University of South Australia

COMP 2019

Week 5

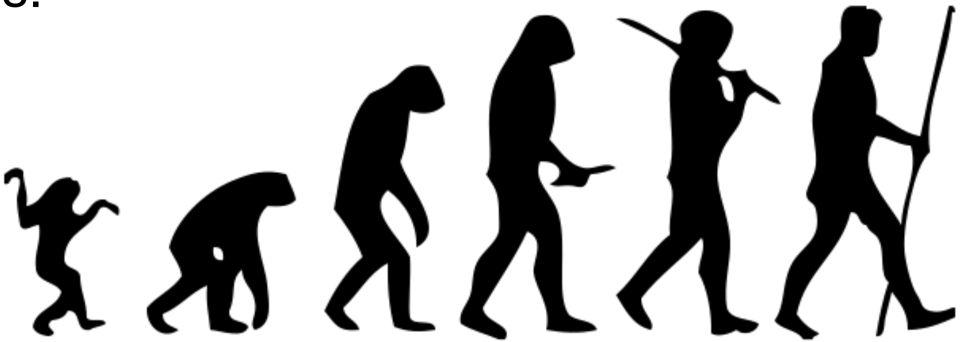
Evolutionary Algorithms

Learning Objectives

- Explain how evolutionary algorithms work (CO2)

Evolutionary Motivation

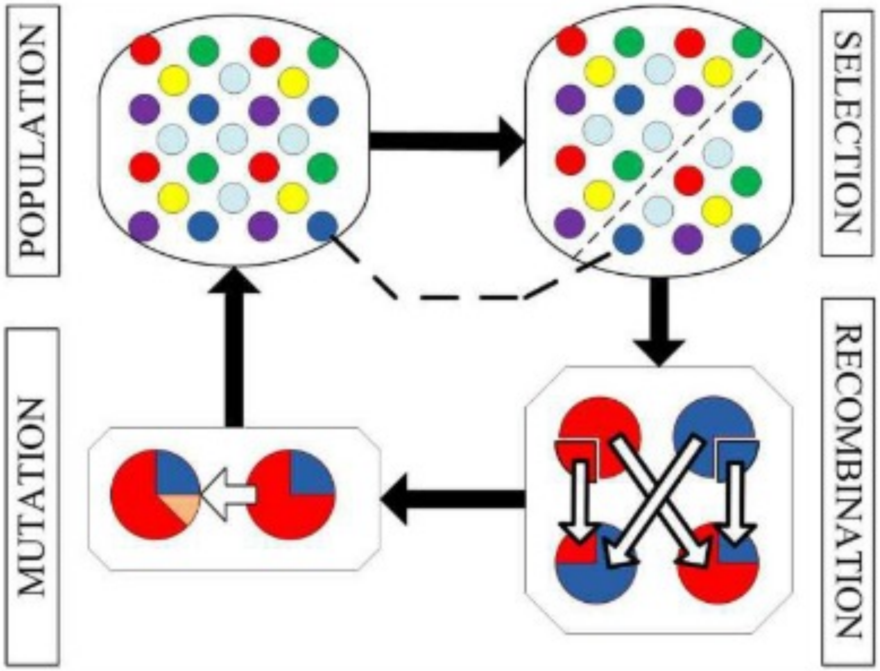
- Biological inspiration:
  - iterative optimisation by competition among a population of evolving candidate solutions.
  - competition isolates essential properties of good solutions encoded in building blocks.



https://commons.wikimedia.org/wiki/File:Human_evolution_scheme.svg

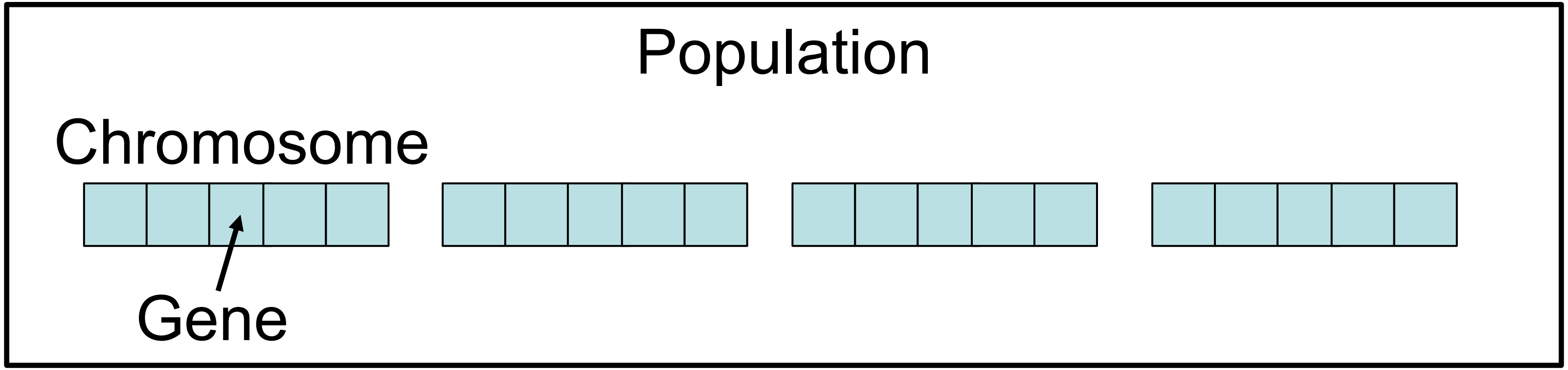
EA Approach

- Keep a pool of prototypical individuals (“population”)
- Select individuals to compete against each other
- Winner is more likely to contribute to next generation
- Performance is measured using a fitness function
- Obtain successor generation by recombining parts of the individuals that performed well



Source: http://www.engineering.lancs.ac.uk

Population Representation



Genetic Algorithm

```
population ← Initial_population
loop
    new_population ← ∅
    repeat Size(population) times
        p1 ← Select(population,EvalFn)
        p2 ← Select(population,EvalFn)
        child ← Reproduce(p1, p2)
        if (small probability) then child ← Mutate(child)
        new_population ← new_population ∪ {child}
    end repeat
    population ← new_population
until (fit enough individual found) or (time limit reached)
return best individual in population
```

GA for State Space Search

- Represented as individuals in the population
- Encode state as features that describe potential solutions
  - Bit strings
  - Sequences of integers
  - Permutations
- Similar to chromosomes that encode characteristics of individuals

Fitness

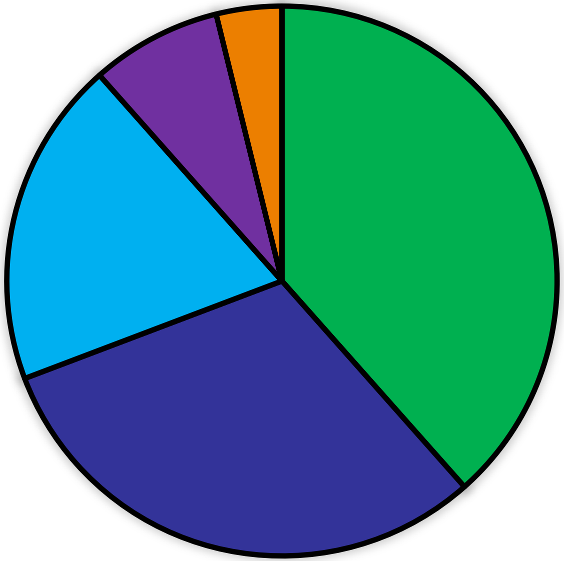
- Determines how good a solution encoded in a chromosome is
  - Decode the representation to obtain a potential solution
  - Return a (normalised) number describing the quality of the solution
  - Penalise poor and invalid solutions

Initial Population and Stopping Criteria

- Initial population
  - Size (100-100k)
  - Distribution (random or seeded)
- Stopping criterion
  - An individual satisfies all desired criteria
  - Fixed number of iterations
  - Time limit
  - Manual inspection

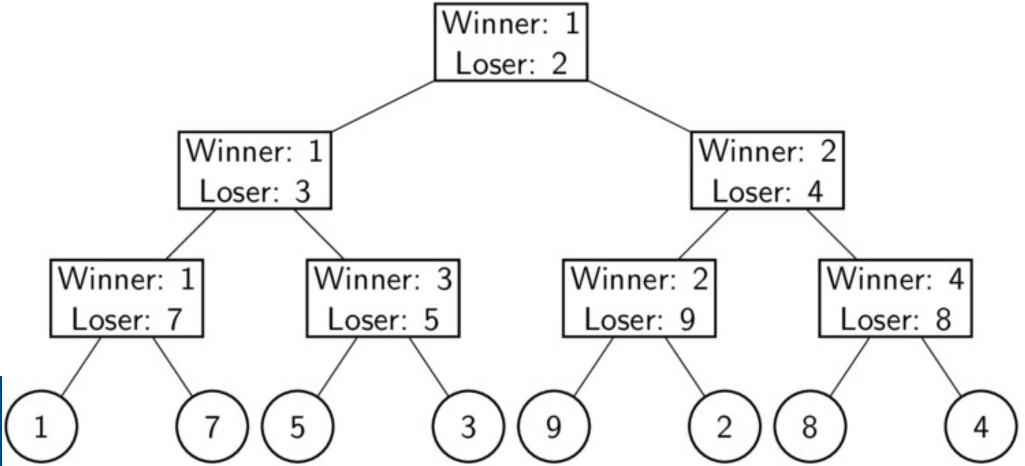
Roulette Wheel Selection

- Fitness proportionate selection
  - Probability for selection is proportional to each individual's fitness



Tournament Selection

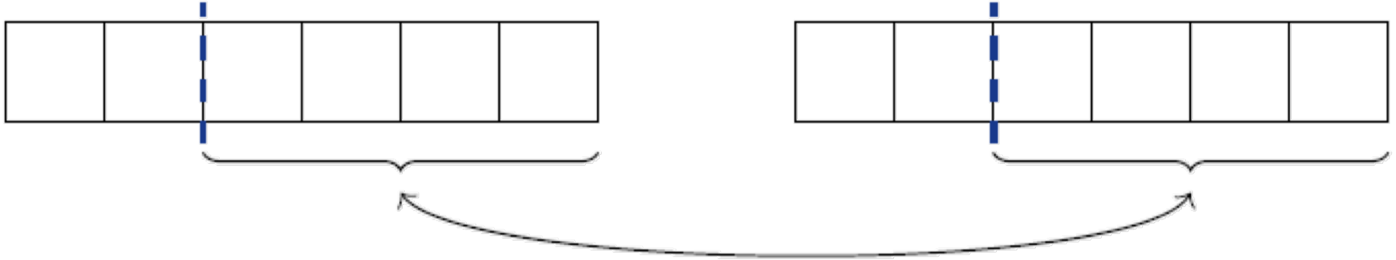
- Randomly select *n* individuals from population
- Sort according to decreasing fitness
- Choose *k*th individual with probability $p(1-p)^{k-1}$



https://commons.wikimedia.org/wiki/File:Tournament_tree.png

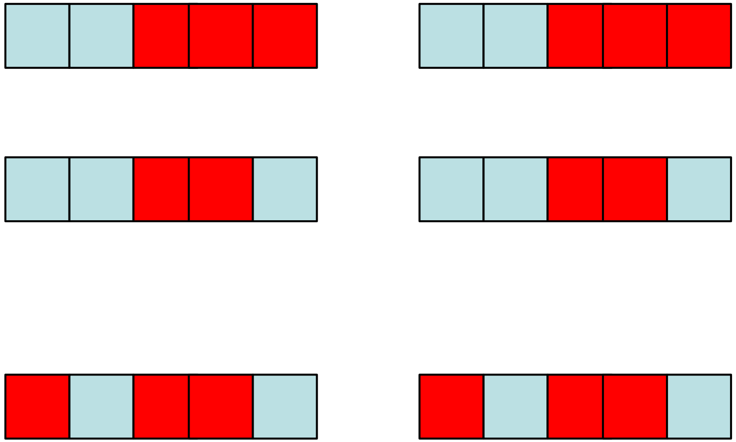
Crossover

- Combine the genes from two selected individuals to form a new individual ("offspring")
  - Swap components of the chromosome, reorder parts, etc
  - Strongly tied to the specific problem and its encoding as chromosomes



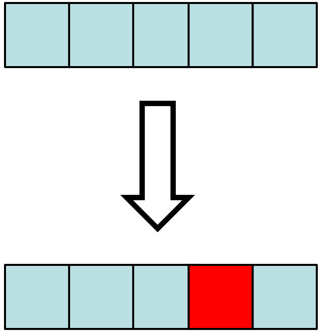
Crossover Operators

- 1 point crossover
- 2 point crossover
- Ordered crossover
- Uniform crossover
- …



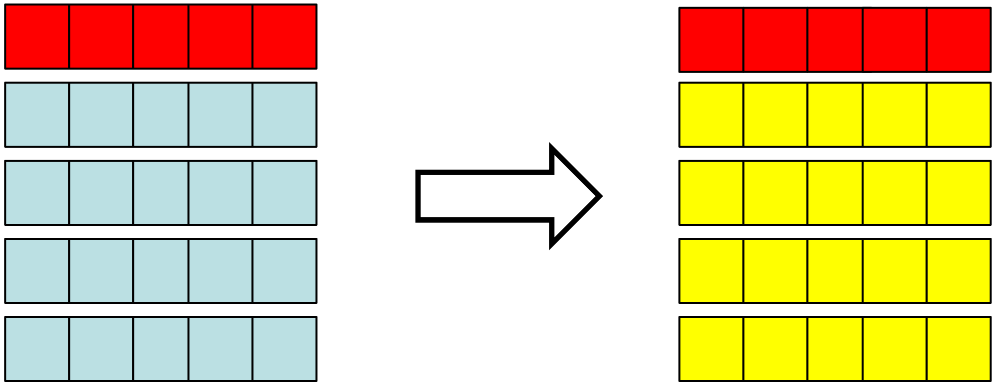
Mutation

- Take a single candidate and randomly change it
- Should have low probability



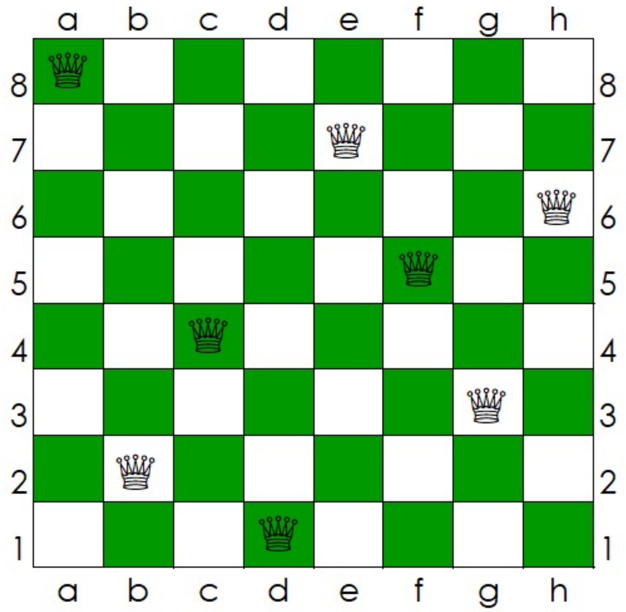
Elitism

- Carry over the best individual(s) from the current population to the next generation



8-Queens Encoding

8	2	4	1	7	5	3	6



Fitness:

28 – (number of pairs attacking each other)

8-Queens Crossover & Mutation

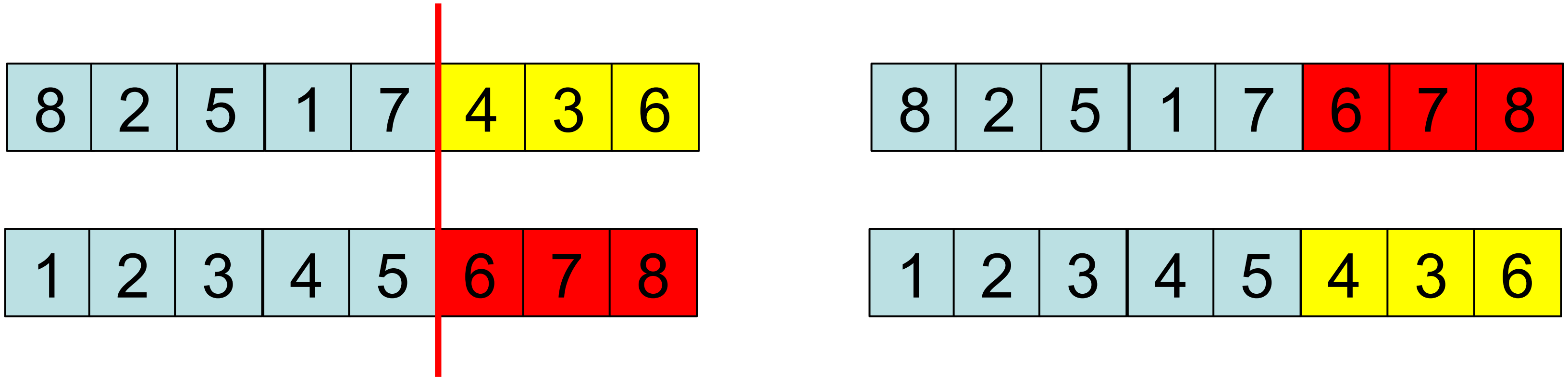
- 1 point crossover

8	2	5	1	7	4	3	6
1	2	3	4	5	6	7	8

8	2	5	1	7	6	7	8
1	2	3	4	5	4	3	6

8-Queens Crossover & Mutation

- 1 point crossover



8-Queens Mutation

- Change a position randomly

8	2	5	1	7	4	3	6

⇩

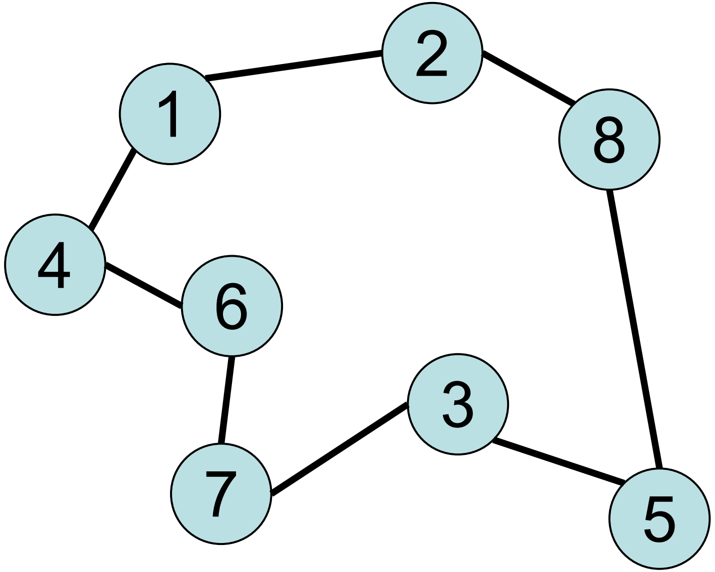
8	2	4	1	7	4	3	6

Traveling Salesman Example (TSP)

Chromosome:
Sequence of visited cities

1	2	8	5	3	7	6	4

Fitness:
Total Distance Travelled



TSP Crossover

- Ordered Crossover:
  - Pick a sequence S_A in chromosome A
  - Copy in order chromosome B, skipping any elements in S_A
  - Insert S_A at the same position in the copy of B

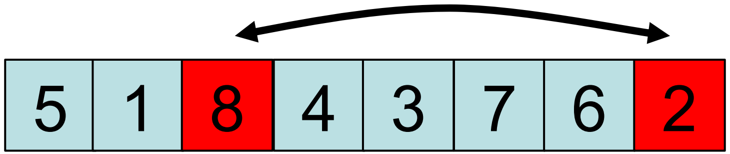
1	2	8	5	3	7	6	4

5	3	1	6	8	4	2	7

5	1	8	4	3	7	6	2

TSP Mutation

- Swap two cities in the sequence at random



Learning to Walk

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

GA and Machine Learning

https://www.youtube.com/watch?v=qv6UVOQ0F44

Summary

- Can solve very complex problems efficiently.
- Typically used for problems other search algorithms and mathematical optimisation theory cannot handle
- Idea is to distil building blocks that combine into good solutions
- Choice of parameters, representation, operators and probabilities is crucial.
- Optimality and completeness are <u>not</u> guaranteed, but results are often close to optimum.
- Anytime algorithm: can interrupt and ask for a solution at any time.

University of South Australia

Questions?