Evolutionary Computation and Learning

Represented by:

V. Ghasemi

Outline

- GA Introduction and for 0-1 Knapsack
- GA for TSP
- GA for Feature Selection
- GA Results Analysis

0-1 Knapsack Problem

- Which treasures to carry?
  - Maximise the value while fitting the capacity

Value= \$10K
Weight = 7kg

Value= \$25K
Weight = 20kg

Value= \$50K
Weight = 38kg

Capacity = 50kg

Value= \$20K
Weight = 6kg

Value= \$7K
Weight = 10kg

0-1 Knapsack Problem

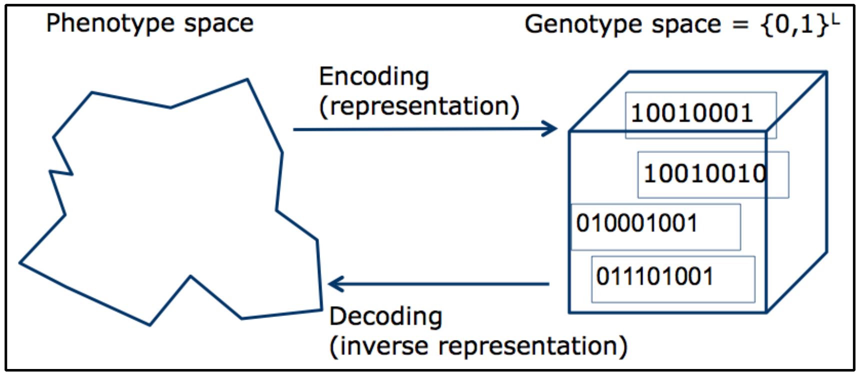
- M items to select, a knapsack with capacity Q
  - Item i has a value v_i and weight w_i
- $\max \sum_{i=1}^{M} v_i x_i = v_1 x_1 + v_2 x_2 + \cdots + v_M x_M$
- $\sum_{i=1}^{M} w_i x_i = w_1 x_1 + w_2 x_2 + \cdots + w_M x_M \leq Q$
- $x_i = 1$ if item i is selected; $x_i = 0$ if item i is not selected;

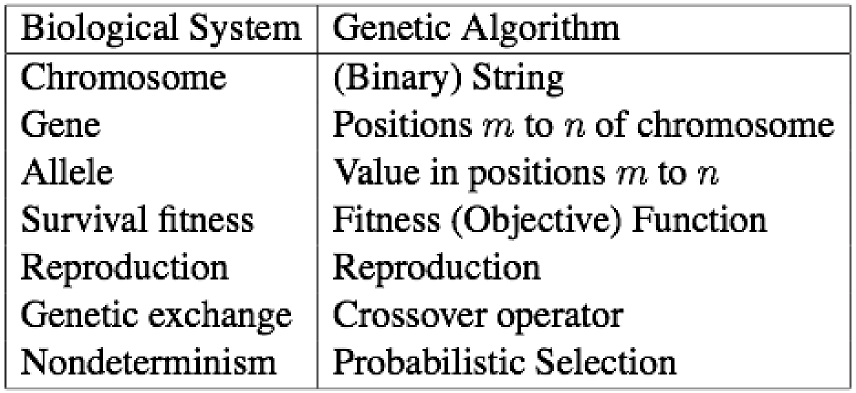
- Exhaustive search
  - 2^M possible solutions
- Known to be NP-hard problem

- Use Genetic Algorithm (GA) to solve it

Genetic Algorithm

- **Genotype**: The particular sequence of DNA in particular genes
- **Phenotype**: The physical attributes of an organism
- **Crossover**: The phenomenon whereby two genes combine by exchange of genetic material
- **Mutation**: A random change to a gene exchanging genetic material
- **Natural selection**: An adaptive mechanism of evolution ‘Survival of the fittest’
- **Fitness**: Survival capability

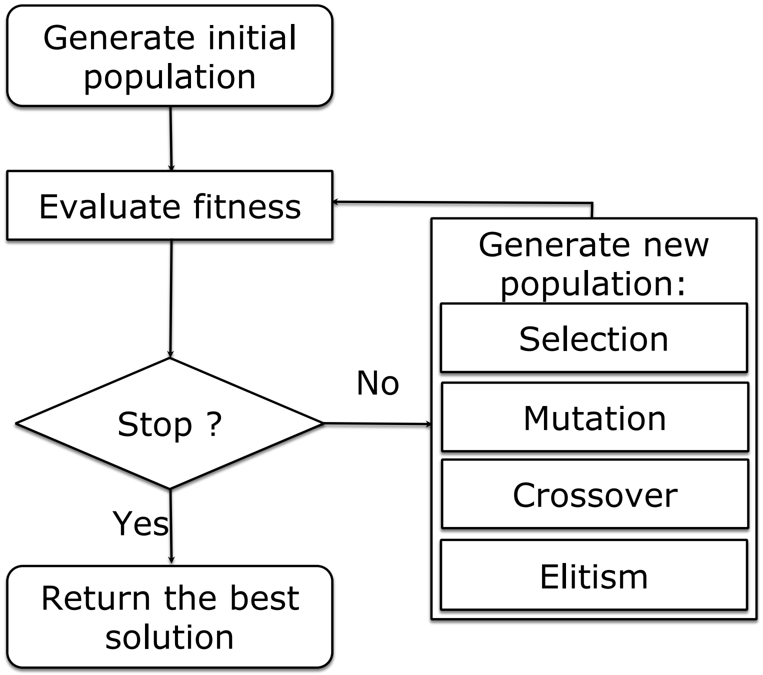




Biological System	Genetic Algorithm
Chromosome	(Binary) String
Gene	Positions m to n of chromosome
Allele	Value in positions m to n
Survival fitness	Fitness (Objective) Function
Reproduction	Reproduction
Genetic exchange	Crossover operator
Nondeterminism	Probabilistic Selection

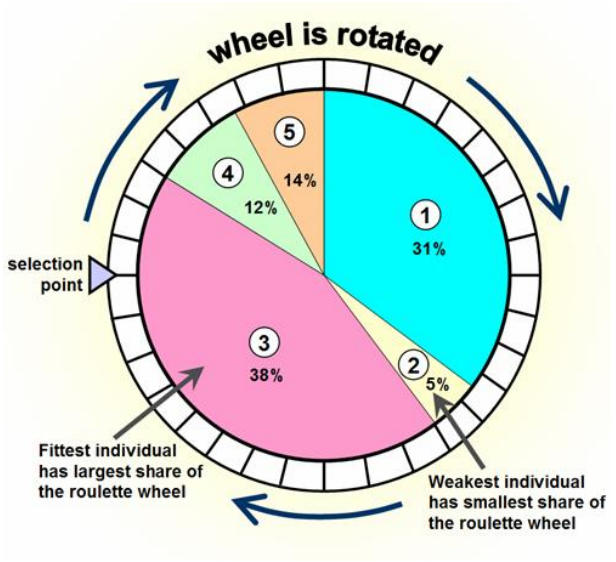
A Simple GA

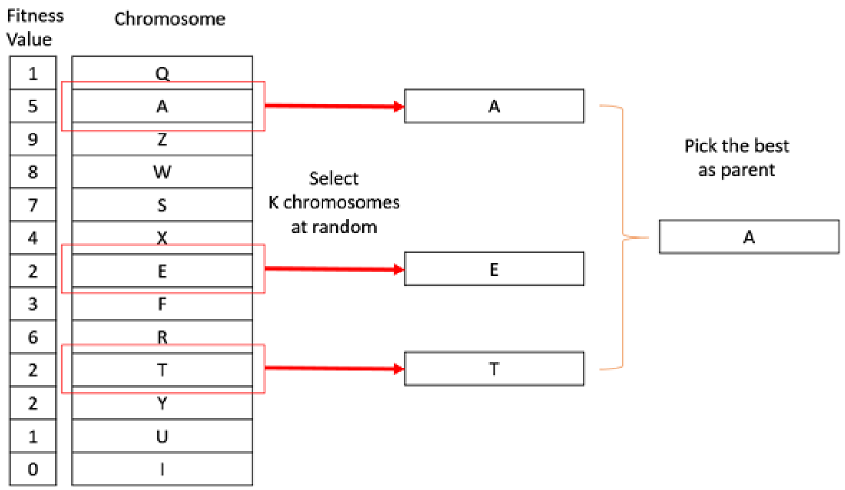
- Initialise a population of chromosomes
- **Repeat until** stopping criteria are met:
  - Fitness evaluation of each individual
  - Construct an empty new population
  - Do elitism (copy top individuals)
  - **Repeat until** the new population is full:
    - Select **two** parents from the population
    - Apply crossover to the two parents to generate two children
    - Each child has a probability (mutation rate) to undergo mutation
    - Put the two (mutated) children into the new population
  - **End Repeat**
- **End Repeat**



Selection

- Uniform selection
  - Each individual has the same chance to be selected
- Fitness-proportional (Roulette wheel) selection
  - The probability of being selected is proportional to the fitness
  - Fitness must be **maximised**
- K-tournament selection
- Truncate selection
  - Select the best individual(s)

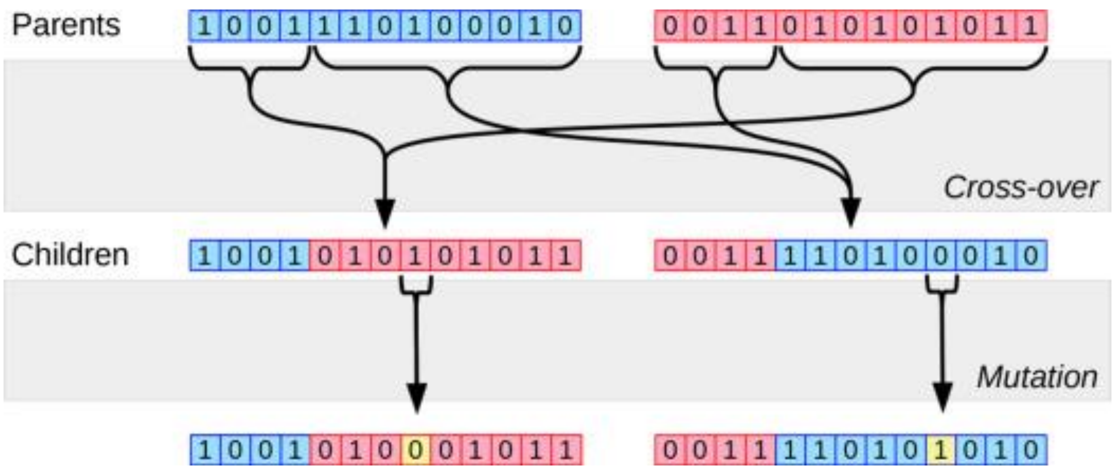

wheel is rotated
selection point
5
14%
4
12%
1
31%
3
38%
2
5%
Fittest individual has largest share of the roulette wheel
Weakest individual has smallest share of the roulette wheel

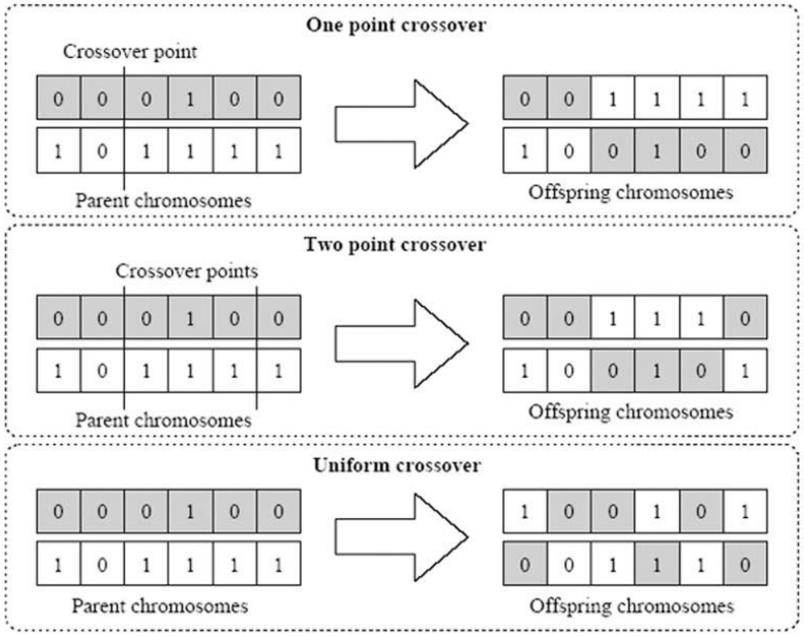


Fitness Value	Chromosome
1	Q
5	A
9	Z
8	W
7	S
4	X
2	E
3	F
6	R
2	T
2	Y
1	U
0	I

Genetic Operators

- Parents -> crossover -> (probably) mutation -> offspring
  - 1-point crossover
  - 2-point crossover
  - Uniform crossover
  - Flip mutation
  - …





GA for 0-1 Knapsack Problem

- Randomly initialise a population of individuals (bit string, each bit has 50% probability to be 1, and 50% to be 0)
- Best feasible solution is null
- **Repeat until** stopping criteria are met:
  - Fitness evaluation of each individual, update the best feasible solution

$$fit = \sum_{i=1}^{M} v_i x_i - \alpha * \max(0, \sum_{i=1}^{M} w_i x_i - Q)$$

  - Construct an empty new population
  - Do elitism (copy top individuals)
  - **Repeat until** the new population is full:
    - Select **two** parents from the population by roulette wheel selection
    - Apply one-point crossover to the two parents to generate two children
    - Each child has a probability (mutation rate) to undergo flip mutation
    - Put the two (mutated) children into the new population
  - **End Repeat**
- **End Repeat**
- **Return** the best feasible solution

GA for 0-1 Knapsack Problem

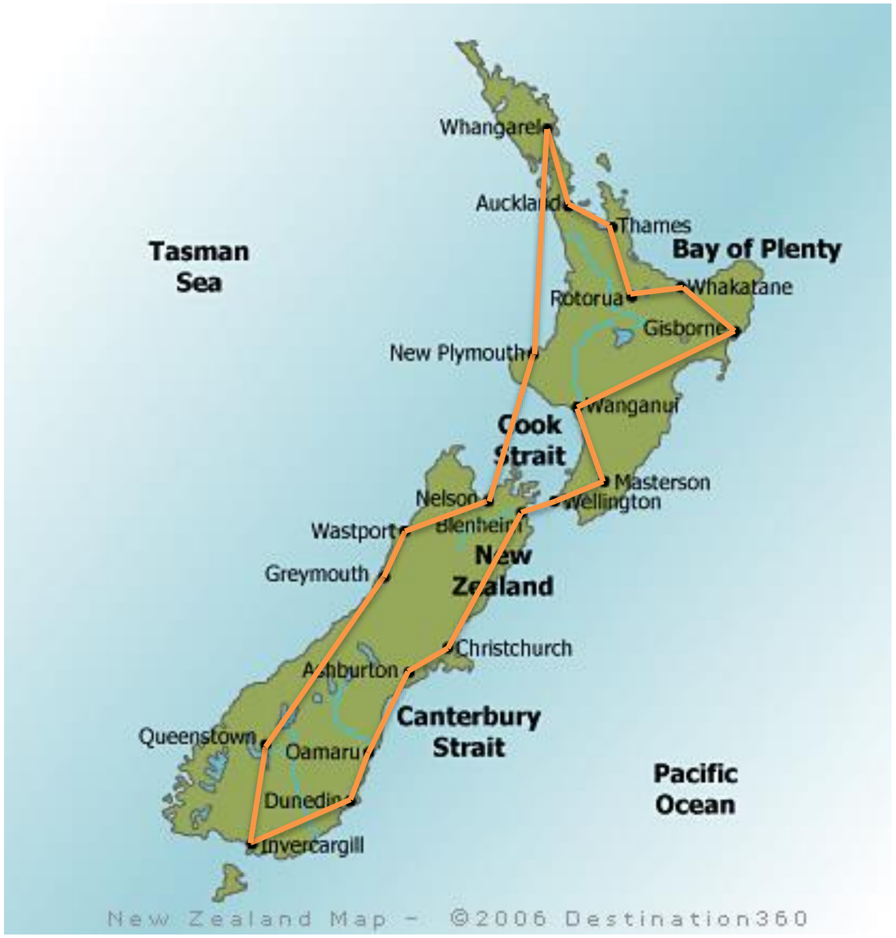
- Population size: 30/50/100
  - Make your best guess, more resource can afford larger population
- Fitness evaluation: penalise infeasible individuals
  - Penalty coefficient α: quality *v.s.* constraint violation
    - Very large value: always ignore infeasible solutions
    - Zero: only consider quality (bad choice, will get 1111....1)
    - Somewhere in between, parameter tuning, or adaptively change α
- Crossover rate 100%: always do crossover
- Mutation rate: 10%/20% **after crossover**
- Elitism: small value, e.g., top 5%, or top one or two individuals
- Stopping criteria: 100/200 generations
  - Observe the convergence curves to increase or decrease
  - Larger instances (more items) need more generations

GA for 0-1 Knapsack Problem

- **Advanced techniques**
  - Seeding better initial individuals, e.g., heuristics
  - Non-uniform crossover or mutation operator
    - E.g., an item with a higher value/weight ratio should be more likely to be selected (the bit more likely to be 1)
  - Adaptive strategies of α
    - If most individuals are poor but feasible, decrease α
    - If most individuals have high value but infeasible, increase α
    - …
  - Combine with local search (memetic)
    - Instead of flip mutation, enumerate flipping all the bits and select the best feasible one ($O(M)$ complexity)

GA for TSP

- Shortest road trip in NZ?



GA for TSP

- Traveling Salesman Problem
  - Given a graph with nodes {1, 2, ..., N}
  - Assume fully connected
  - Find the min-cost Hamiltonian Cycle
    - E.g., [1, 3, 5, 2, 4, 1]
    - [1, 4, 3, 2, 5, 1]
    - [2, 4, 3, 5, 1, 2]

- What are the constraints?

Tasman Sea
Whangarei
Auckland
Thames
Bay of Plenty
Whakatane
Rotorua
Gisborne
New Plymouth
Wanganui
Cook Strait
Masterson
Wellington
Nelson
Blenheim
Wastport
Greymouth
New Zealand
Christchurch
Ashburton
Canterbury Strait
Queenstown
Oamaru
Dunedin
Invercargill
Pacific Ocean
New Zealand Map - ©2006 Destination360

GA for TSP

- Individual representation?
  - Adjacency matrix
    - Binary entries
    - Each row/column must have exactly a single 1
    - Cannot have sub-cycle
  - Adjacency list
    - Each node list has only one distinct element
    - Cannot have sub-cycle
  - Permutation
    - [v1, v3, v2, v1]

	v1	**v2**	**v3**
v1	0	0	1
v2	1	0	0
v3	0	1	0

v1	**v2**	**v3**
v3	v1	v2

- Different representations lead to different search space
  - Need different genetic operators

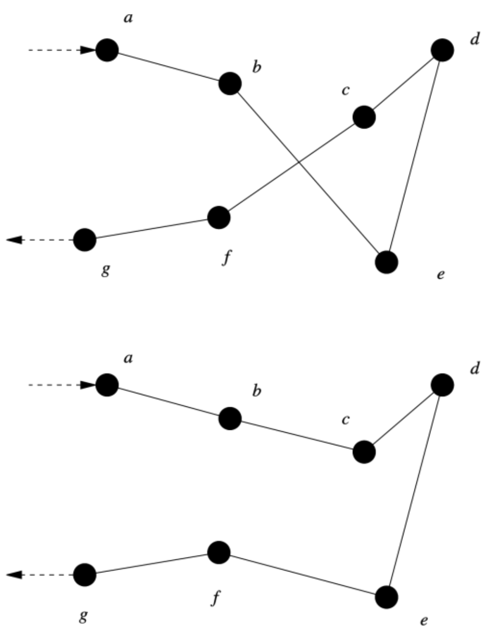
GA for TSP

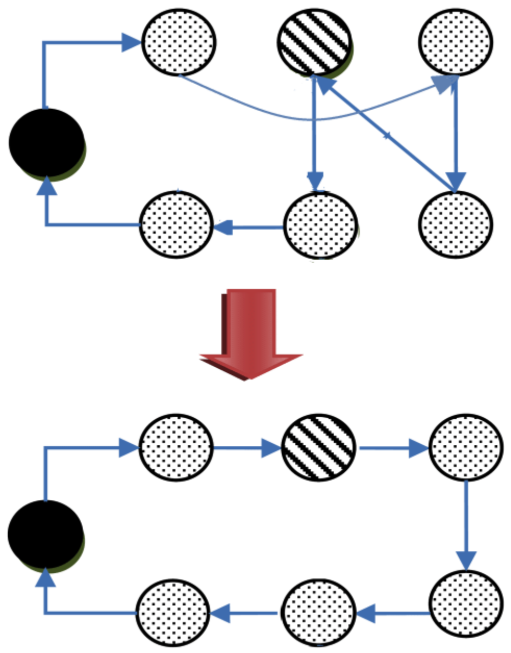
- Mutation operator?
  - Adjacency matrix
    - Flip 0/1?
    - Swap two rows/columns?
  - Adjacency list
    - Swap two node lists/neighbours?
  - Permutation
    - Swap two node locations?

- It's hard to design operators from representation to meet the problem-specific constraints
  - Each node is visited exactly once, and the solution is a cycle

GA for TSP

- Mutation based on domain knowledge
  - 2-opt: reverse a sub-tour in the cycle
  - Move one node to another place
- Change representation accordingly
  - Permutation is the most straightforward

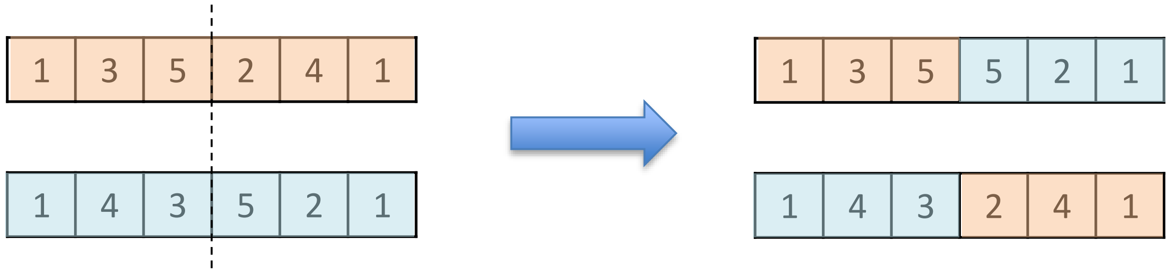




GA for TSP

- **Crossover?**
  - How to crossover two permutations to create a new permutation?

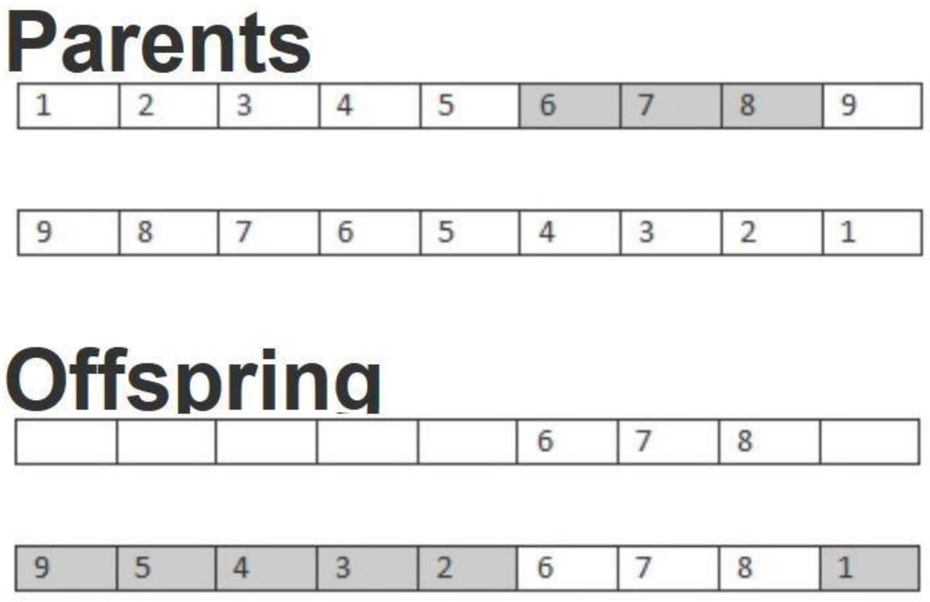
- One-point crossover cannot work
  - Some nodes will be missing, some will be duplicated



GA for TSP

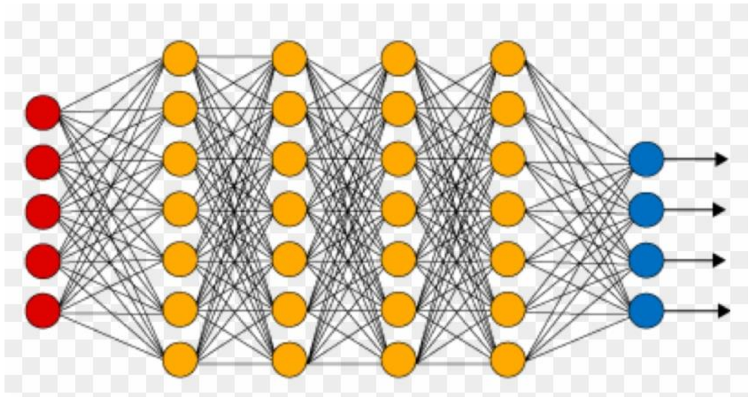
- **Partially Mapped Crossover**
  - Copy a segment from a parent
  - Scan the other parent to fill in the blank positions
    - Skip the duplicate nodes

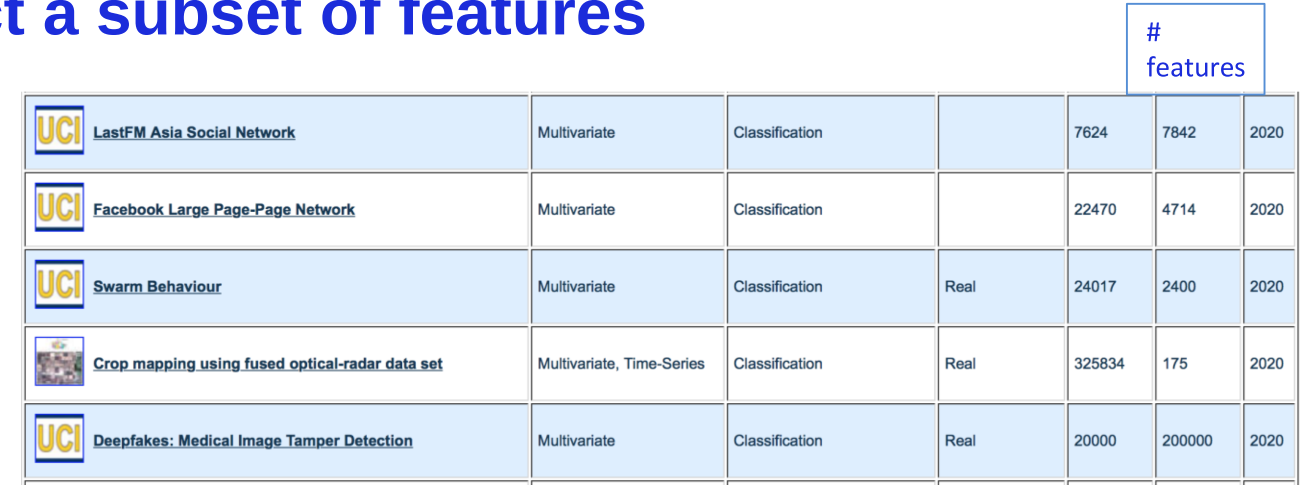
- Can keep some order of parents
- Satisfy the TSP constraints



Feature Selection

- Machine learning: prediction from features
- **Big data**: can be MANY features
  - Hard to train/generalize
  - Worsen the accuracy and efficiency
  - Hard to understand the learned model
  - Some cases hard/expensive to collect data
- **Select a subset of features**

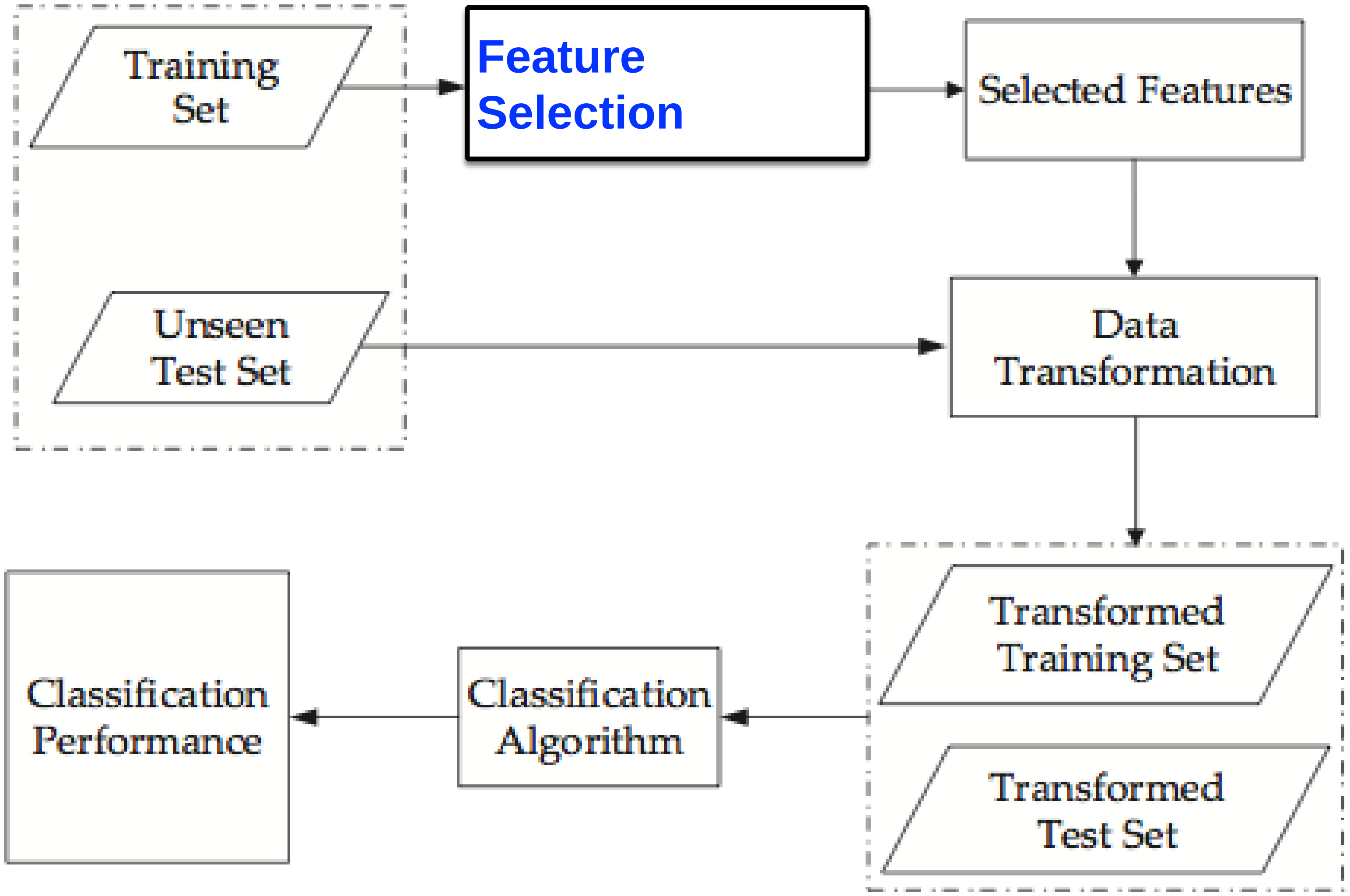




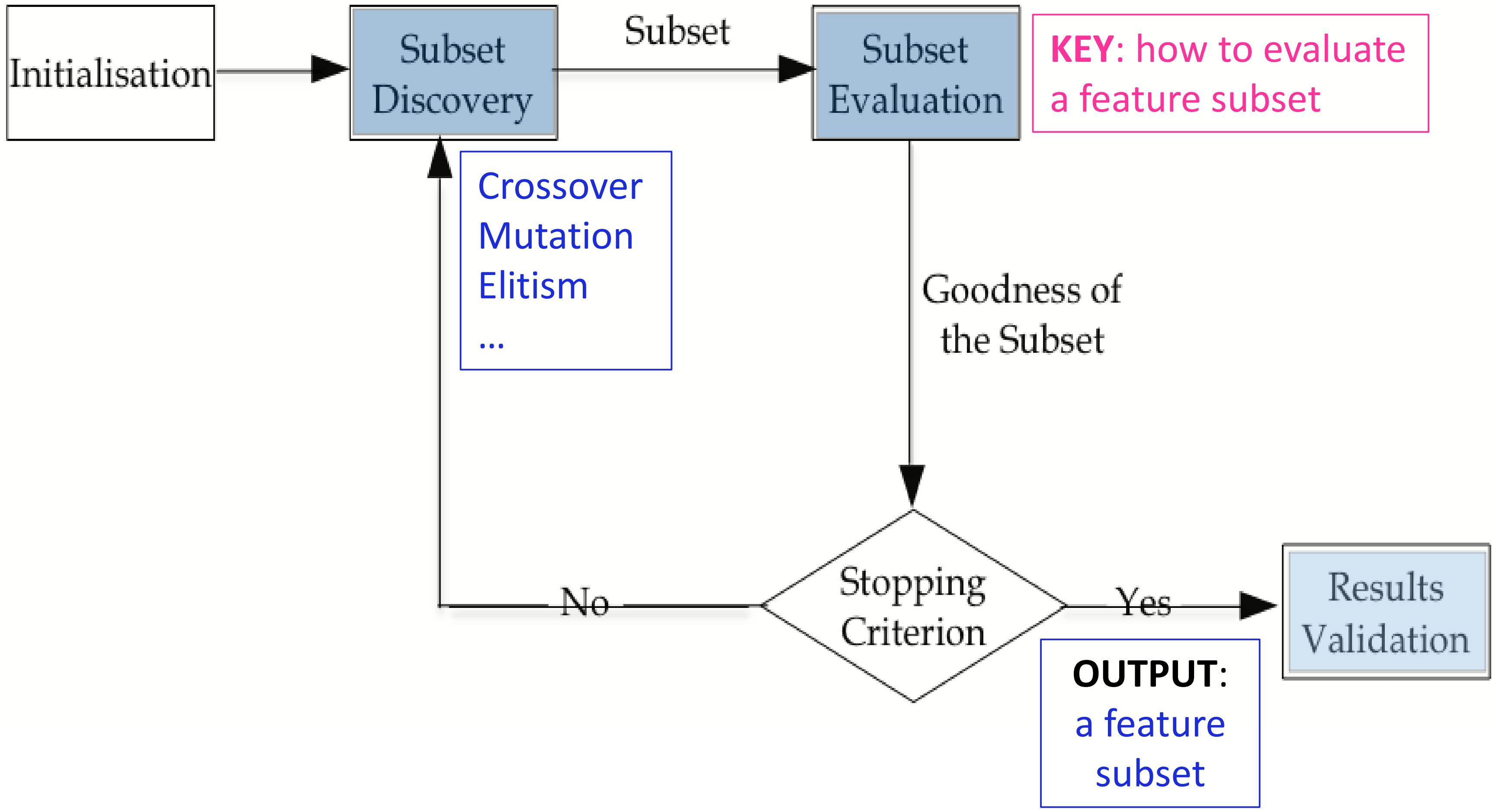
					# features	
LastFM Asia Social Network	Multivariate	Classification		7624	7842	2020
Facebook Large Page-Page Network	Multivariate	Classification		22470	4714	2020
Swarm Behaviour	Multivariate	Classification	Real	24017	2400	2020
Crop mapping using fused optical-radar data set	Multivariate, Time-Series	Classification	Real	325834	175	2020
Deepfakes: Medical Image Tamper Detection	Multivariate	Classification	Real	20000	200000	2020

Examples in UCI machine learning datasets

Feature Selection



GA for Feature Selection



GA for Feature Selection

- Randomly initialise a population of individuals (bit string, each bit has 50% probability to be 1, and 50% to be 0)
- **Repeat until** stopping criteria are met:
  - **Fitness evaluation** of each individual
  - Construct an empty new population
  - Do elitism (copy top individuals)
  - **Repeat until** the new population is full:
    - Select **two** parents from the population by roulette wheel selection
    - Apply one-point crossover to the two parents to generate two children
    - Each child has a probability (mutation rate) to undergo flip mutation
    - Put the two (mutated) children into the new population
  - **End Repeat**
- **End Repeat**
- **Return** the best feature subset

The same as GA for 0-1 knapsack, the ONLY difference is the fitness evaluation

GA for Feature Selection

- Filter-based fitness function
  - Goodness of a feature (subset) independent of a model
  - Example: mutual information (assume features are discrete)
    - **Step 1**: calculate entropy of the class $H(Y) = -\sum_y p(y) * \log_2 p(y)$
    - **Step 2**: calculate conditional entropy of Y given a feature subset $X = (X_1, \dots, X_m)$

      $$H(Y|X) = \sum_{x_1,\dots,x_m} p(x_1, \dots, x_m) H(Y|X_1 = x_1, \dots, X_m = x_m) = -\sum_{x_1,\dots,x_m} \sum_y p(x_1, \dots, x_m) * p(y|x_1, \dots, x_m) * \log_2 p(y|x_1, \dots, x_m)$$

    - Step 3: calculate mutual information between Y and X: $I(Y;X) = H(Y) - H(Y|X)$
  - Information Gain Fitness of a feature (subset)

    $$Fit_{IG}(X) = I(Y;X)$$

  - Information Gain Ratio Fitness

    $$Fit_{IGR}(X) = \frac{I(Y;X)}{H(X)}$$

  - Symmetric Uncertainty Fitness

    $$Fit_{SU}(X) = \frac{2 * I(Y;X)}{H(X) + H(Y)}$$

Continuous features must be discretised!

GA for Feature Selection

- Wrapper-based fitness function
  - Goodness of feature (subset) depends on model
  - Select a classifier to be used (e.g., KNN, Decision tree)
  - Given a feature subset to be evaluated, transform the training dataset by removing the unselected features
  - Apply the classifier to the transformed training dataset
  - Set the classification accuracy to be the fitness of the feature subset

  - Usually better than filter-based fitness functions
  - Dependent on the classifier selected
  - Slower than filter-based fitness functions

Randomness in GA

- A lot of randomness/uncertainty
  - Initialisation
  - Parent selection
  - Crossover (random cutting point)
  - Mutation (random flipped bit)
  - Mutation rate
- GA is a stochastic algorithm (in fact almost all EC algorithms are stochastic)
  - Run with different random seeds will get different results
- To see how good a GA is, need to run many (at least 30) times
  - Calculate mean and standard deviation
  - Boxplot
  - Statistical test (e.g., t-test, Wilcoxon rank sum test)

Randomness in GA

- Showing results of GA

Raw data

Run	**Gen**	**Fitness**
1	1	5.4
1	2	3.2
1	3	2.1
1	4	1.1
1	**5**	**1.0**
2	1	6.7
2	2	3.1
…	…	…
5	4	1.4
5	**5**	**1.2**

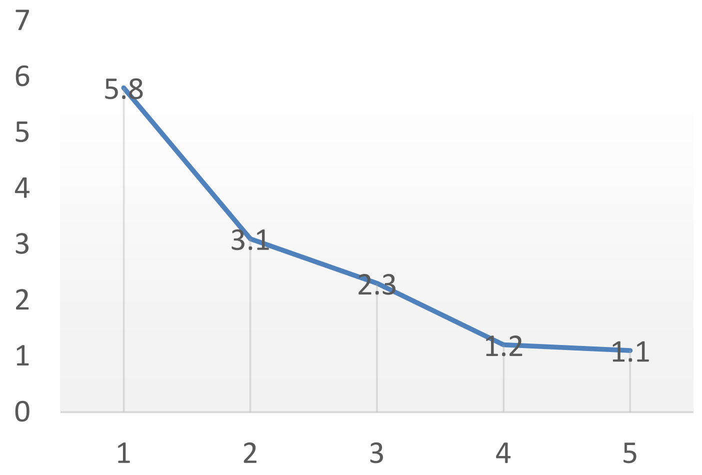
Avg Fitness per generation

Gen	**Avg Fitness**
1	5.8
2	3.1
3	2.3
4	1.2
5	**1.1**

Final performance

Mean	**Std**
1.1	0.1

Convergence Curve



Summary

- Genetic Algorithm Design
  - Bit string
  - 1-point / 2-point crossover, flip mutation
- GA for 0-1 Knapsack
- GA for TSP
- GA for feature selection
- Showing results of multiple GA runs
  - Final mean and standard deviation
  - Convergence curve