitness of a solution
- A selection method to choose parents
- A way to generate offspring by breeding the parents
- A way to select next generation

One generation

Select, Produce, Repeat!

An Evolutionary Learning Example

- Knapsack Problem

Pack Volume=b

Size A1

Size A2

Size A3

Size A4

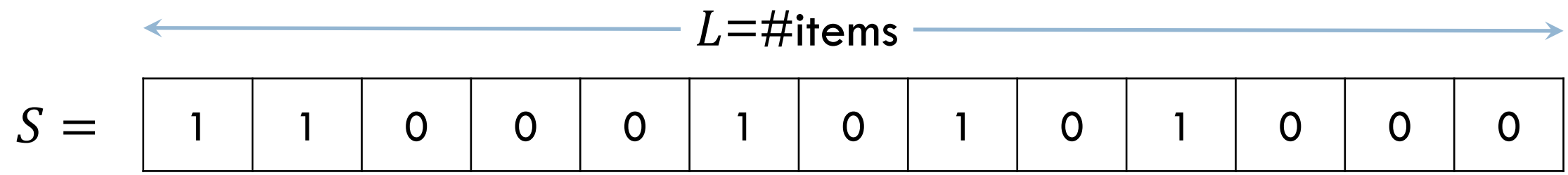
Size A5

Size A6

Can we find k objects which will fit the pack volume b perfectly?

- Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

String Representation



S_i=1 if i^{th} item is included in the solution, 0, otherwise

Fitness Estimation

- More valuable items => Better fitness
  - Fitness Value = Value of all items in a solution
- Solution should be feasible
  - Fitness Value = 0 (if items don't fit)

Exploitation vs. Exploration

- Exploitation: use the best solution so far to explore further solutions
- Exploration: use sub-optimal solutions to explore further solutions

- Exploration: Give infeasible solutions a chance for next generation
- Fitness Value of infeasible solutions = Total value of all items - $2 \times$value of extra items

Selecting Parents for the "Mating Pool"

- Exploitation: select the "fittest" solutions
- Exploration: allow some sub optimal solutions

1. **Tournament Selection**
2. **Truncation Selection**
3. **Fitness Proportional Selection**

Tournament Selection

- Way of selecting one parent (individual) at a time
  - Choose k (the tournament size) individuals from the population at random
  - Choose the best individual from the tournament with probability p
  - Choose the second best individual with probability $p \times (1 - p)$
  - Choose the third best individual with probability $p \times ((1 - p)^2)$

    …and so on (till one parent is selected)
  - Deterministic version: $p = 1$

Truncation Selection

- MP = Pick f fraction of the best strings in the mating pool
- Mating pool $= \frac{1}{f} \times MP$
  - So that mating pool size = Initial population size
- Randomly shuffle the pool and make pairs
- Easy to implement but biased towards exploitation

Truncation selection is a type of selection method used in genetic algorithms (GAs). In truncation selection, a subset of the fittest individuals in the population are selected for reproduction, while the rest are discarded.

The basic idea behind truncation selection is to rank the individuals in the population according to their fitness, and then select the top n individuals for reproduction, where n is a predefined parameter. The individuals with the highest fitness scores have a higher probability of being selected for reproduction than those with lower fitness scores.

Fitness Proportional Selection

- Select (with replacement) a string (α) probabilistically in proportion to its fitness

$$p^{\alpha} = \frac{F^{\alpha}}{\sum_{\alpha'} F^{\alpha'}}$$

fitness

total fitness sum

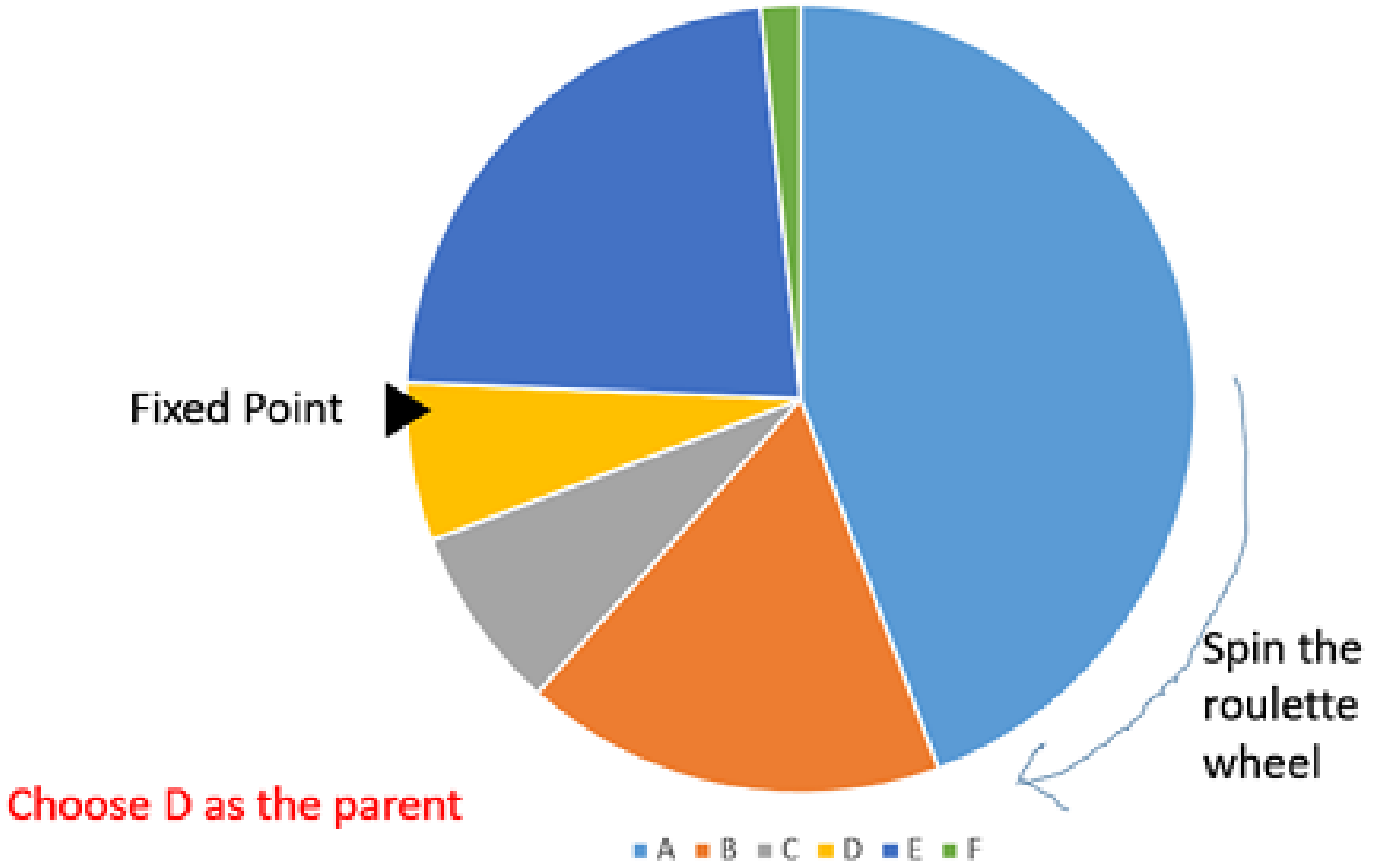
- If F can be –ve

$$p^{\alpha} = \frac{\exp(sF^{\alpha})}{\sum_{\alpha'} \exp(sF^{\alpha'})}$$

s: selection strength

  - higher s gives higher (or less –ve) F^{α} a higher probability

Roulette Wheel Selection



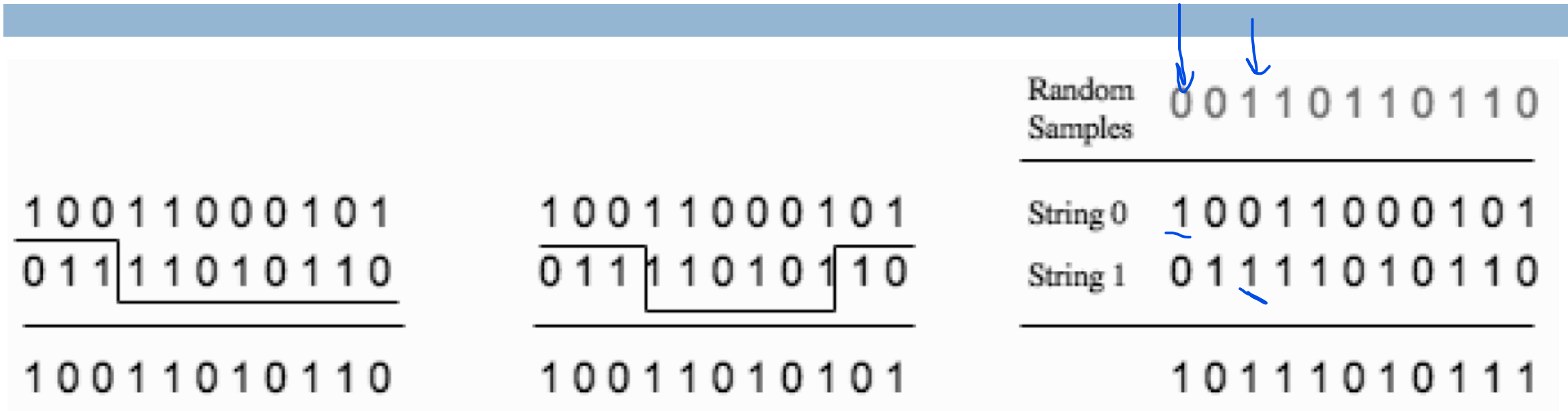
Choose D as the parent

Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Generating Offspring

- Genetic Operators
  - Cross Over
  - Mutation

Crossover



Single Point

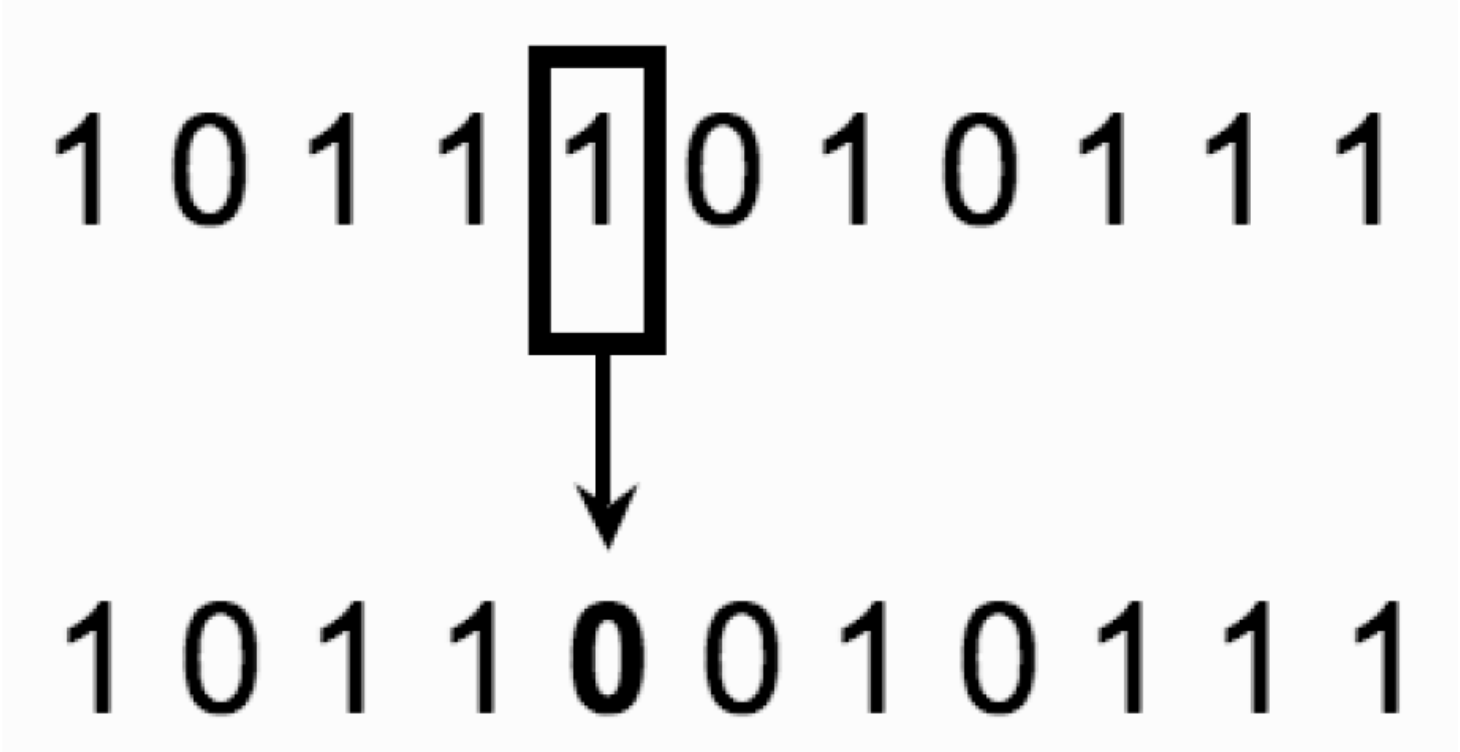
Multi Point

Random

- Global exploration
  - Offspring are radically different than their parents

Mutation

- Flip a bit with a low probability $p = \frac{1}{L}$



Choosing Next Generation

- Choosing only off spring can be risky
  - New generation can have lesser fitness values
  - We can lose a really good sample
- Elitism
  - Select a few fittest strings from parents, say X
  - Replace strings from offspring with X
    - Replacement at random, or,
    - Replace the least fit
- Tournament
  - Two fittest parent and their offspring
  - Tournament winners proceed to the next round

- Elitism and Tournament can lead to premature convergence
  - Both promote fitter members
  - After some time, same set of fittest members keep getting promoted
  - Exploration is downplayed
- Solution:
  - Niching (or "using island populations")
  - Fitness sharing

Niching

- Separate populations into subpopulations
  - Each subpopulation converges independently to different local minima
  - A few members of one sub-population are randomly injected as "immigrants" to another.

Niching and fitness sharing are two techniques used in genetic algorithms (GAs) to promote diversity and maintain a balance between exploration and exploitation of the search space.

Niching is the process of creating and maintaining multiple subpopulations within a larger population, where each subpopulation specializes in a particular area of the search space. The goal of niching is to prevent premature convergence of the population towards a single solution by maintaining diversity and exploring multiple regions of the search space simultaneously

Fitness Sharing

$$F^{\alpha} = \frac{F^{\alpha}}{\#\text{Times } \alpha \text{ appears in a population}}$$

- Biased for uncommon strings
- But, can loose very good common strings

Fitness sharing is a technique that assigns a sharing function to each individual in the population, which reduces the fitness of an individual based on the proximity of other individuals in the same niche. The idea behind fitness sharing is that individuals in the same niche compete for the same resources, and thus, their fitness should be adjusted accordingly. By reducing the fitness of individuals in crowded niches, fitness sharing promotes diversity and encourages the search to explore new regions of the search space.

Modelling a problem as a GA

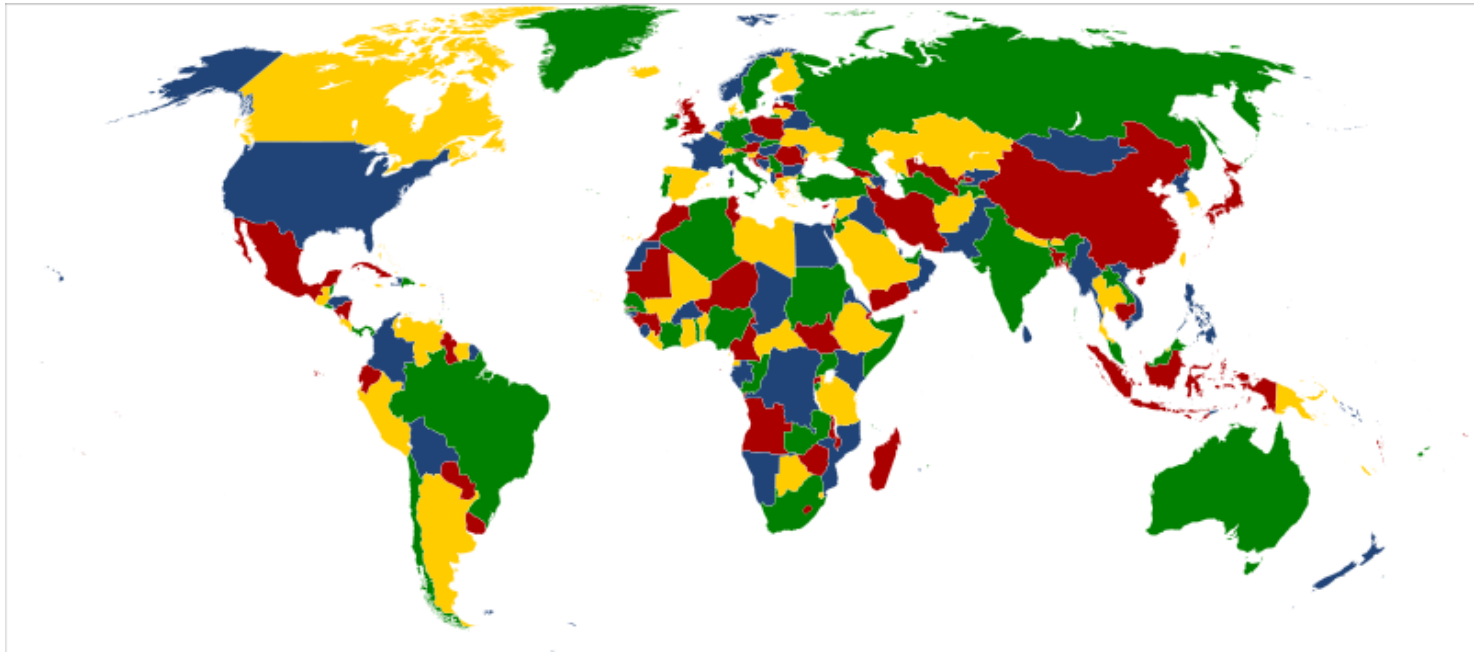
- A method for representing solutions as chromosomes (or string of characters)
- A way to calculate the fitness of a solution
  - Exploration vs. exploitation
- A selection method to choose parents
  - Tournament, Truncation, Fitness Proportional selection
- A way to generate offspring by breeding the parents
  - Crossover and mutation
- A way to select next generation
  - Elitism, Tournament, Niching, Fitness sharing

One generation

Select, Produce, Repeat! => *Until stopping criteria is met*

Four Color Theorem: Map Colouring Problem

- No more than four colors are required to color the regions of the map so that no two adjacent regions have the same color.



- We will formulate three-Color-Problem using GA

Image Credit: https://brilliant.org/wiki/four-color-theorem/

Encoding Solutions

- Three colours: $\{\text{black}\ (b), \text{dark}\ (d), \text{light}(l)\}$
- We assign a fixed order to region
- For a six-region map, a solution looks like

$$\alpha = \{bdblbb\}$$

Fitness Function

- A negative point for every two adjacent regions having the same color
  - Fitness value can be negative

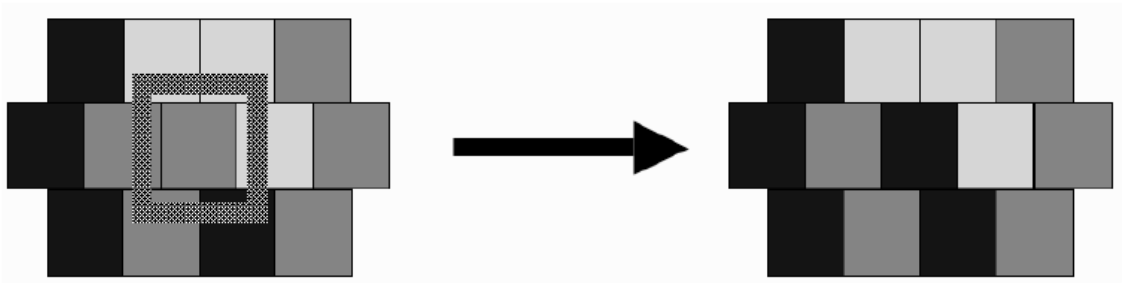
$$\frac{\exp(sF^{\alpha})}{\sum_{\alpha'} \exp(sF^{\alpha'})}$$

- Or,

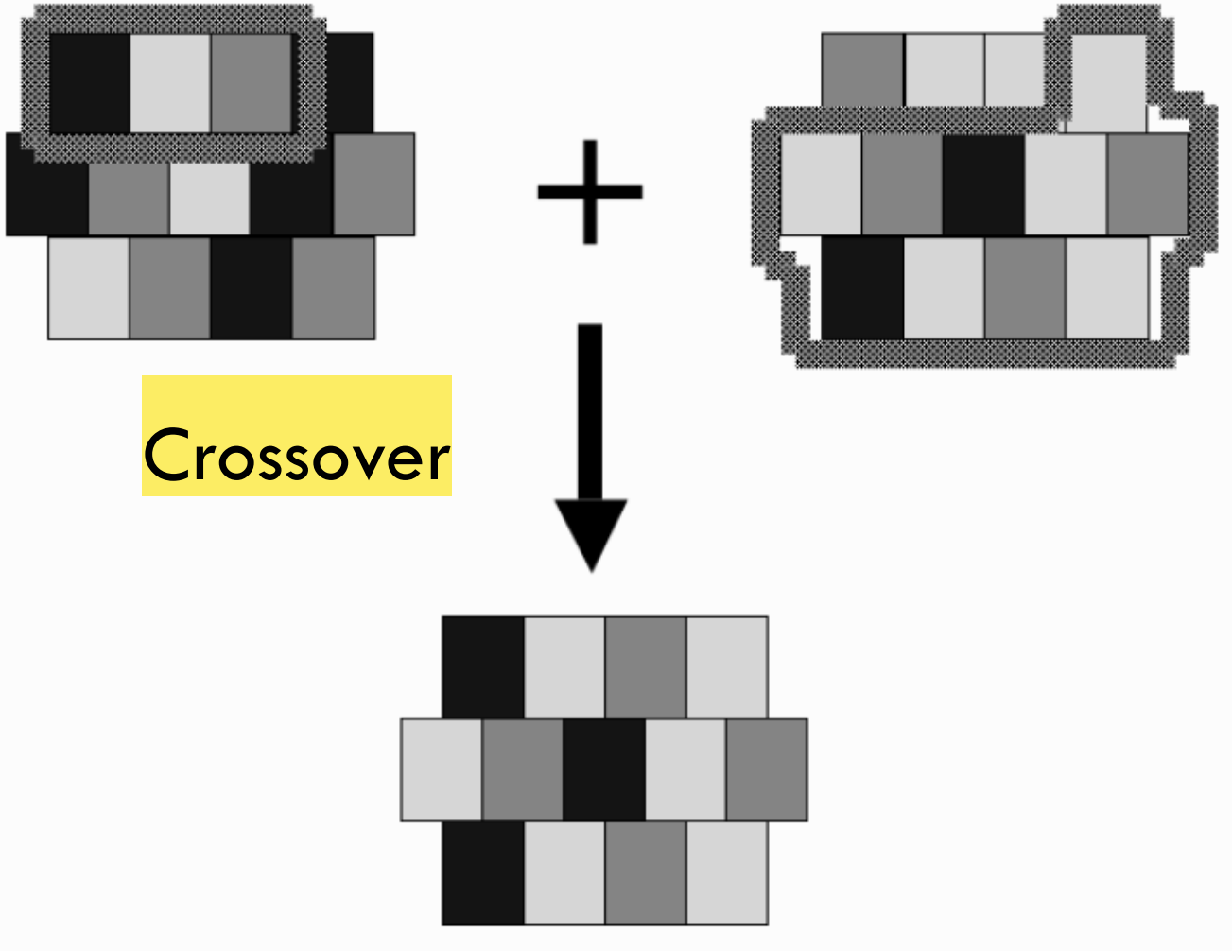
$$F^{\alpha} = \#\text{Toal Boundaries} - \#\text{Adjacent regions having same color}$$

No negative fitness value

Genetic Operators



Mutation



Crossover

Limitations of GA

- Can get stuck to a local minima for a very long time
- Like a black box:
  - We don't know how error landscape looks like and how it is working
  - No guarantee to converge

Training Neural Networks with GA

- Encode weights as strings
- Fitness function: sum-of-square errors
- Reasonably good results.

- Problems:
  - Local error information at an o/p node is lost – the entire error is clubbed into one number
  - Not using gradient information

Training a neural network with a genetic algorithm involves using a form of evolutionary optimization to tune the weights and biases of the neural network. The process involves generating a population of candidate solutions, where each candidate solution represents a set of weights and biases for the neural network.

Types of Encoding in GA

- Binary Encoding: a string of 0s and 1s
  - E.g.: Knapsack problem

- **Permutation Encoding**: string of numbers representing a sequence
  - E.g.: Sorting a sequence of numbers traveling salesman problem

4	2	5	9	0	7	8	1	3	6

  - How to do crossover?

4	2	5	9	0	7	8	1	3	6

+

8	1	9	4	2	5	6	3	0	7

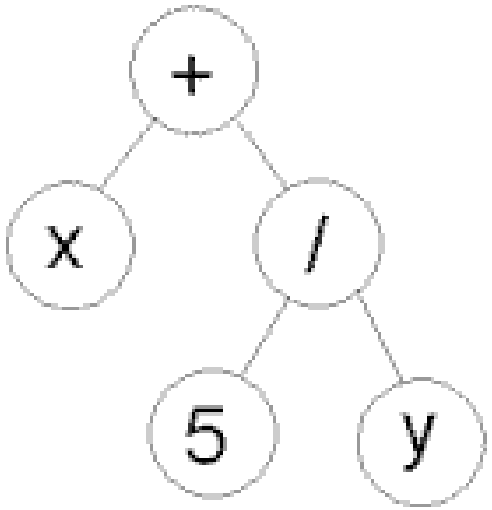
||

4	2	5	9	0	8	1	6	3	7

  - How to mutate?

- Value Encoding: a string of real numbers
  - E.g.: Training weights in neural networks

- Tree Encoding: Each chromosome is a tree
  - E.g.: Given input and output values, find a function, which will give the best (closest to wanted) output to all inputs.



(+ x (/ 5 y))