Fundamentals of Artificial Intelligence

VICTORIA UNIVERSITY OF WELLINGTON
TE HERENGA WAKA

COMP307/AIML420

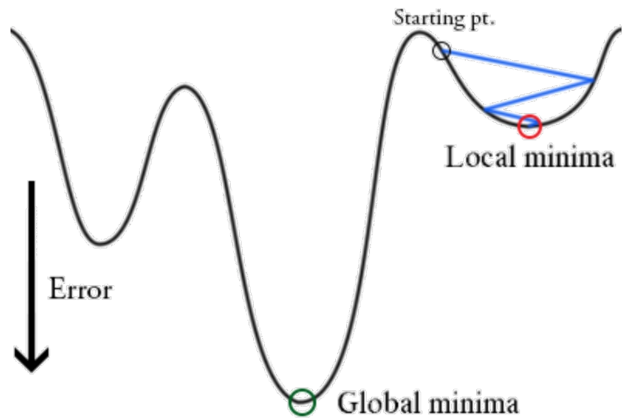
Evolutionary Computation 1: Evolutionary Computation and Learning

Outline

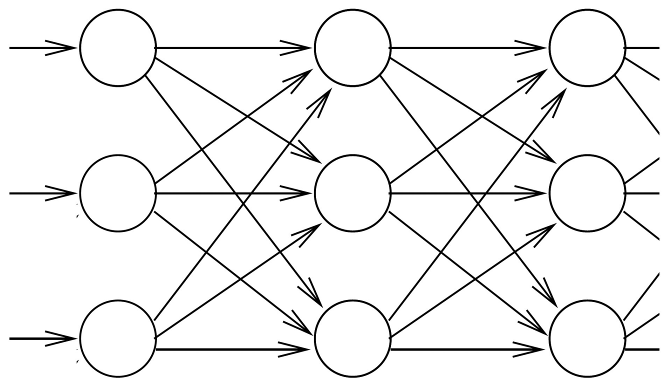
- Why evolutionary computation (EC) and learning?
- What is EC?
- EC Techniques
- Key characteristics and design questions
- Genetic algorithms: representation, selection and genetic operators
- Overview of other evolutionary algorithms

Why Do We Need Evolutionary Computation?

- We have discussed several methods and algorithms in ML
- But they have limitations:
  - Local optima



  - Needs to predefine/fix the structure/model of the solution, and only learns the parameters/coefficients



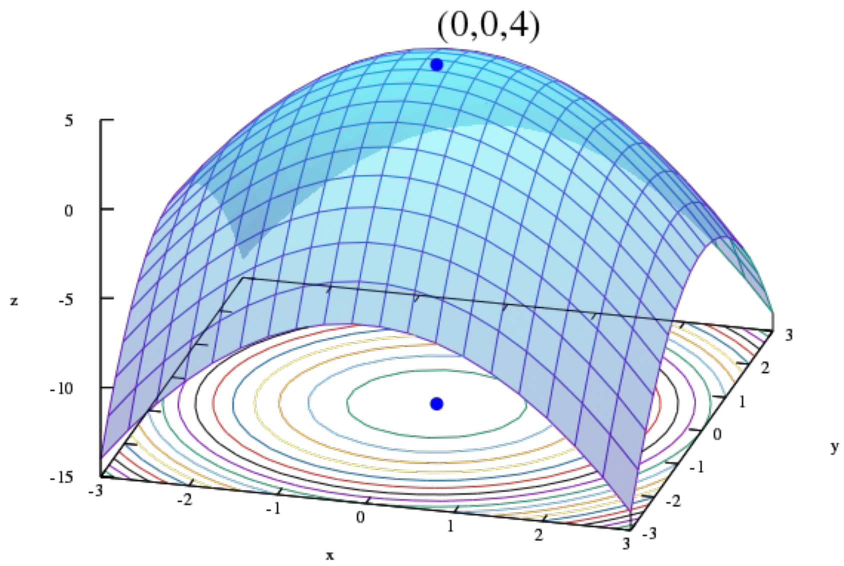
  - Many parameters to learn (high-dimensional optimization)
- Evolutionary Computation (EC) is one technique that can avoid some of the problems

What is Optimization?

- 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., minimize fuel use of courier deliveries: time, distance
  maximize classification accuracy

- Decision variable(s)
- Objective function(s)
- Constraint(s)
- …



Examples

- In machine learning
  - Optimize the weights of a neural network
  - Optimize the architecture (#layers, #nodes) of a neural network
  - 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 computing
  - **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.
- Good individuals have better chance to survive in the nature.

- Three different interpretations of this idea were developed independently:
  - Evolutionary programming: Lawrence Fogel (USA)
  - Evolution 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

Evolutionary Computation and Learning

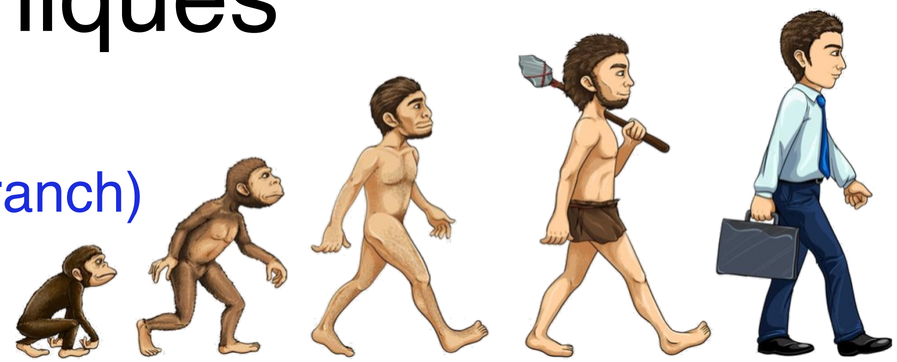
- In computer science, **evolutionary computation** is a family of "*nature inspired*" AI algorithms for global optimization.

- In technical terminology, they are a family of population-based trial-and-error problem solvers with a metaheuristic or stochastic optimization character.

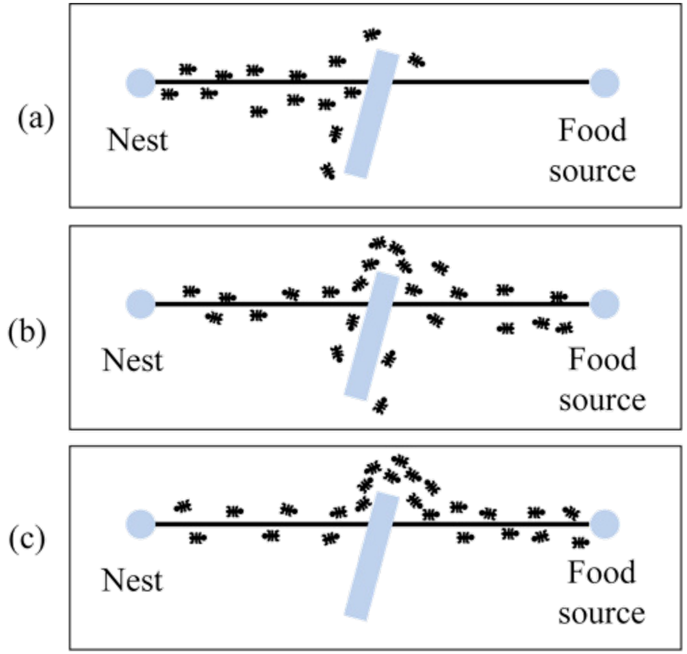
- **Evolutionary Learning** is the use of evolutionary computation methods for tackling machine learning tasks

EC Techniques

- Evolutionary algorithms (EAs)
  - Genetic algorithms (the biggest branch)
  - Evolutionary programming
  - Evolution strategies
  - Genetic Programming (Koza, 1990s, fast growing area)

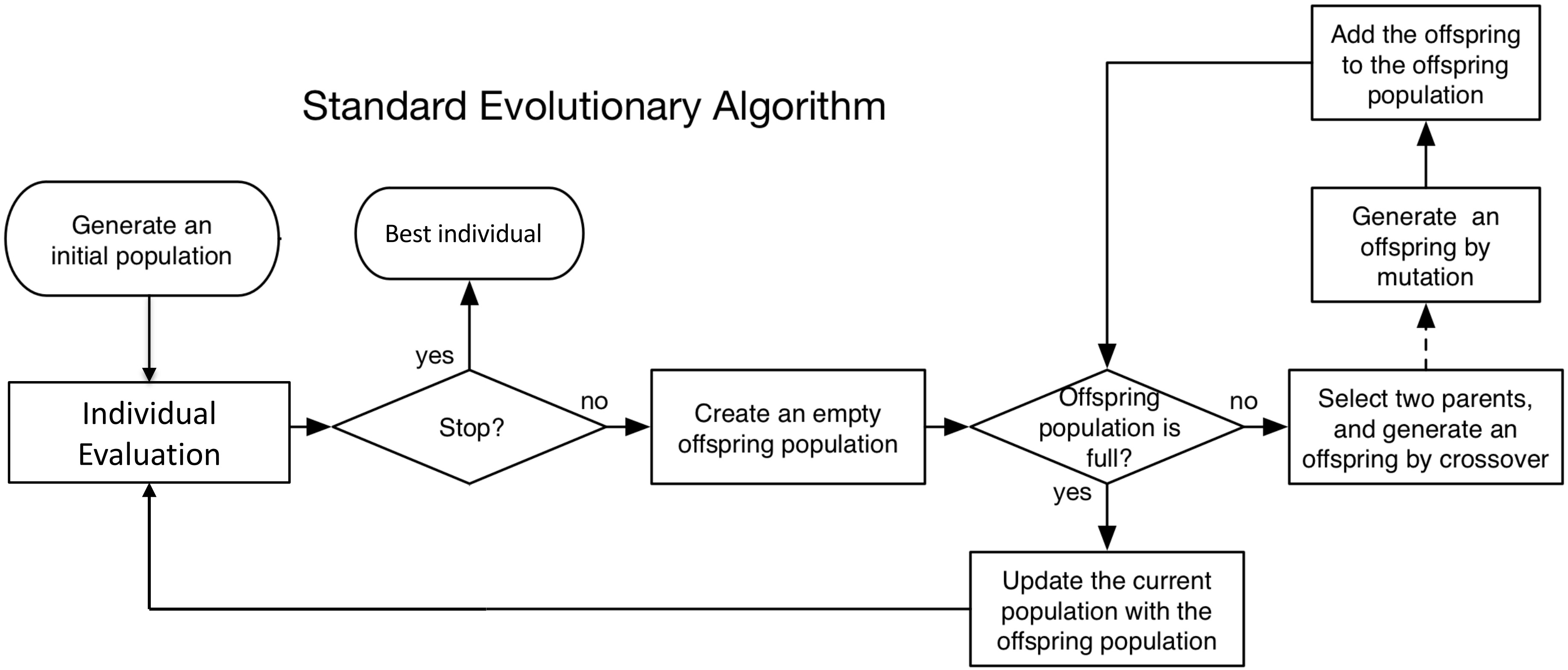


- Swarm intelligence (SI)
  - Ant colony optimization
  - Particle swarm optimization (PSO)
  - Artificial immune systems



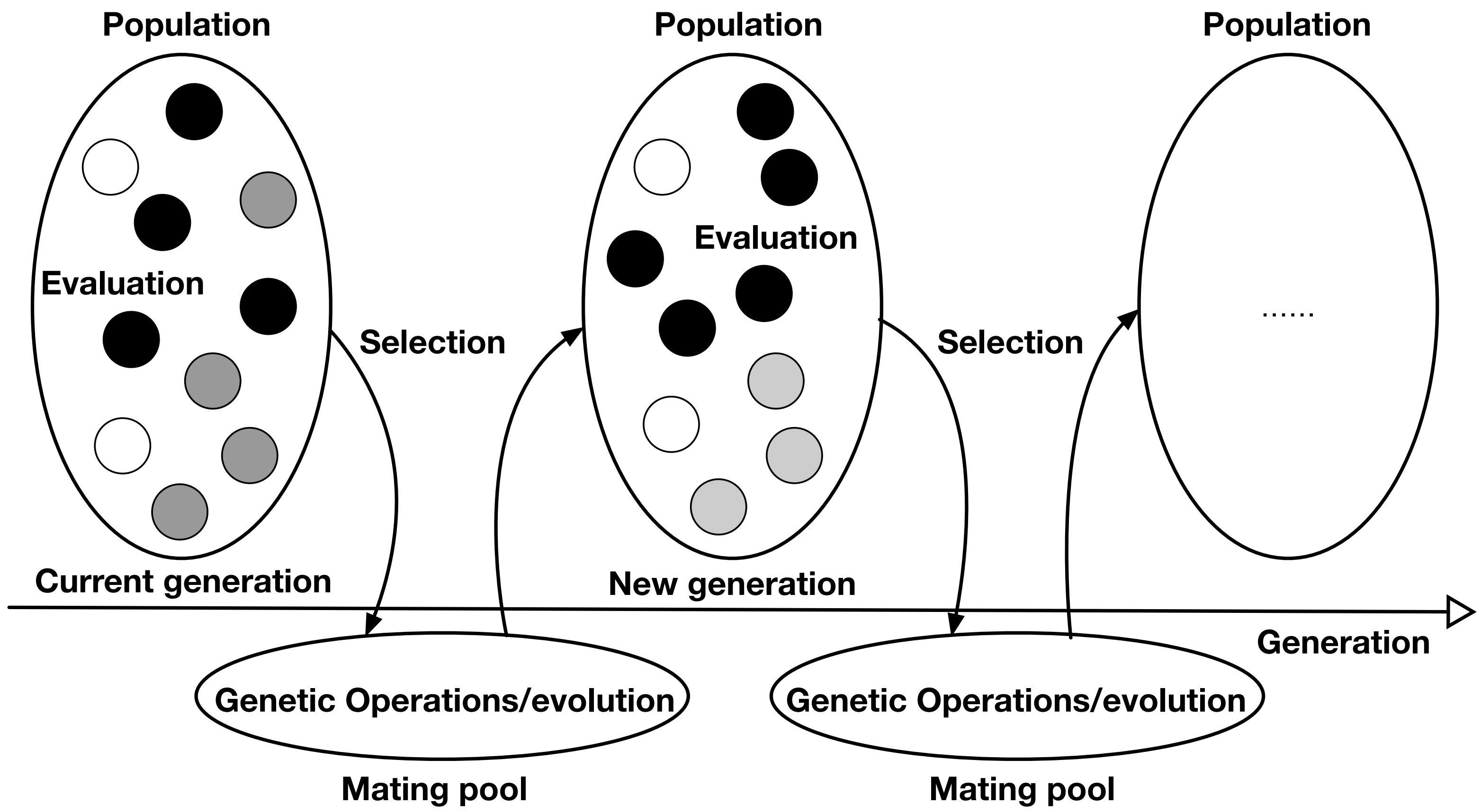
- Other techniques
  - Differential evolution
  - Estimation of distribution algorithms
  - …

Evolutionary Algorithm



Evolutionary Algorithms

- Search for the best individual by evolving a *population* with genetic operators (e.g., reproduction, 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

Population VS Individual

Evolution

- Final solution (individual): the one with the best *fitness*
- Fitness could be accuracy, cost, error, …

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 optimization)

-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)

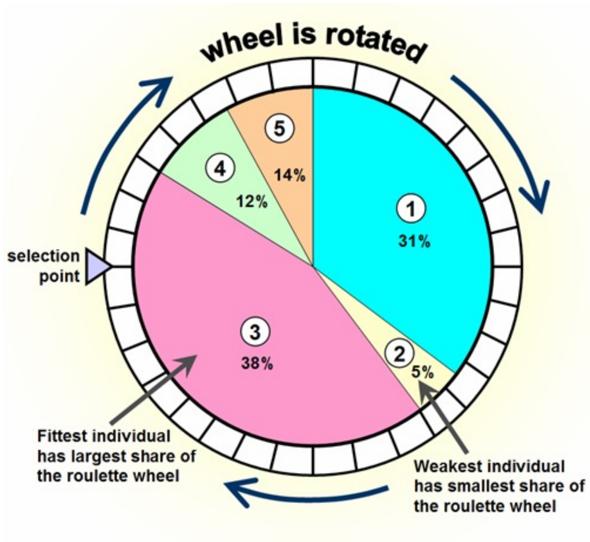
Fitness Evaluation

- **Fitness function**: reflect the quality of individuals
  - Must correspond to optimality property
  - Must be computable
  - Smoothness (in general):
    - Small changes to candidate -> small changes to quality/fitness
    - Large changes to candidate -> large change

- Depending on the problem, the fitness function could be:
  - the larger, the better — maximization
  - the smaller, the better — minimization
  - e.g., for classification, maximizing accuracy or minimizing error

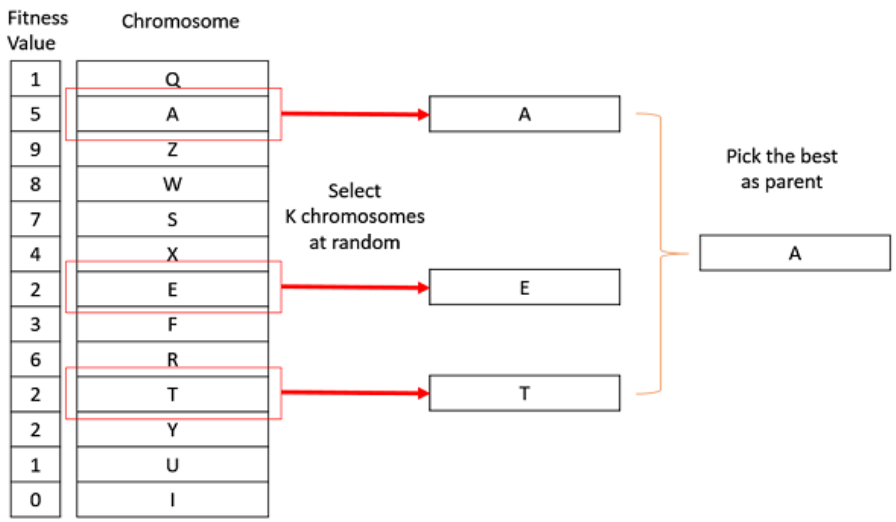
Selection

- Uniform selection
  - Each individual has the same chance to be selected



- Roulette wheel selection (GA)
  - The probability of being selected is proportional to the fitness
  - Assume fitness is to be maximized

- K-tournament selection (GP)



- Truncate selection
  - Select the best k individuals

Selection pressure VS diversity

Genetic Operators

- Depends on the problem – individual representation

A representative: Genetic Algorithms

– Relocate a bit of a binary vector

1	0	0	1	1	→	1	**1**	0	**0**	1

– Resample an element of a continuous vector

-0.73	**0.10**	0.35	-0.06	0.23	→	-0.73	**0.18**	0.35	-0.06	0.23

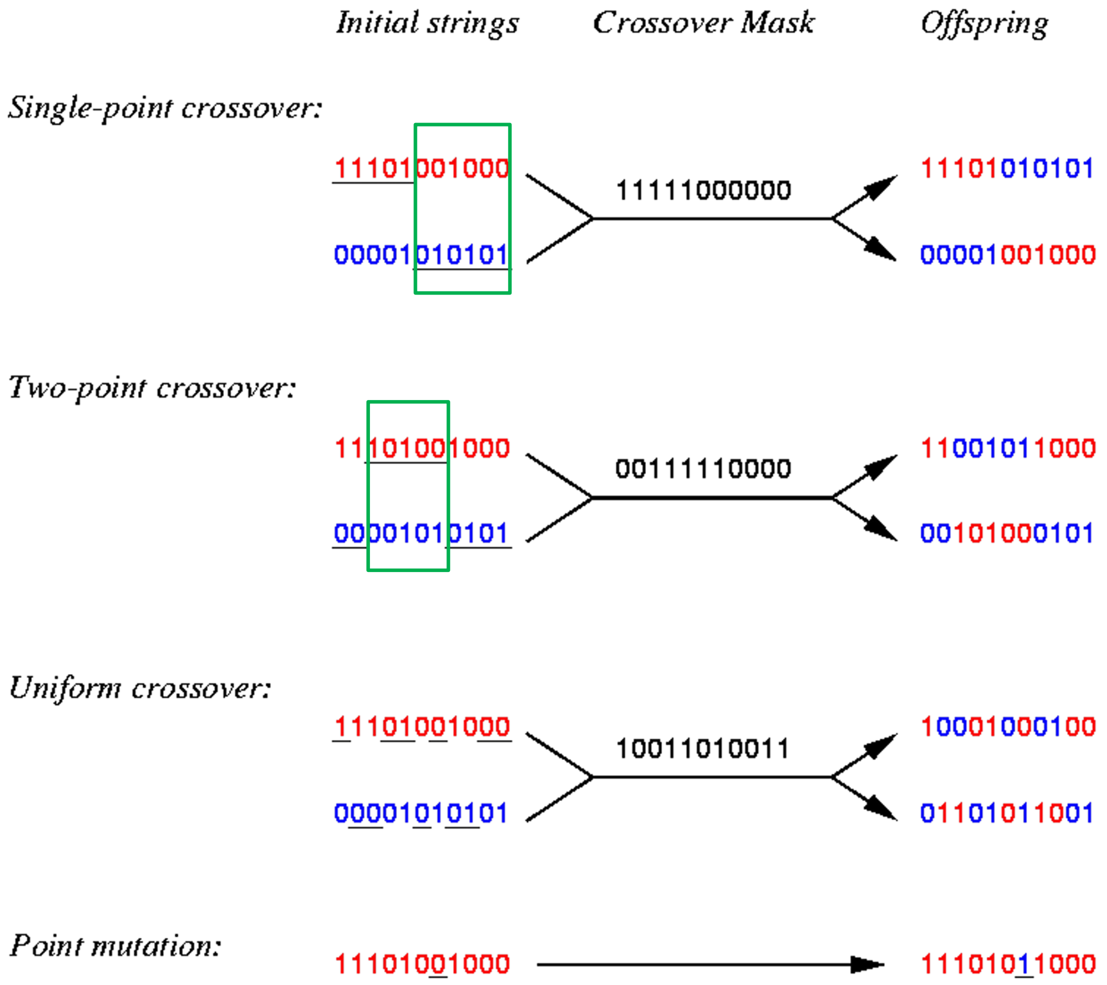
– Shuffle a part of a sequence

1	3	5	2	4	→	1	4	5	2	3

– ...

Genetic Algorithm

- Representation: individuals are binary strings
- An individual is also called a chromosome
- Elitism (keep best *k* individuals to the next generation, e.g., 1)



A Basic Genetic Algorithm

- Randomly initialize 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)

  Or just $\sum x_i$ (more greedy), Or $100 + \sum x_i$ (any constant number)

- Selection: Roulette wheel selection
- Genetic operators: Crossover: single-point crossover

  Mutation: point mutation

A Simple GA Example

- 10 bits (Optimal fitness = 11)
- population size = 20
- mutation rate = 0.1 (10%), crossover rate = 0.8 (80%), reproduction rate = 0.1 (10%)
- 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
```

Keep elites (i.e., best ones) to the next generation!!!

Summary

- Evolutionary computation
- Representations
- Selection and genetic operators
- Genetic algorithms

- **Next Tutorial**: EC and its applications