Evolutionary Computation and Learning

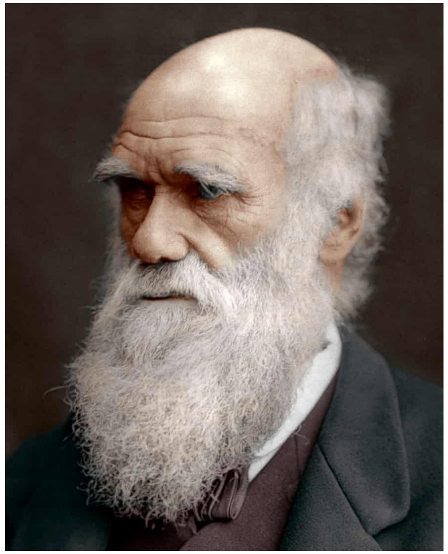
Represented by:
Vahid Ghasemi
v.ghasemi@kut.ac.ir

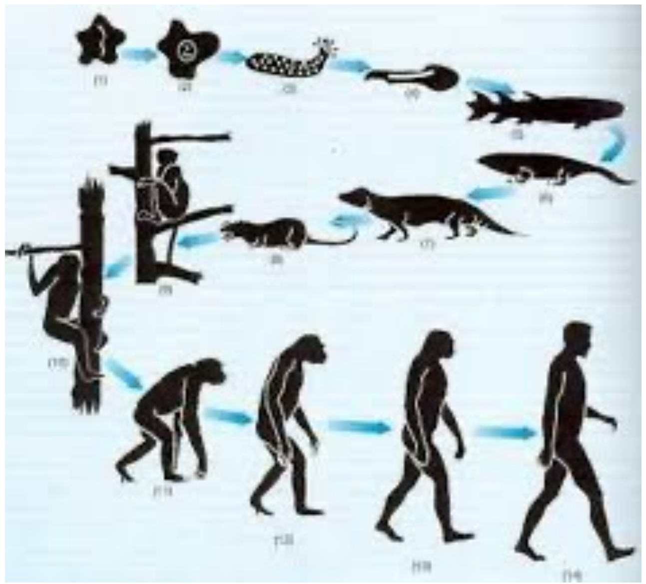
Outline

- What is Evolutionary Computation?
- Why Evolutionary Computation
- Key Design Issues in EC and Examples
- A Unified View of Evolutionary Algorithms

Evolution in Nature

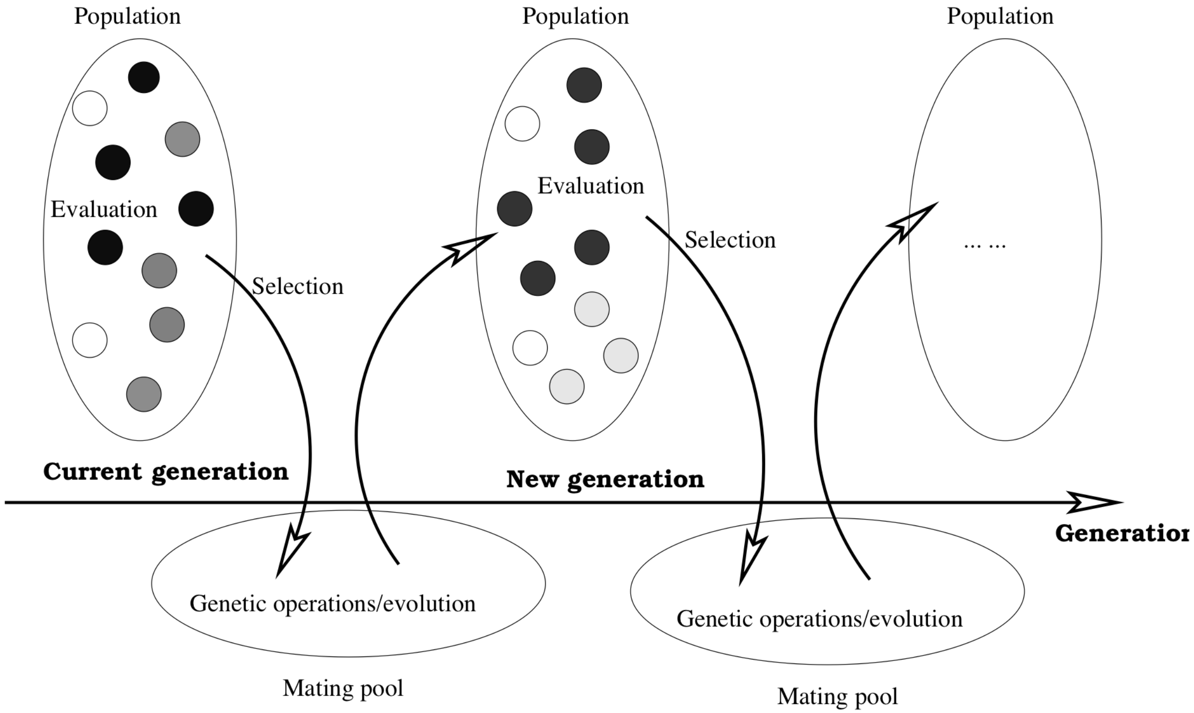
- Darwin's Theory of Biological Evolution
  - "Survival of the fittest"
  - Breeding and random mutation
  - Natural selection
- Can we use evolution in computation?





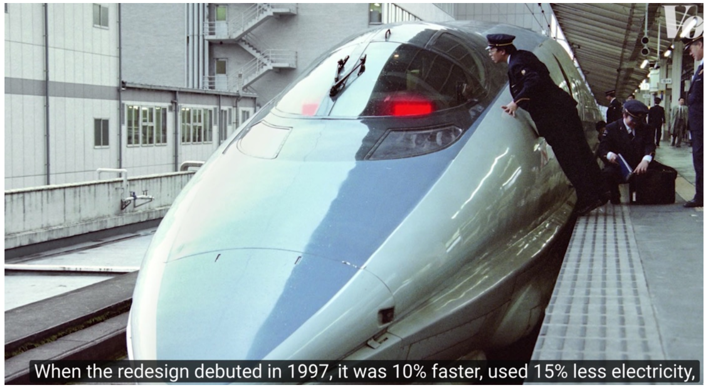
Evolutionary Computation

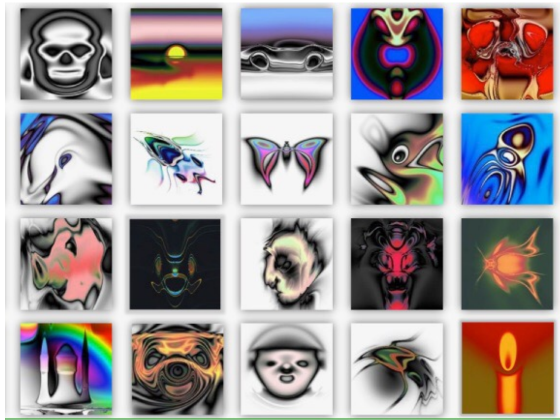
- A group of techniques inspired by the biological evolution
  - A population (set) of individuals
  - Breeding process: new offspring generated, old died
    - Crossover, mutation
  - Natural selection (Survival of fittest)
    - Fitness evaluation
- **What can EC do for you?**

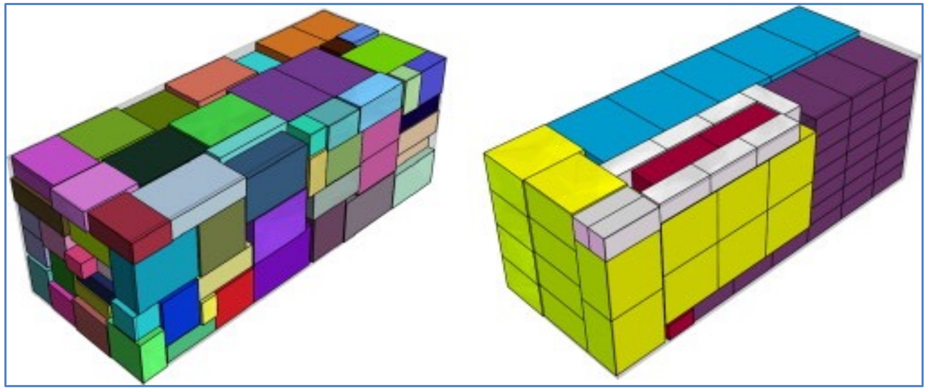


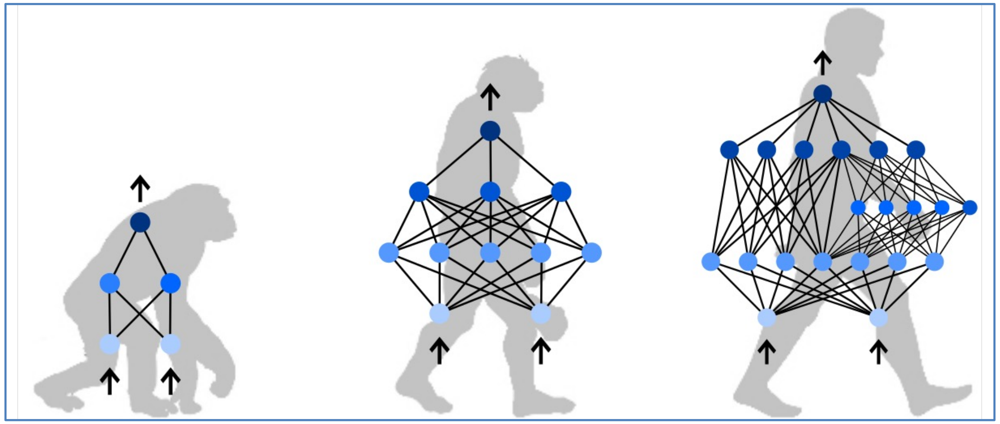
What Can EC Do For You?

- Optimisation
- Learning
- Creative design
- …



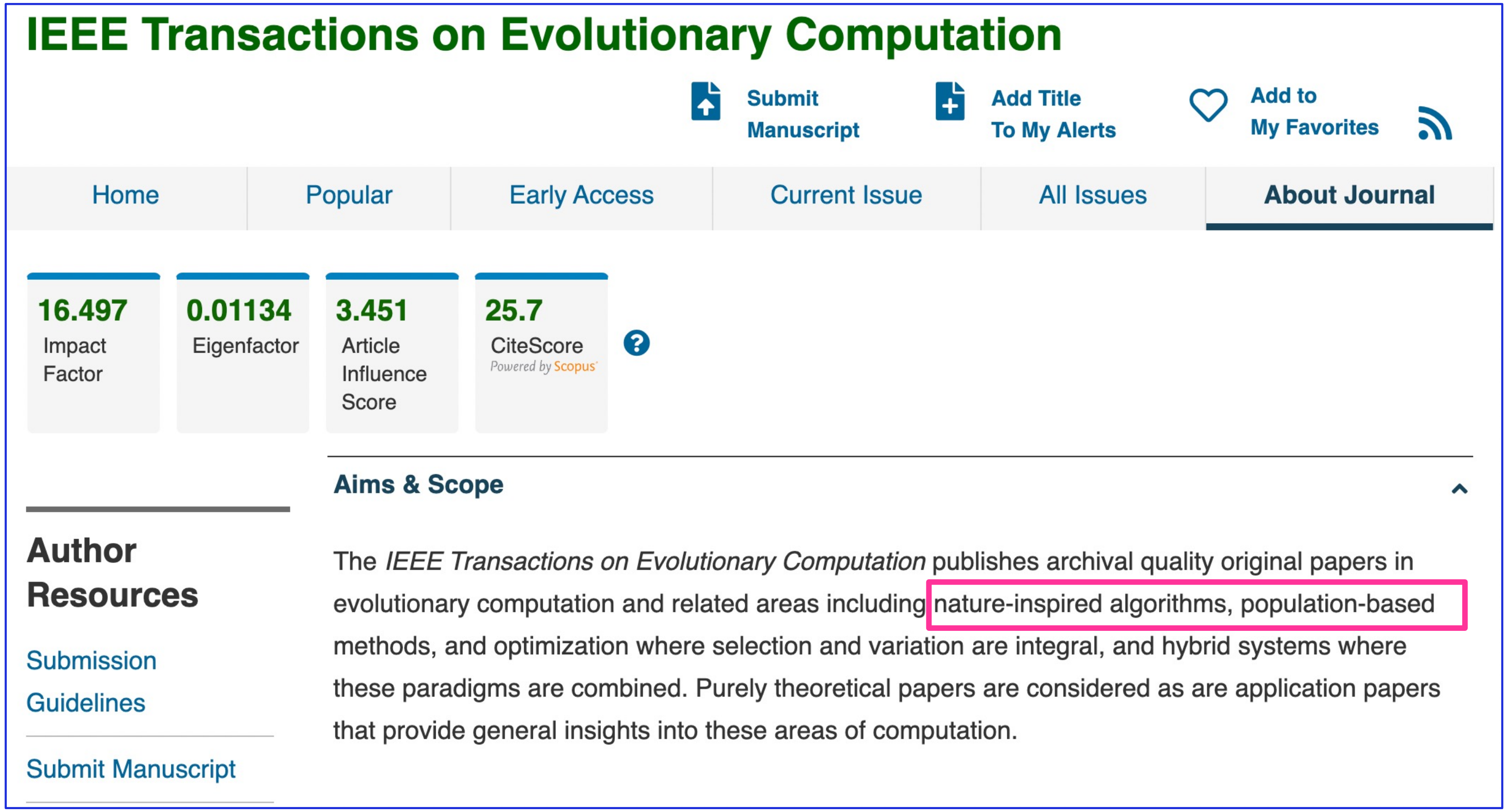






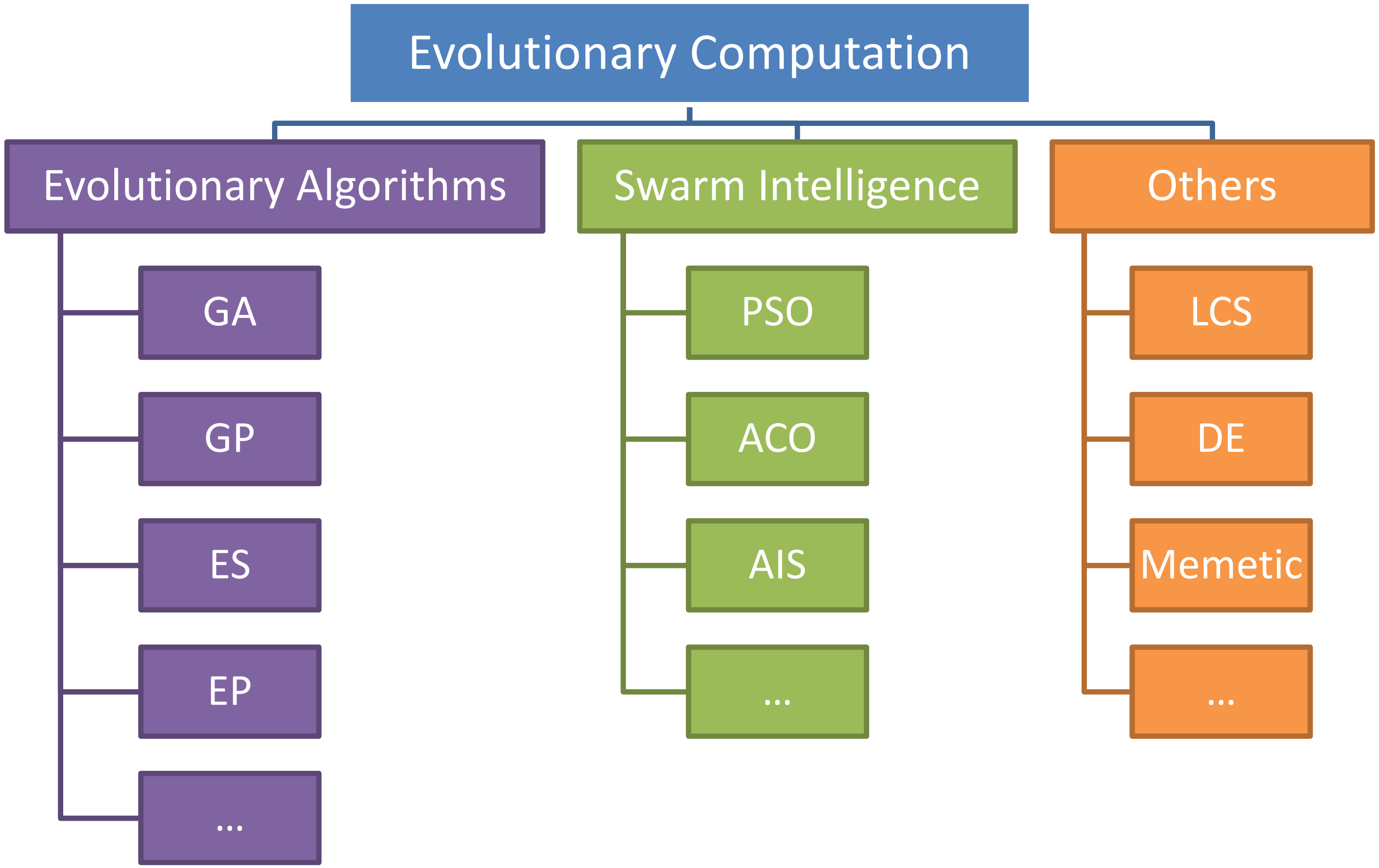
Evolutionary Computation

- Included other related nature-inspired techniques and population-based approaches
  - Evolutionary algorithms (natural evolution-inspired)
  - Swarm intelligence (more social-inspired)
  - Others …



Evolutionary Computation

- Included other related nature-inspired techniques and population-based approaches
  - Evolutionary algorithms (natural evolution-inspired)
  - Swarm intelligence (more social-inspired)
  - Others …



Why Evolutionary Computation

- Can solve a problem without requiring domain knowledge
  - Incorporating domain knowledge can enhance its performance
- NO strict assumption
  - Continuous, differentiable, linear, convex, …
- Easy to handle constraints
- Can simultaneously learn model structure and parameters
  - Genetic Program for Symbolic Regression
- Population-based search is ideal for multi-objective optimisation

Principles of Evolutionary System

- One or more populations of individuals competing for **limited resources**
- Dynamically changing populations due to the birth and death of individuals
- A concept of fitness which reflects the ability of an individual to survive and reproduce/breed
- A concept of variational inheritance: offspring closely resemble their parents, but are not identical

Key Design Issues in EC

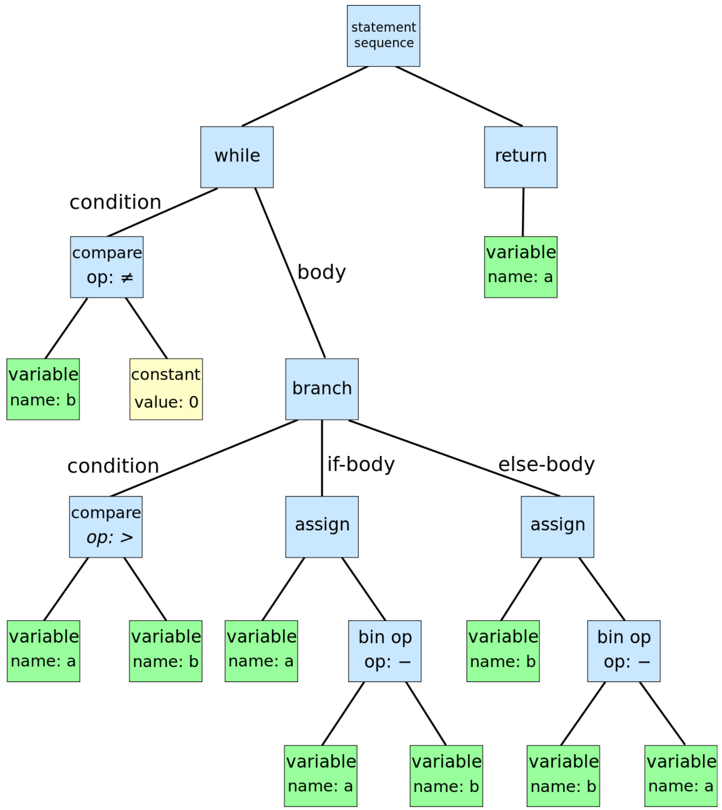
- Representation: How does an individual look like?
  - The designed shape/topology
  - A machine learning model (e.g., neural network, rule-based system)
  - Solutions (e.g., packing plan, schedules, decision-making rules)
- Population structure
  - How many population? How many individuals in each population?
  - Fixed/Variable population size
- Fitness evaluation
  - How good an individual is (compared with another individual)?
- Breeding
  - How to generate offspring (new individuals) from parents (existing)?
  - Parent selection, genetic operators, …
- Evolution
  - Which to survive, which to die?
  - When to stop?

Example: Genetic Algorithm

- Representation
  - Binary string: 011101000
- Population structure
  - A single population
  - Problem-specific population size parameter (e.g., 30, 50)
- Fitness evaluation: problem dependent
  - E.g., packing solution: minimize wasted space
- Breeding
  - Crossover/mutation operators
  - Elitism
- Evolution (Generational GA)
  - N parents generate N offspring (by crossover/mutation/elitism)
  - Parents are selected proportional to their fitness
  - N offspring replace the N parents to next generation

Example: Genetic Programming

- Representation: **Tree**/Graph/Linear …
- Population structure
  - A single population
  - Problem-specific population size parameter (e.g., 500, 1000)
- Fitness evaluation: problem dependent
  - E.g., regression accuracy/error
- Breeding
  - Crossover/mutation/reproduction
- Evolution
  - N parents generate N offspring (by crossover/mutation/reproduction)
  - Parents are selected proportional to their fitness
  - N offspring replace the N parents to next generation



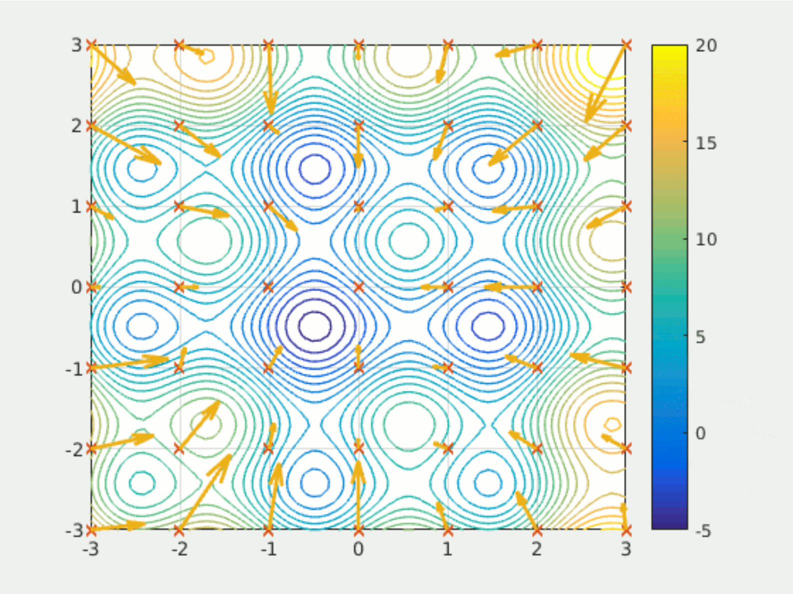

```
while b ≠ 0
  if a > b
    a := a − b
  else
    b := b − a
return a
```

Example: (1+λ)-Evolutionary Strategy

- Representation
  - Continuous vector: [0.1, 0.5, -1,2, …]
- Population structure
  - A single population, a single individual/parent
- Fitness evaluation: problem dependent
  - E.g., quality of the designed shape
- Breeding
  - For each number, add a noise from the normal distribution $\mathcal{N}(0, \sigma)$
- Evolution
  - Generate λ offspring from the single parent, and select the fittest offspring to replace the parent

Example: Particle Swarm Optimisation

- Representation
  - Continuous vector: [0.1, 0.5, -1,2, …]
- Population structure
  - A single swarm (population) with N particles (individuals)
- Fitness evaluation: problem dependent
  - E.g., quality of the designed shape
- Breeding
  - Based on movement of particles
  - Each particle follows the best particle
  - Each particle follows its historical best
- Evolution
  - Continuous move the location of the particles



2-D PSO example

Phenotype vs Genotype

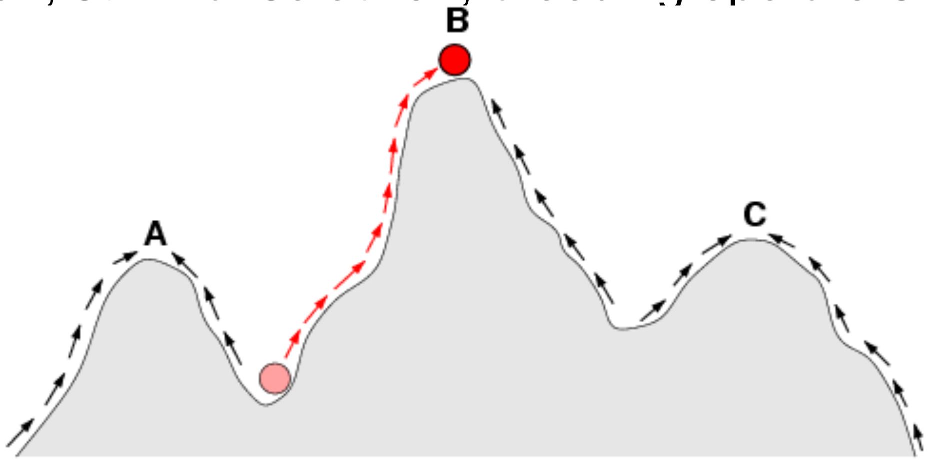
- **Phenotypic** representation
  - An individual correspond directly to a solution
  - E.g., a path/route: [A, D, E, B, C]
  - E.g., a numeric vector: [2, 5, 8, 1]
  - E.g., a clustering {{A, C}, {B, D}}
- **Genotypic** representation
  - An individual is an encoded solution, needs to be decoded
  - E.g., encoded path: [0.2, 0.6, 0.3, 0.7] -> [A, C, B, D]
  - E.g., binary code of numbers: 101 -> 5
  - E.g., encoded clustering [0, 1, 0, 1]
  - Genotype-Phenotype mapping
    - A slight change in the genotype can lead to a small or large permutation in the phenotype
    - We want to make the **mapping smooth** (small change in genotype always leads to small change in phenotype)
- Neither phenotypic nor genotypic representation is always better than the other. Depends on problem, and associated search operators

Constraint Handling

- Fitness Function/Assignment
  - Simple comparison
    - If A is feasible and B is infeasible, then A is better than B
    - If A and B are both feasible, then compare the objective value
    - If A and B are both infeasible, then compare the degree of violation
  - Penalty method
    - $fitness = obj + \alpha * violation$
    - **Special case**: if α is much larger than obj, it will be simple comparison
    - Proper α setting is critical: balance between quality and feasibility
    - Can be very helpful when infeasible solutions contain promising building blocks
  - Special representation that can always satisfy the constraint
    - Genotype + decoding (always decode to a feasible solution)
  - Special search operator (modify individuals) that always satisfy the constraint

Understanding EC

- Population-based search in a space
  - Continuous space or discrete space
  - Constrained or unconstrained (Feasible/Infeasible regions)
- Different points/individuals explore different regions
- Different points interact with each other
  - Find better unexplored regions
- Each point exploits local regions around it
- **KEY design principle**
  - Balance between exploration/diversity/randomness and exploitation/convergence/greediness
  - Parent selection, survival selection, breeding operators



A Unified View of EAs

- A population of M individuals evolving over time
- The current population is used to produce N offspring
- The expanded population is reduced from M+N to M individuals

- M: the degree of parallel search an EA supports
- N: how long one is willing to continue to use the current parent population as the basis for generating new offspring without integrating the newly generated high-fitness offspring back into the parent population

- Problem dependent
  - How hard the problem is (more local optima requires larger M)
  - How many resources we have (more resources can afford larger M and N)

A Unified View of EAs

- **Two selections**
  - Parent selection: select parents to generate offspring
  - Survival selection: select M individuals from the M+N individiuals into the next generation
- **Selection schemes**
  - Uniform
    - Each individual has the same chance to be selected, fitness is not used
  - Fitness-proportional (roulette wheel)
    - The probability of selecting each individual is proportional with its fitness
  - Size-K tournament selection
    - Randomly select K individuals, then select the one with the best fitness
  - Truncate selection
    - Directly select the top individual(s) with the best fitness
- **Selection pressure (Greediness)**
  - Uniform < Fitness-proportional < Tournament selection < truncate
  - Tournament selection: larger K is greedier

Summary

- Evolutionary computation is a group of techniques inspired by biological evolution (and also swarm intelligence)
- Evolutionary computation is good at solving complex problems
  - No domain knowledge required
  - Without strong assumptions
  - Can be easily tailored by incorporating domain knowledge
  - Constraint handling and multi-objective optimisation/decision making
- Balance between exploration and exploitation is the key
  - How to do parent selection and survival selection

- Suggested readings:
  - Kenneth A.. De Jong. (2006). Evolutionary Computation: A Unified Approach. MIT Press.