Algorithms and their Applications CS2004 (2020-2021)

Dr Mahir Arzoky

13.1 An Introduction to Genetic Algorithms

CodeRunner Class Tests and Laboratory Sessions…

- ❑ Class Test CRI: 228 attempts
- ❑ Class Test CRII: 144 attempts
- ❑ Class Test CRIII: 80 attempts
- ❑ Class Test CRIV: will be released next week!
- ❑ **All four class tests must be passed to pass Task #1.**
- ❑ **Task #1 weighs 30% of the coursework**
- ❑ **But, if you do not pass Task #1 you will be capped at D- grade (coursework).**
- ❑ **Class tests needs to be completed by 16/02/2021**

Previously On CS2004... – Part 1

- Traditional and foundational parts of algorithms:
  - Concepts of Computation and Algorithms
  - Comparing algorithms
  - Some mathematical foundation
  - The Big-Oh notation
  - Computational Complexity
  - Data Structures
  - Sorting Algorithms
  - Graphs and Graph Algorithms
- We then moved focus to Heuristic Search Algorithms:
  - Concepts
    - Fitness
    - Representation
    - Search Space
  - Methods
    - Hill Climbing
    - Stochastic Hill Climbing
    - Random Restart Hill Climbing
    - Simulated Annealing
    - Tabu Search
    - Iterated Local Search

Previously On CS2004... – Part 2

- For the next three lectures:
  - We are going to look at some more esoteric Heuristic Search Algorithms
    - Evolutionary Algorithms, e.g. Genetic Algorithms and Evolutionary Programming
    - Swarm Algorithms i.e. Ant Colony Optimisation and Particle Swarm Optimisation

Genetic Algorithms

- **Genetic Algorithm** (GA) is a powerful tool
- They can perform numerical optimisation and AI search
- Inspired by evolutionary biology…
- GAs can help in areas where there seems to be no solution
- GAs can usually find a partial answer
  - Other methods may well do better!

Are GAs Controversial?

- ❑ Evolution Theory is controversial
- ❑ GA takes ideas from biological evolution
- ❑ This is **NOT** a lecture on Evolution
- ❑ But we need to understand the basic concepts of **Evolution** to understand Genetic Algorithms…

Biological Evolution – Part 1

- Genetic Algorithms "mimic" evolution
- Evolution is the change of a **gene** pool over time
- A gene is a biological hereditary unit that is passed on (usually unaltered) for many generations
- Genes are contained within the nucleus of a cell, within **Chromosomes**
- Most organisms have multiple chromosomes

Biological Evolution – Part 2

- The **Gene Pool** is the set of all genes for a species
- Evolutionary theory states
  - "That if the environment changes, the Gene Pool must change for survival"
- This process is called **adaptation**
- This is **apparently** happening all of the time

The Process

- Genes **mutate** through random change
- Individuals are selected/survive, through **Natural Selection**
- **Populations** evolve and breed through **recombination**
- Charles Darwin developed the basic idea in 1859
- The subject has advanced a lot since then…

History of Genetic Algorithms

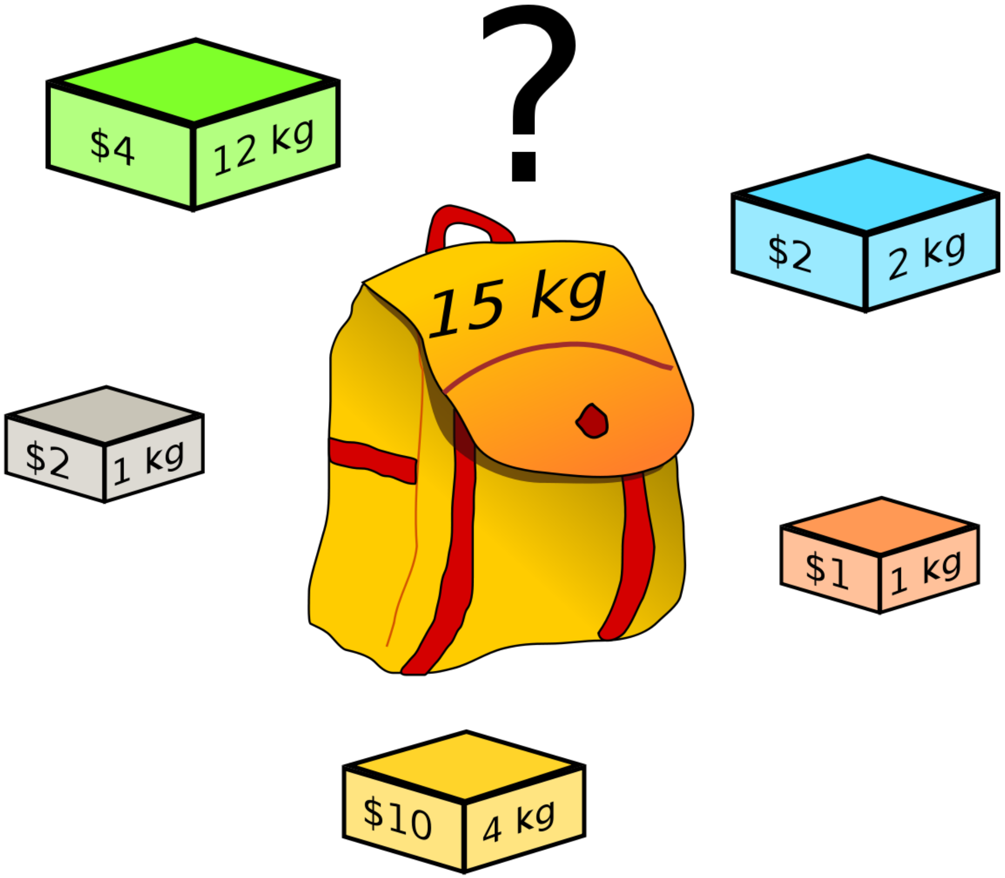
- Developed by John Henry Holland
  - February 2nd 1929 to August 9th 2015
  - In the early 1970's
  - MIT, IBM, Michigan
- He was one of the first PhD students in computer science
- We will look at Holland's original GA and then look at some of the advances

Genes and Chromosomes

- The technique uses biological metaphors
  - Each **gene** is a binary digit
  - A **chromosome** is a single string of genes
- A **solution** to a problem is encoded as a Chromosome
  - The encoding is called the **representation**
  - It must cover the whole **search space**
- A **Fitness Function** is needed to rate how good a solution a chromosome represents
- We should be **very** familiar by now with these concepts.....

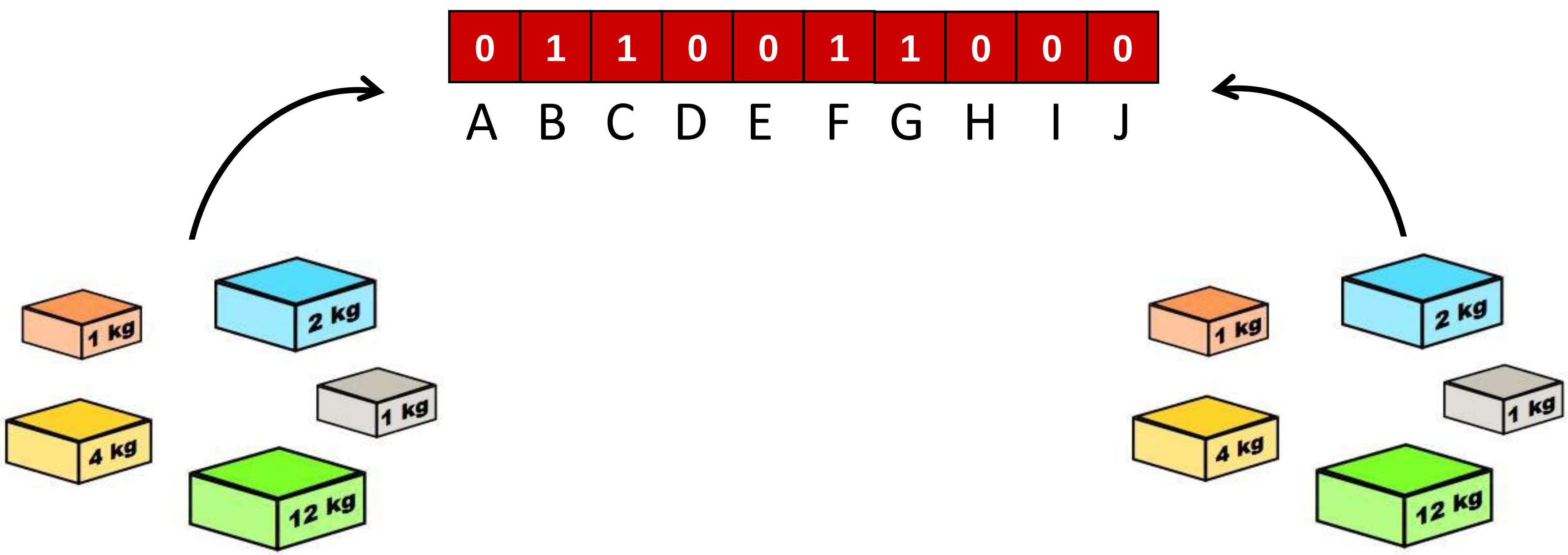
An Example: Knapsack Problem

Given *n* items, each with a **weight** and a **value**, determine the items to include so that the *total weight is less than or equal to a given limit* and the *total value is as large as possible*



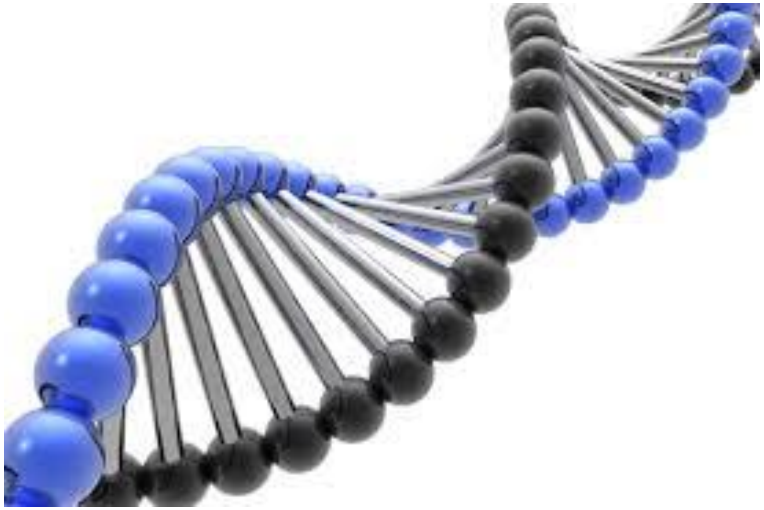
Chromosome Example

- We could solve this with a Genetic Algorithm
- The representation could be as follows:
- (Note there may be **invalid chromosomes**)



Population and Generation

- The **population** is the number of chromosomes "alive" at any one time
- The term **generations** is the number of times breeding has occurred



Genetic Algorithm Overview

- ❑ Create a population of random chromosomes (solutions)
- ❑ Each chromosome in the population is scored using a fitness function
- ❑ Create a new generation through genetic operators called **selection**, **crossover** and **mutation**
- ❑ Repeat until done – best solution to the problem!

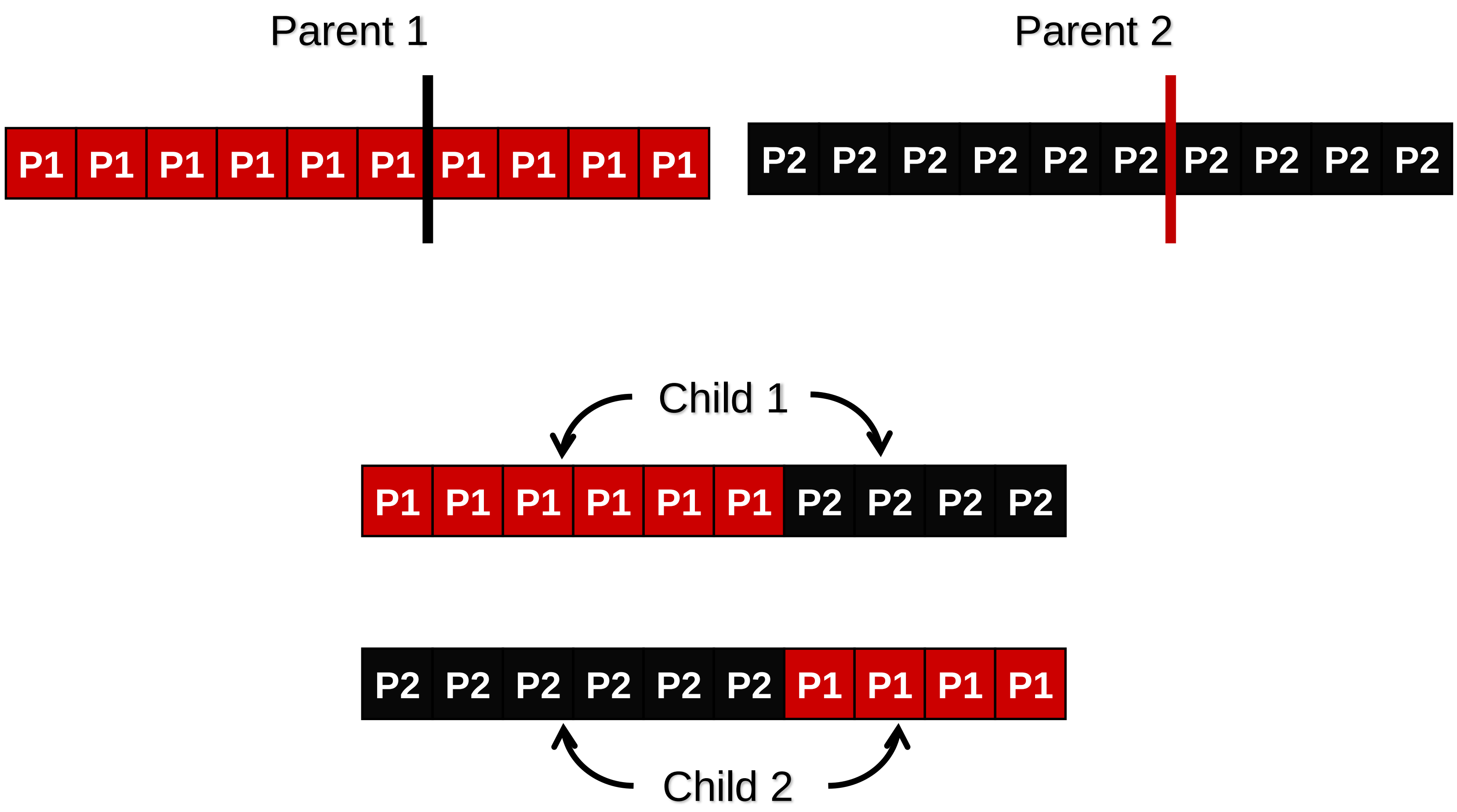
Crossover

- This is analogous to recombination or breeding
- Typically genetic material from two parents are combined to create **children**
- Various crossover operators:
  - One-point crossover
  - Uniform crossover

Crossover – One Point

- Chromosomes (with n genes) move to the crossover pool with CP chance
- Each are randomly paired up (A and B)
- Two children are created (C and D)
- A random number p between 2 and n-1 is generated for each parent pair
  - $1..p$ of D become $1..p$ of A
  - p+$1..n$ of C become p+$1..n$ of A
  - $1..p$ of C becomes $1..p$ of B
  - p+$1..n$ of D become p+$1..n$ of B
- Parents and children go back to population

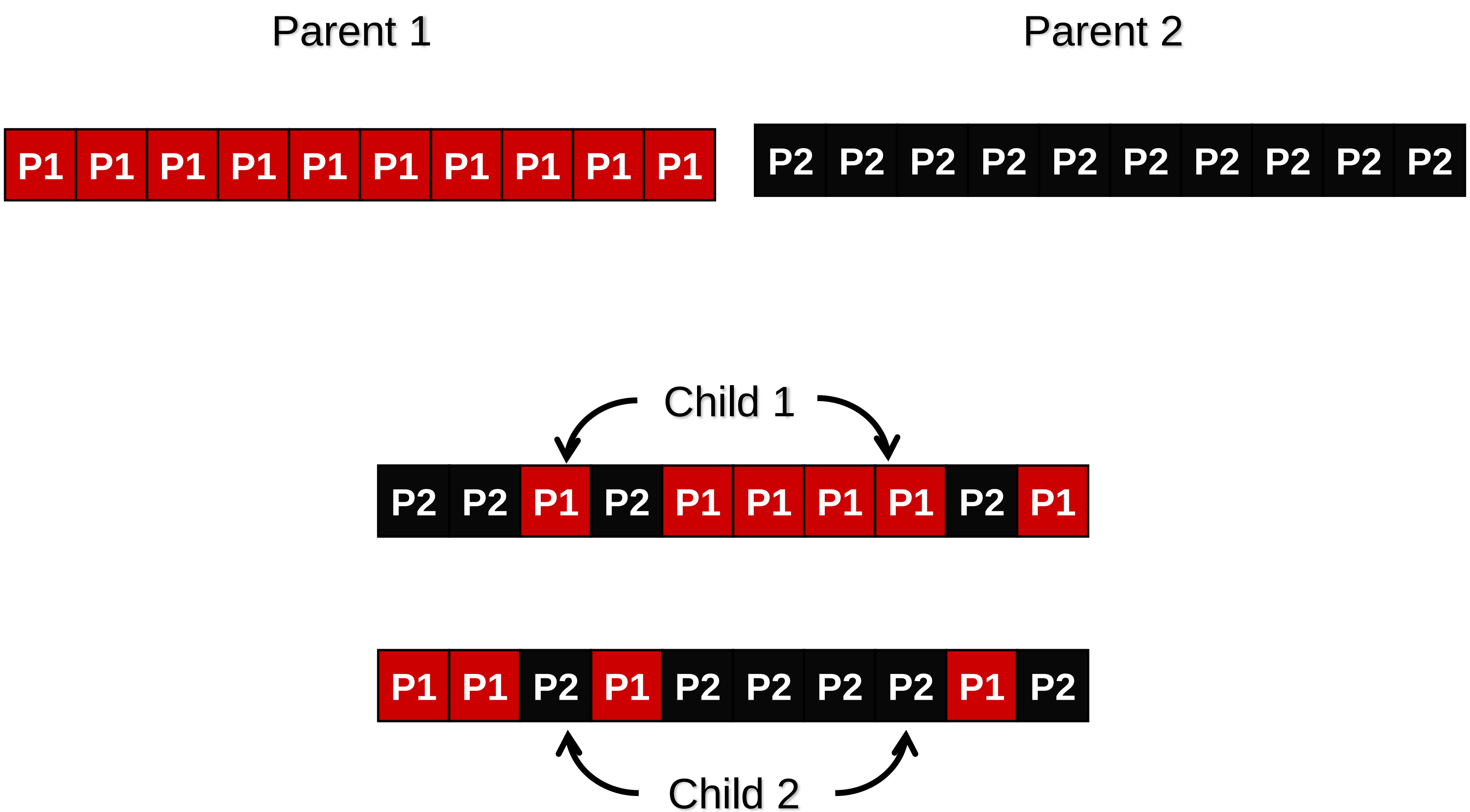
One Point Crossover Example



Crossover – Uniform

- Uniform crossover is a more powerful extension
- For each gene, there is a 50% chance that child C gets the gene from parent A and a 50% chance that it is from parent B
- Child D gets the gene that child C does not

Uniform Crossover Example



Mutation

- This is analogous to biological mutation
- Small random tweak of the gene (in the chromosome), to get a new solution
- Mutation allows the genetic algorithm to explore more of the search space and avoid falling into local minima

Example Mutation Operator

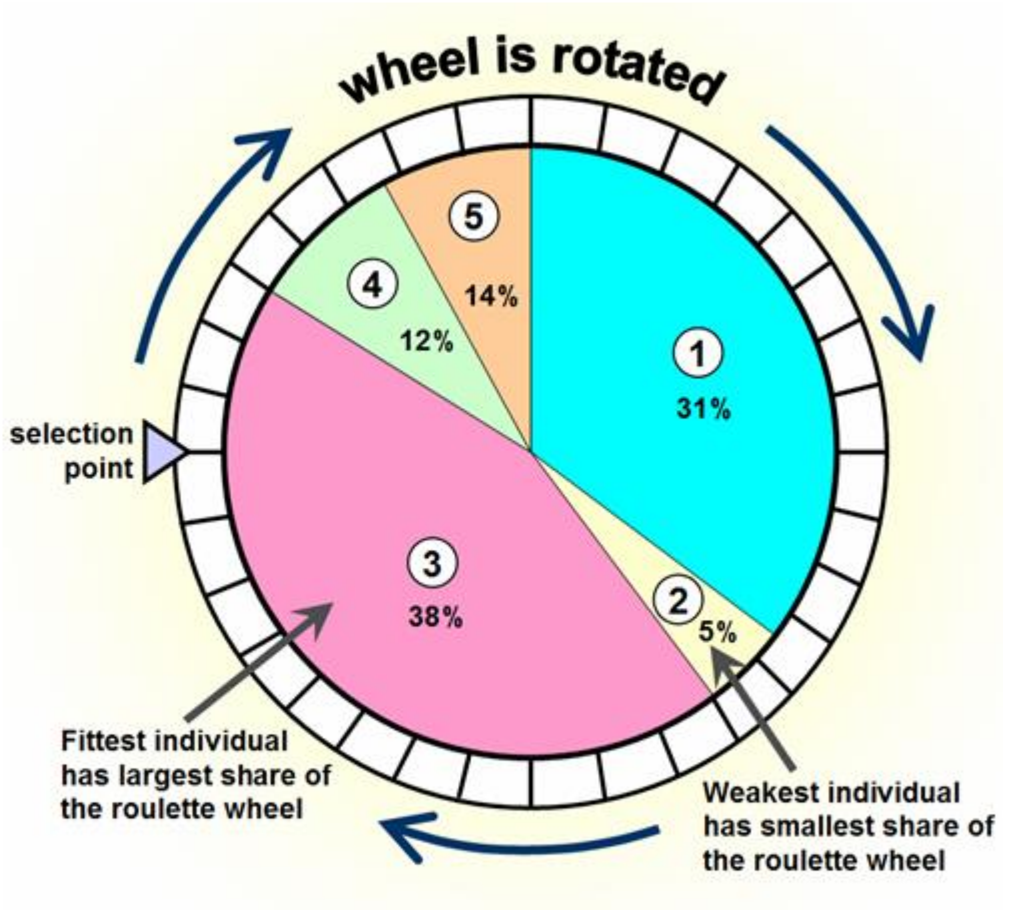
- Each bit (gene) of a chromosome is given a chance (probability) MP of inverting
  - A ‘1’ becomes a ‘0’
  - A ‘0’ becomes a ‘1’

$01101101 \longrightarrow 0\underline{0}10111\underline{1}1$

(These ones)

Selection operator: Roulette

- ❑ Aim to retain the best performing chromosomes (solutions) from one generation to the next
- ❑ Forming new population
    - ❑ Equal in size to the original
- ❑ The chance of a chromosome surviving is proportional to it's fitness vs. the total of the others
- ❑ Survival of the Fittest via a biased Roulette Wheel!
- ❑ There are many other types



GAs - Parameters

NG	Number of Generations
PS	Population Size
CP	Crossover Probability
MP	Mutation Probability
n	The number of bits (genes) making up each Chromosome

Holland’s Algorithm

```
Input: The GA parameters: NG, PS, CP, MP and n
       The Fitness Function
1) Generate PS random Chromosomes of length n
2) For i = 1 to NG
3)   Crossover Population, with chance CP per Chromosome
4)   Mutate all the Population, with chance MP per gene
5)   Kill off (or fix) all Invalid Chromosomes
6)   Survival of Fittest, e.g. Roulette Wheel
7) End For
Output: The best solution to the problem is the Chromosome
        in the last generation (the NGth population) which
        has the best fitness value
```

Parameters

- Population size
  - [10,100] depending on the problem
- Generations
  - [100,1000] depending on the problem
- Chromosome size
  - Dependent on problem
  - As small as possible (not too small)
- Mutation rate: 0.1-10% ($1/n$)
- Crossover rate: 50%-100%

Where are they used?

- ❑ Search space is irregular
- ❑ Search space is very large
- ❑ Fitness function is noisy
- ❑ Task does not require an exact global maximum, just a good fast approximation
- ❑ No other method can help

GAs - Applications

- Optimisation
- Economics
- Parallelisation
- Image processing
- Vehicle routing problems
- Design of aircraft
- Scheduling applications
- DNA analysis
- Etc…

The Laboratory

- The laboratory will involve applying a GA to the Scales problem

Next Lecture

- We will look at using a GA to solve an example problem
- We will also look at other aspects of Evolutionary Computation