

# Introduction to Artificial Intelligence



VICTORIA UNIVERSITY OF  
**WELLINGTON**  
TE HERENGA WAKA

**COMP307/AIML420**

## **Evolutionary Computation 1: Evolutionary Computation and Learning**

Dr Andrew Lensen

[Andrew.Lensen@vuw.ac.nz](mailto:Andrew.Lensen@vuw.ac.nz)

# Outline

- Why evolutionary computation (EC) and learning?
- What is EC?
- EC Techniques
- Key characteristics and design questions
- Genetic algorithms: representation, selection and genetic operators
- Overview of other evolutionary algorithms

# Why Do We Need Evolutionary Computation?

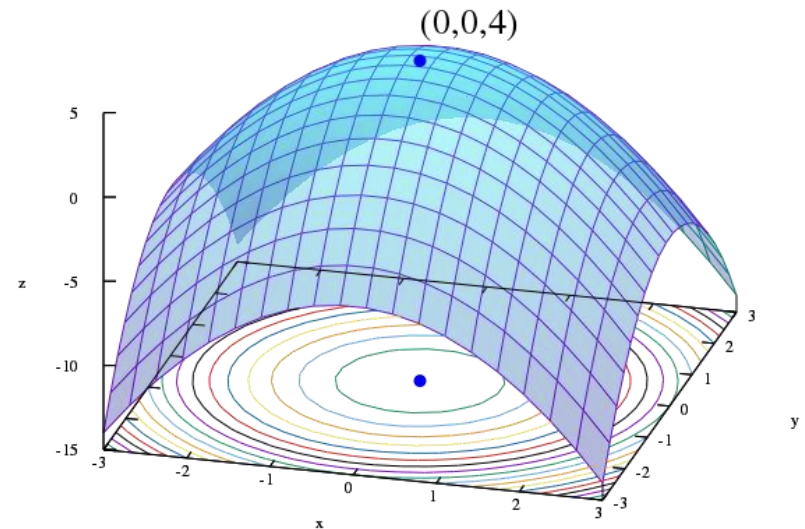
- We have discussed several methods and algorithms in ML
- But they have limitations:
  - Local optima
  - Unreasonable assumptions
  - Needs to predefine/fix the structure/model of the solution, and only learns the parameters/coefficients
  - Many parameters to learn (high-dimensional optimisation)
- Evolutionary Computation (EC) is one technique that can avoid some of the problems

# Evolutionary Computation and Learning

- In **computer science**, **evolutionary computation** is a family of “*nature inspired*” AI algorithms for **global optimisation**.
- In **technical terminology**, they are a family of **population-based** trial-and-error problem solvers with a metaheuristic or **stochastic** optimisation character.
- **Evolutionary Learning** is the use of **evolutionary computation** methods for tackling **machine learning** tasks
- *(Shameless) Source:*  
[https://en.wikipedia.org/wiki/Evolutionary\\_computation](https://en.wikipedia.org/wiki/Evolutionary_computation)

# What is Optimisation?

- In an optimization problem, we are trying to find the best **values of the variables** that gives the **optimal value** of the **function** that we are optimising.
- E.g. minimise fuel use of courier deliveries.
- **Decision variable(s)**
- **Objective function(s)**
- **Constraint(s)**
- ...



# Examples

- In machine learning
  - Optimise the **weights of a neural network**
  - Optimise the **architecture** (#layers, #nodes) of **a neural network**
  - Optimise the **distance measure for k-NN classifier**
  - Optimise the **distance measure for clustering methods**
  - **Feature selection** (select a subset of important features to use)
- Other domains
  - Design the **shape of a racing car/plane wings**
  - **Schedule** lecture rooms (timetabling)
  - **Schedule** jobs in cloud network
  - **Schedule** trucks for delivery

# Evolutionary Computation: Origin Story

- In the **1950s**, long before computers were widely used, the idea to use *Darwinian* principles for automatic problem solving was first suggested.
- **Three different interpretations** of this idea were developed independently:
  - **Evolutionary programming**: Lawrence Fogel (USA)
  - **Evolutionary strategies**: Ingo Rechenberg (Germany)
  - **Genetic algorithms**: John Holland (USA)
- These areas developed separately for over 15 or 20 years
- Since the early **1990s**, they have been seen as different representatives of one technology: **evolutionary computation**

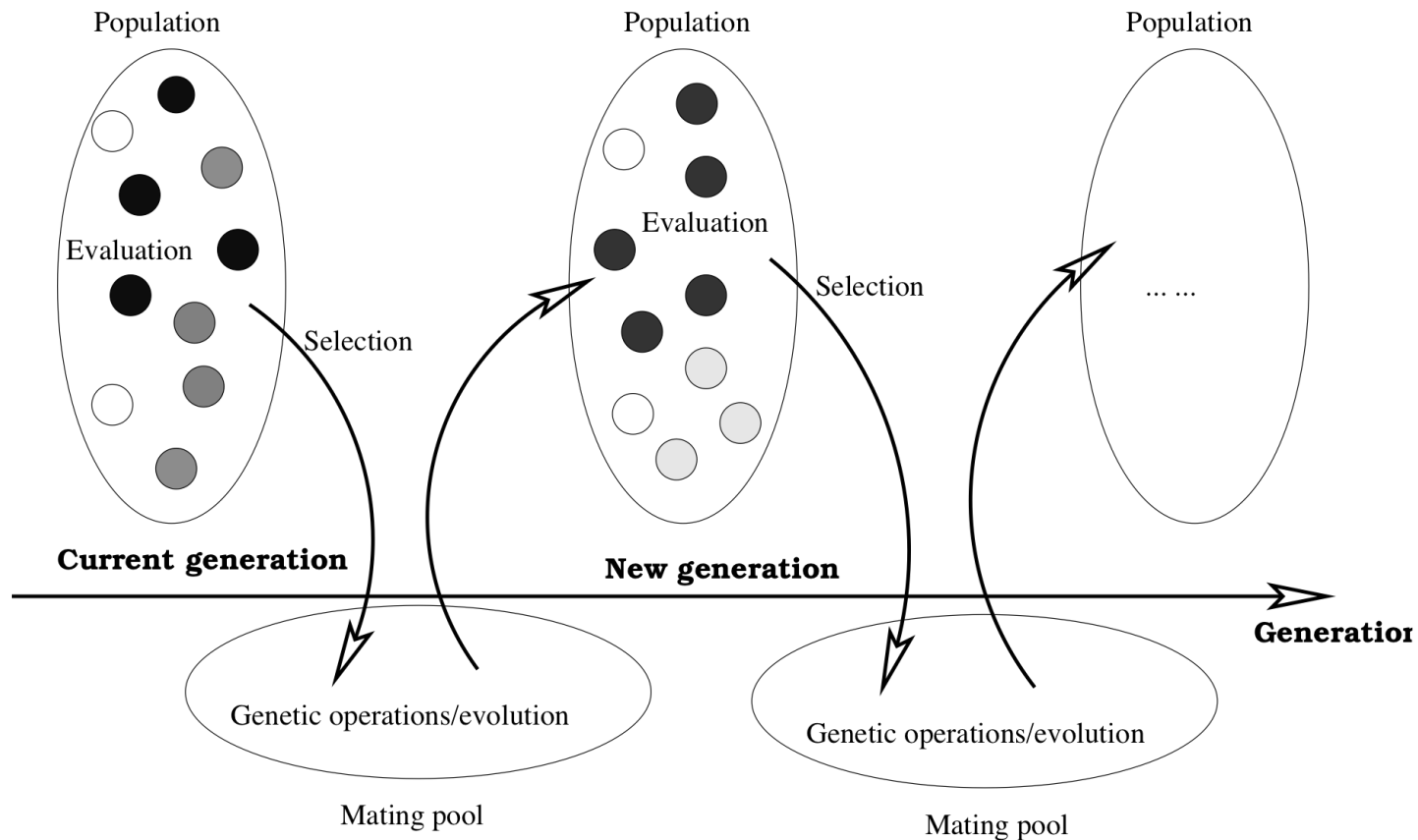
# EC Techniques

- Evolutionary algorithms (EAs)
  - Genetic algorithms (the biggest branch)
  - Evolutionary programming
  - Evolutionary strategies
  - Genetic Programming (Koza, 1990s, fast growing area)
- Swarm intelligence (SI)
  - Ant colony optimisation
  - Particle swarm optimisation (PSO)
  - Artificial immune systems
- Other techniques
  - Differential evolution
  - Estimation of distribution algorithms
  - ...



# Evolutionary Algorithms

- Search for the **best individual** by **evolving a *population*** with **reproduction** (e.g. crossover, mutation)



# Key Characteristics

- One (or more) *populations of individuals*
- Dynamically changing populations due to the *birth and death of individuals* (through *crossover, mutation, ...*)
- A *fitness function* which reflects the ability of an individual to survive and reproduce (“*survival of the fittest*”)
- *Variational inheritance*: offspring closely resemble their parents, but are not identical
- *Final solution (individual)*: the one with the best *fitness*
- *Fitness* could be accuracy, cost, error, ...

# Evolutionary Search

- Search space of **candidate solutions**
  - Not space of **partial solutions**
  - Modify **whole solutions** rather than extending partial solutions
- Genetic beam search
  - Keep track of a **set of good solutions**
  - **Not all candidate solutions**, unlike best first or A\*
  - **Not only the best candidates**, unlike in hill climbing or gradient descent
- *Combine* good candidates to construct new candidates
  - Can **modify candidates** in isolation (**mutation**)
  - Or **different candidates** can *interact* in evolution (**crossover**)

# Key Design Questions

- Representation
  - How can we **represent** individuals (solutions)?
- Evaluation
  - How can we **evaluate** individuals (**fitness function**)?
  - A **fitter** individual should have a **better objective value** (e.g. smaller error)
- Selection
  - How to **select** individuals into the mating pool (**selection** scheme)?
  - **Fitter** individuals should be **more likely to survive/reproduce**
  - **Selection pressure**
- Genetic Operators
  - How to **generate** new individuals (**crossover**, **mutation** operators)?
  - Children **inherit strong parts** of parents
  - **Maintain diversity** (jump out of local optima)
- Other **parameters**
  - population size, mating pool size, stopping criteria, ...

# Individual Representation

- Problem dependent

- Binary string (e.g. feature selection)

1	0	0	1	1
---	---	---	---	---

0	0	0	1	1
---	---	---	---	---

- Continuous vector (e.g. ANN weight optimisation)

-0.73	0.10	0.35	-0.06	0.23
-------	------	------	-------	------

-0.13	0.10	0.35	-0.06	-0.29
-------	------	------	-------	-------

- Permutation (e.g. traveling salesman problem)

1	3	5	2	4
---	---	---	---	---

1	3	5	4	2
---	---	---	---	---

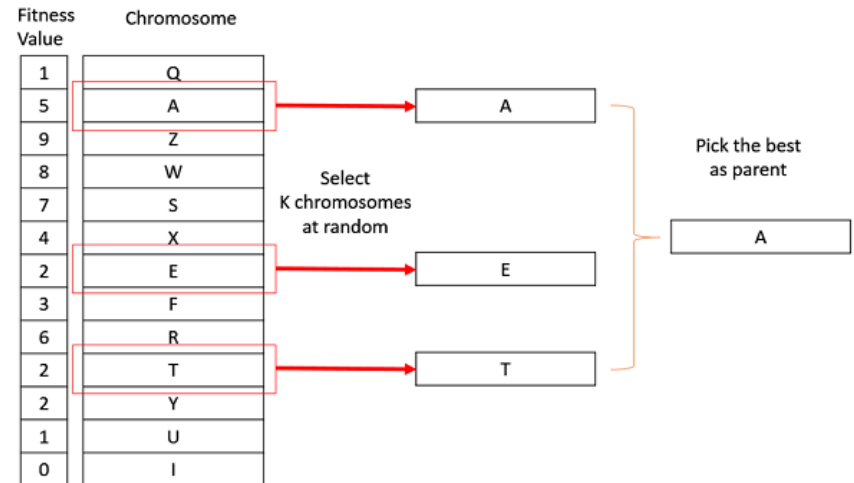
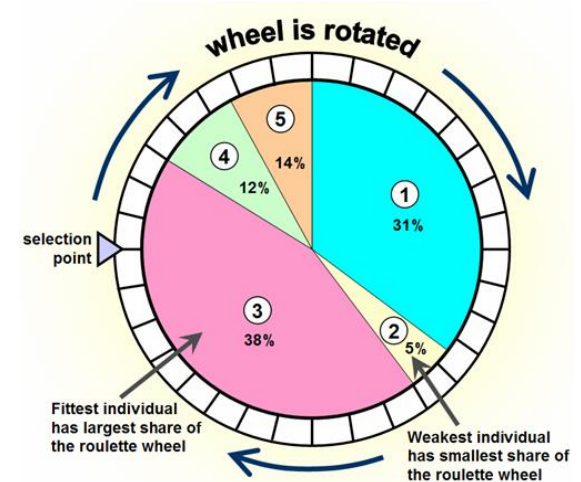
- Variable length (e.g. symbolic regression)

# Fitness Evaluation

- **Fitness function:** reflect the **quality** of individuals
  - Must correspond to **optimality** property
  - Must be **computable**
  - **Smoothness:**
    - Small changes to candidate -> small changes to quality/fitness
    - Large changes to candidate -> large changes?
- Depending on the problem, the fitness function could be:
  - the larger, the better --- **maximisation**
  - the smaller, the better --- **minimisation**

# Selection

- Uniform selection
  - Each individual has the **same** chance to be selected
- Roulette wheel selection
  - The probability of being selected is **proportional** to the fitness
  - Assume fitness is maximised
- **K**-tournament selection
- Truncate selection
  - Select the best k individuals



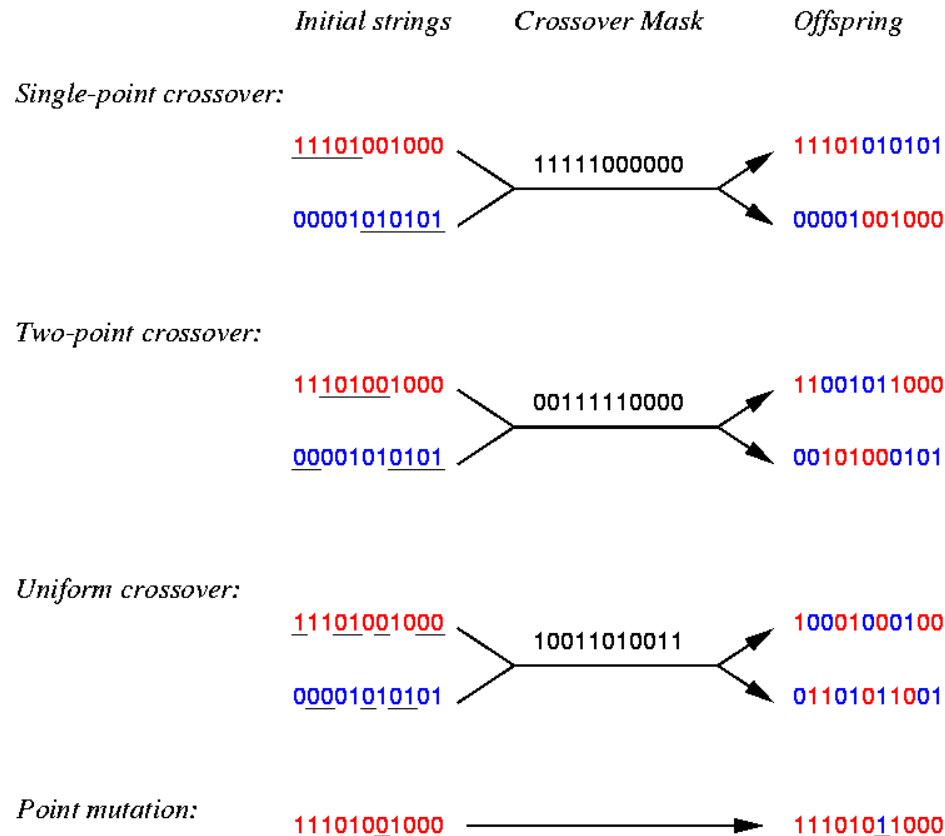
# Genetic Operators

- Depends on the problem – individual representation
  - Swap a bit of a binary vector
  - Resample an element of a continuous vector
  - Shuffle a part of a sequence
  - ...
- A representative: Genetic Algorithms



# Genetic Algorithm

- Representation: binary string
- An individual is also called a chromosome



- Other representations as well: continuous vector, permutation, ...

# A Basic Genetic Algorithm

- Randomly **initialise** a population of chromosomes
- **Repeat until** stopping criteria are met:
  - Construct an empty new population
  - **Repeat until** the new population is full:
    - **Select two parents** from the population by roulette wheel selection
    - Apply **crossover** to the two parents to generate two children
    - Each child has a probability (**mutation rate**) to undergo **mutation**
    - Put the two children into the **new population**
  - **End Repeat**
  - **Move to the new population** (new generation)
- **End Repeat**
- Output the best individual from the final population

# A Simple GA Example

- **OneMax Problem**
  - Target to (11111...1)
  - More zeros means worse: far away from the target
  - Simple “benchmark” problem!
- **Representation**: bit string
- **Fitness function**:  $1 + \sum_i x_i$  (the larger the better)
- **Crossover**: single-point crossover
- **Mutation**: point mutation
- ***Assume our algorithm does not know the problem or fitness function!***

# A Simple GA Example

- 10 bits (Optimal fitness = 11)
- population size = 20
- mutation rate = 0.25 (25%)
- Run for 10 generations

```
At generation 0 average fitness is 6.0, best fitness is 9
At generation 1 average fitness is 6.65, best fitness is 10
At generation 2 average fitness is 6.8, best fitness is 11
At generation 3 average fitness is 6.9, best fitness is 9
At generation 4 average fitness is 6.45, best fitness is 9
At generation 5 average fitness is 6.95, best fitness is 9
At generation 6 average fitness is 7.3, best fitness is 11
At generation 7 average fitness is 6.65, best fitness is 10
At generation 8 average fitness is 6.25, best fitness is 8
At generation 9 average fitness is 6.6, best fitness is 8
```

# Other Techniques

- Particle swarm optimization (PSO):
  - [http://en.wikipedia.org/wiki/Particle swarm optimization](http://en.wikipedia.org/wiki/Particle_swarm_optimization)
- Learning Classifier Systems:
  - [http://en.wikipedia.org/wiki/Learning classifier system](http://en.wikipedia.org/wiki/Learning_classifier_system)
- Ant colony optimization:
  - [http://en.wikipedia.org/wiki/Ant colony optimization](http://en.wikipedia.org/wiki/Ant_colony_optimization)
- Differential evolution:
  - [http://en.wikipedia.org/wiki/Differential evolution](http://en.wikipedia.org/wiki/Differential_evolution)
- Other useful links:
  - [http://en.wikipedia.org/wiki/Genetic Algorithm](http://en.wikipedia.org/wiki/Genetic_Algorithm)
  - [http://en.wikipedia.org/wiki/Evolution strategies](http://en.wikipedia.org/wiki/Evolution_strategies)
  - [http://en.wikipedia.org/wiki/Evolutionary programming](http://en.wikipedia.org/wiki/Evolutionary_programming)
- (Wikipedia comes from a CS background!!)

# Summary

- Evolutionary computing overview
- Main idea and process
- Representations of candidate solutions
- Selection and genetic operators
- Genetic algorithms
- Other EC algorithms and techniques
- Next lecture: Genetic programming (GP)