# Further Studies on Metaheuristics

- **[Question]** What are the differences between a heuristic and a metaheuristic?

# Comparison

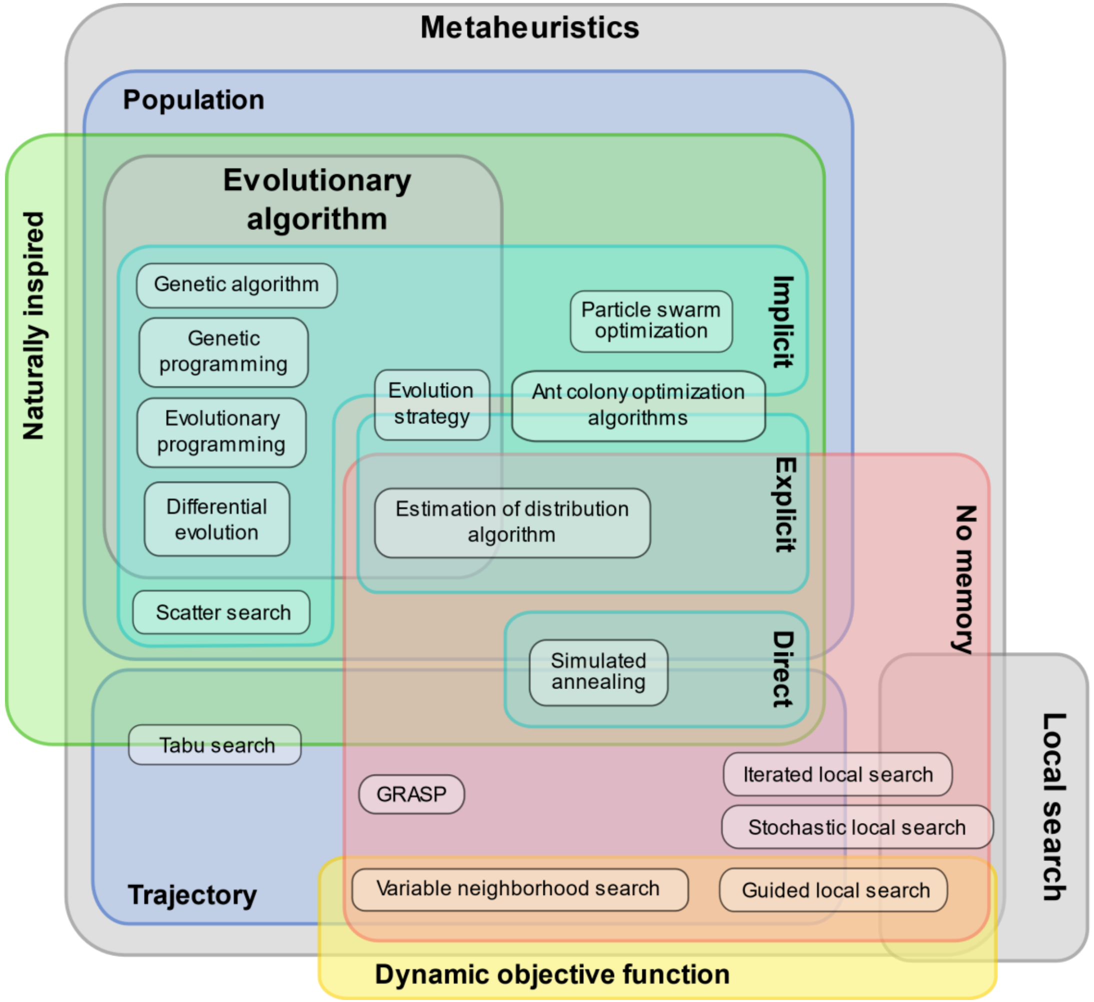
## Heuristic

- Problem-specific
- Can be used by a metaheuristic

## Metaheuristic

- Problem-independent
- Can use different heuristics or a combination of heuristics

# Metaheuristics



**“One general law, leading to the advancement of all organic beings, namely, multiply, vary, let the strongest live and weakest die.”**

- Charles Darwin, *The Origin of Species*

Reproduction (crossover)

Mutation

**“One general law, leading to the advancement of all organic beings, namely, multiply, vary, let the strongest live and weakest die.”**

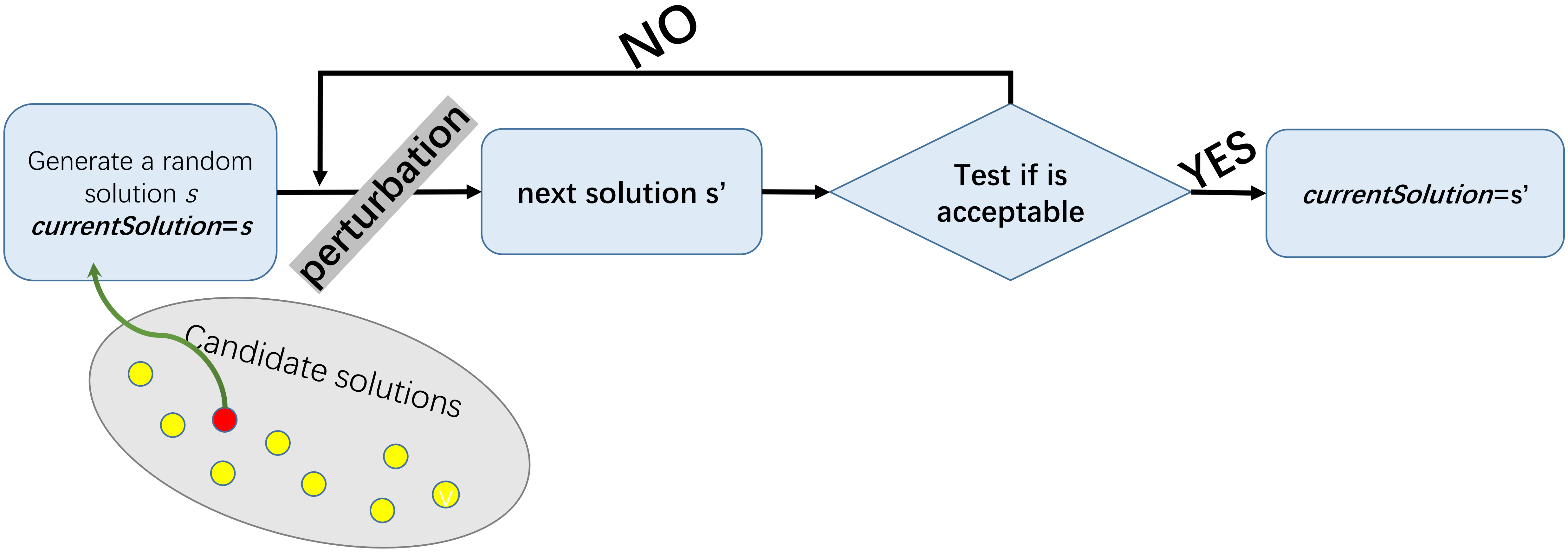
\- Charles Darwin, *The Origin of Species*

Selection

# Evolutionary Computation

- It is the study of computational systems which use ideas and get inspirations from natural evolution.
- One of the principles borrowed is ***survival of the fittest***.
- Evolutionary computation (EC) techniques can be used in optimisation, learning, and design.
- EC techniques do **not** require rich domain knowledge to use. However, domain knowledge can be incorporated into EC techniques.

# Generate-and-Test (G&T)

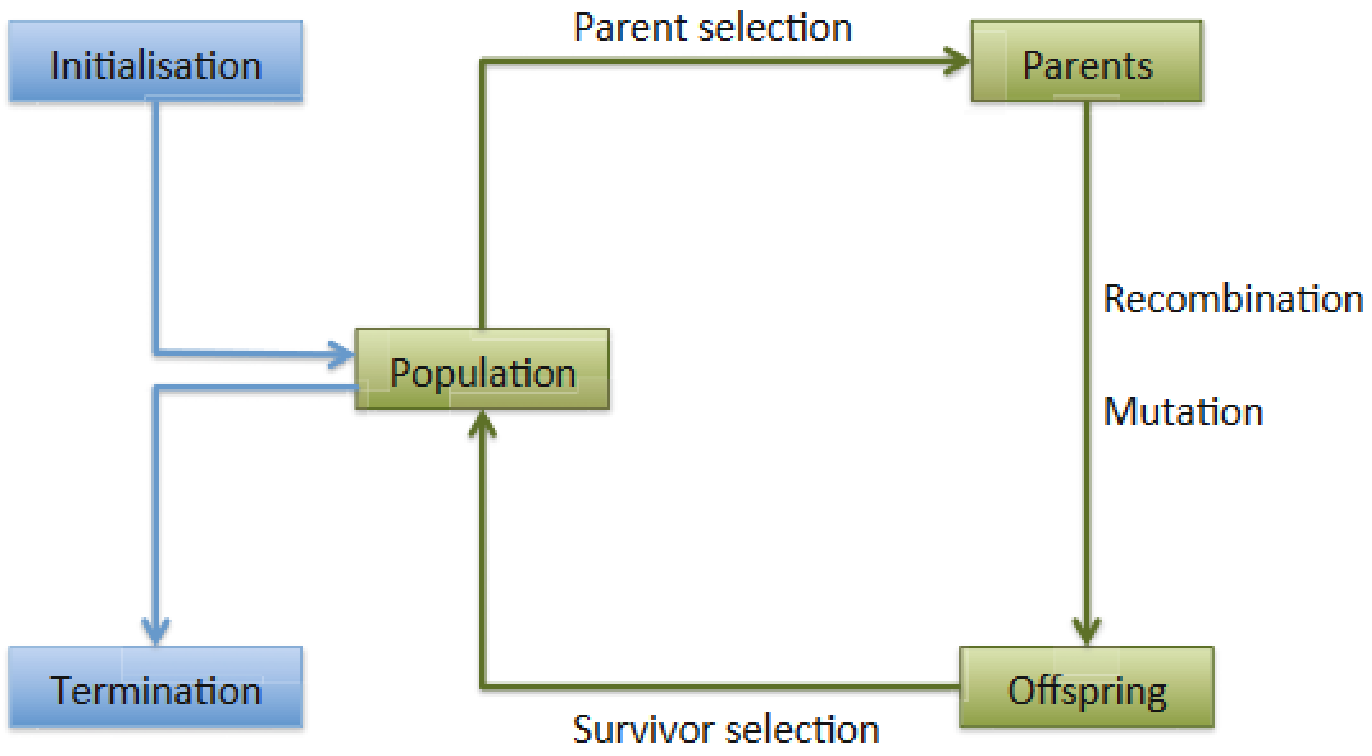


# Generate-and-Test: Steps

1. Generate the initial solution at random and denote it as the **current solution**.
2. Generate the **next solution** from the current one by **perturbation**.
3. Test whether the newly generated solution (**next solution**) is acceptable;
   1. Accepted it as the current solution **if yes**;
   2. Keep the current solution unchanged **otherwise**.
4. Go to Step 2 if the current solution is not satisfactory, stop otherwise.

# EA: Population-based G&T

- **Generate**: Mutate and/or recombine individuals in a population.
- **Test**: Select the next generation from the parents and offspring.



# A Simple Evolutionary Algorithm (EA)

Generate the initial population $P(0)$ at random

$i \leftarrow 0$ *// Generation counter*

**WHILE** halting criteria are not satisfied

**Evaluate** the fitness of each individual in $P(i)$

**Select** parents from $P(i)$ based on their fitness in $P(i)$

**Generate** offspring from the parents using **crossover** and **mutation** to form $P(i+1)$

$i \leftarrow i + 1$

***So how does this simple EA work?***

# Illustration Example

Let's use the simple **EA with population size 4** to maximise the function

$$f(x) = x^2$$

with $x$ in the *integer* interval $[0, 31]$, i.e., $x = 0, 1, \ldots, 30, 31$.

- Population size = 4 ⇔ 4 individuals/chromosomes
- So, what is an **individual** or **chromosome**?

# [Example] Encoding and decoding

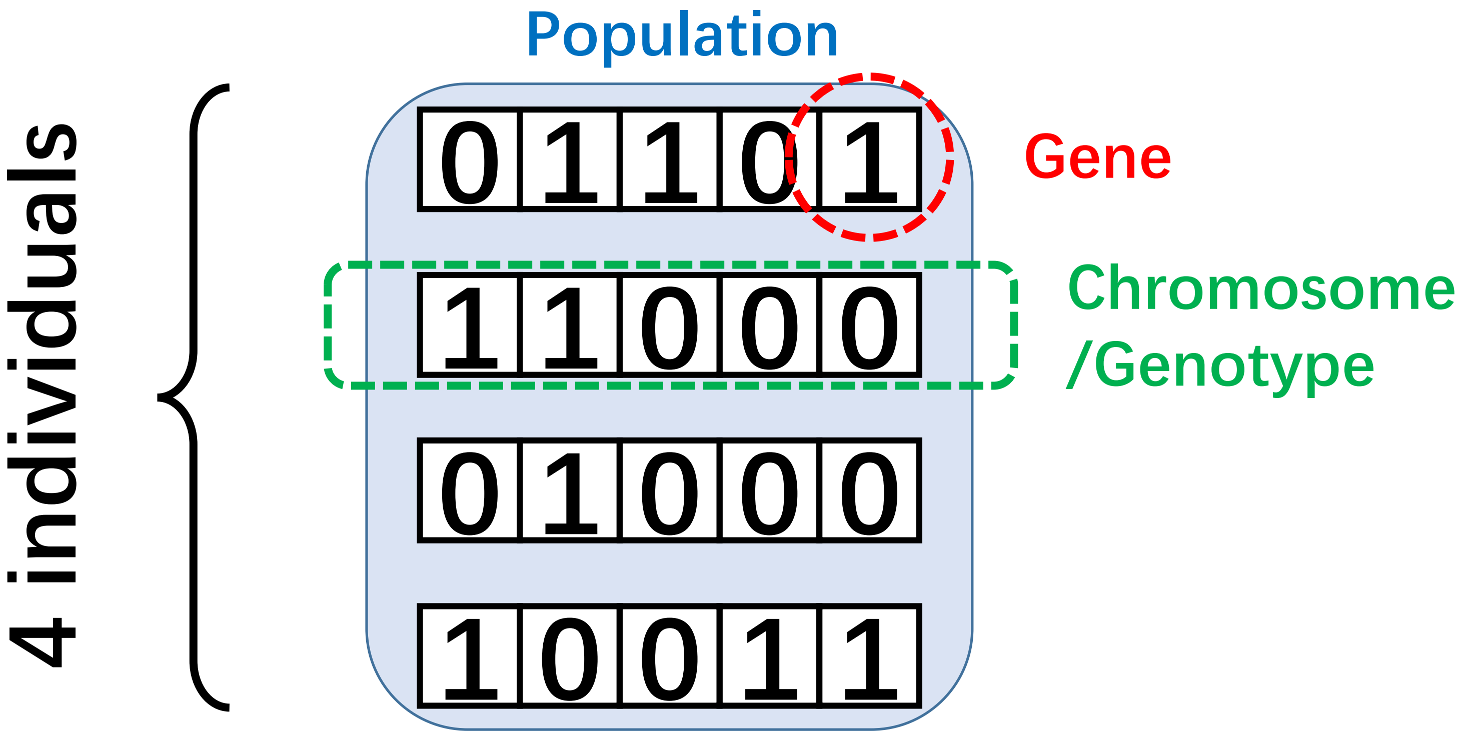
- **<u>Representation</u>**: The first step of EA applications is ***encoding*** (i.e., the representation of chromosomes).
  - We adopt binary representation for integers.
  - 5 bits are used to represent integers up to 31.
  - Examples:

$$31 \xrightarrow{\textbf{Encoding}} \boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{1} \xrightarrow{\textbf{Decoding}} 31 \xrightarrow{\textbf{Evaluation}} 961$$

$$16 \xrightarrow{\textbf{Encoding}} \boxed{1}\boxed{0}\boxed{0}\boxed{0}\boxed{0} \xrightarrow{\textbf{Decoding}} 16 \xrightarrow{\textbf{Evaluation}} 196$$

# [Example] EA: Step 1

1. **Initialisation**: Generate initial population at random, e.g., 01101, 11000, 01000, 10011. These are ***chromosomes*** or ***genotypes***.



# [Example] EA: Step 2

2. <u>**Evaluation**</u>: Calculate fitness value for each individual.

   a) **Decode** the individual into an integer (called ***phenotypes***):

   **01101 -> 13; 11000 -> 24, 01000 -> 8, 10011 -> 19;**

   b) **Evaluate** the fitness according to $f(x) = x^2$:

   $\boldsymbol{f(13) = 169, f(24) = 576, f(8) = 64, f(19) = 361}$.

# [Example] EA: Step 3-a

3. **Crossover**:
   a) Select two individuals for crossover based on their fitness. If **roulette-wheel selection** is used, then

$$P_i = \frac{f_i}{\sum_j f_j}$$

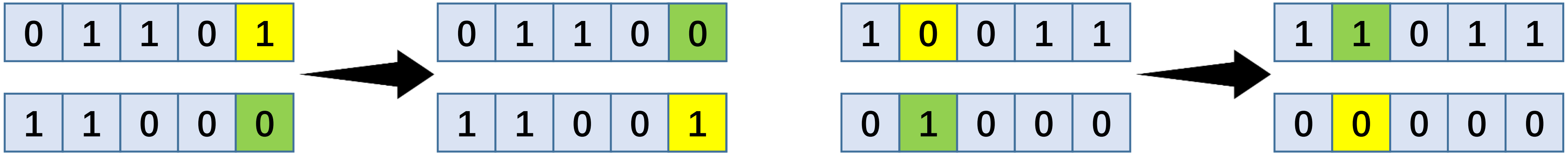
Two offspring are often produced and added to an intermediate population. Repeat this step until the intermediate population is filled.

In our example:

$$P_1(13) = \frac{169}{1170} = 0.14,\ P_2(24) = \frac{576}{1170} = 0.49,\ P_3(8) = \frac{64}{1170} = 0.06,\ P_4(19) = \frac{361}{1170} = 0.31$$

# [Example] EA: Step 3-b

b) Examples of **crossover**




Now the intermediate population is 01100, 11001, 11011, 00000.

# EA: Steps 4-5

4. Apply **mutation** to individuals in the intermediate population with a ***small*** probability. A simple mutation is bit-flipping. For example, we may have the following new population $P(1)$ after random mutation:

Example:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 | | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | | 1 | 0 | 0 | 0 | 0 |

5. Go to step 2 if not stop.

# Different Evolutionary Algorithms

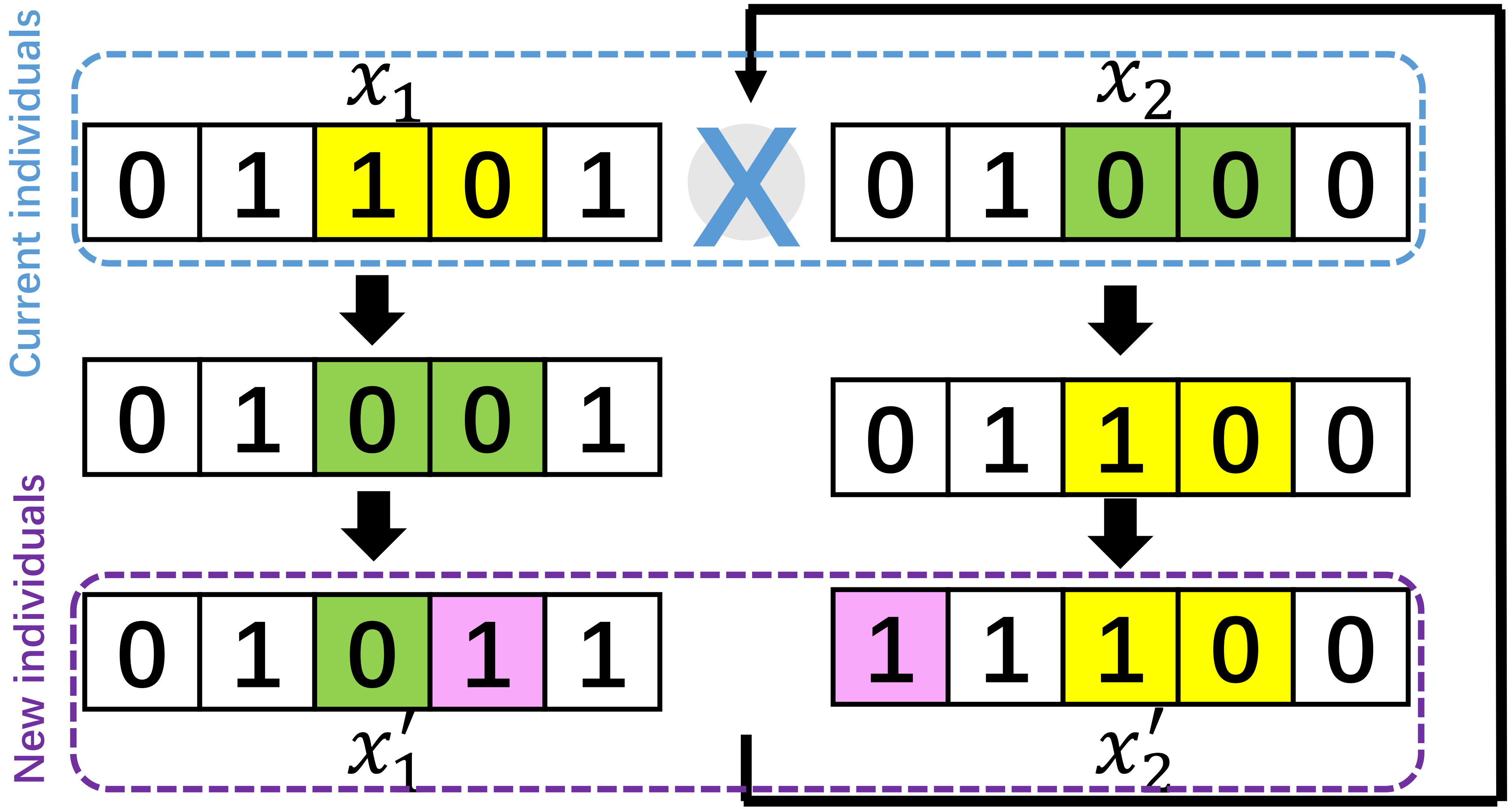
- There are several well-known EAs with different
  - historical backgrounds,
  - representations,
  - variation operators,
  - and selection schemes.

In fact, EAs refer to **a whole family of algorithms**, not a single algorithm.

# EA families

- Genetic Algorithms (GAs)
- Evolutionary Programming (EP)
- Evolution Strategies (ES)
- Genetic Programming (GP)
- …

# Genetic Algorithms (GAs)



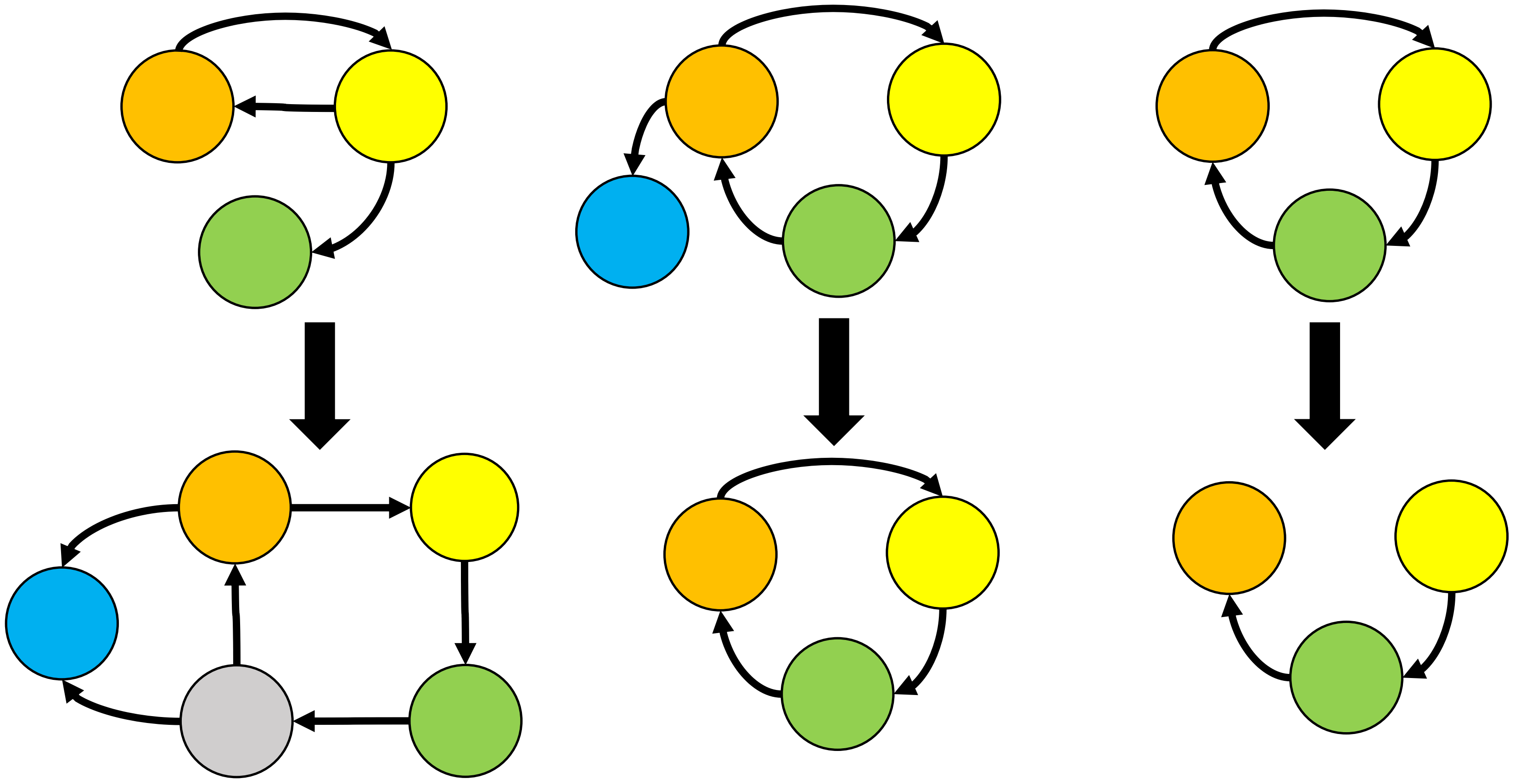
# Genetic Algorithms (GAs)

- First formulated by Holland for adaptive search and by his students for optimisation from mid 1960s to mid 1970s.
- Binary strings have been used extensively as individuals (***chromosomes***).
- Simulate **Darwinian evolution**.
- Search operators are only applied to the ***genotypic*** representation (chromosome) of individuals.
- Emphasise the role of **recombination** (***crossover***). Mutation is only used as a background operator.
- Often use **roulette-wheel** selection.

# Sketch of the simple GA

| | |
|---|---|
| Representation | Bit-strings |
| Recombination | 1-Point crossover |
| Mutation | Bit flip |
| Parent selection | Fitness proportional - implemented by Roulette Wheel |
| Survival selection | Generational |

# Evolutionary Programming (EP)



# Evolutionary Programming (EP)

- First proposed by Fogel *et al.* in mid 1960s for simulating intelligence.
- **Finite state machines** (FSMs) were used to represent individuals, although real-valued vectors have always been used in numerical optimisation.
- It is closer to **Lamarckian evolution**.
- Search operators (mutations only) are applied to the ***phenotypic*** representation of individuals.
- It does ***not*** use any recombination.
- Usually use **tournament** selection.

# Evolution Strategies (ES)

$$[x_0^{(1)}, x_1^{(1)}, \ldots, x_{d-1}^{(1)}] \qquad [x_0^{(2)}, x_1^{(2)}, \ldots, x_{d-1}^{(2)}]$$

$$[x'^{(1)}_0, x'^{(1)}_1, \ldots, x'^{(1)}_{d-1}] \qquad [x'^{(2)}_0, x'^{(2)}_1, \ldots, x'^{(2)}_{d-1}]$$

# Evolution Strategies (ES)

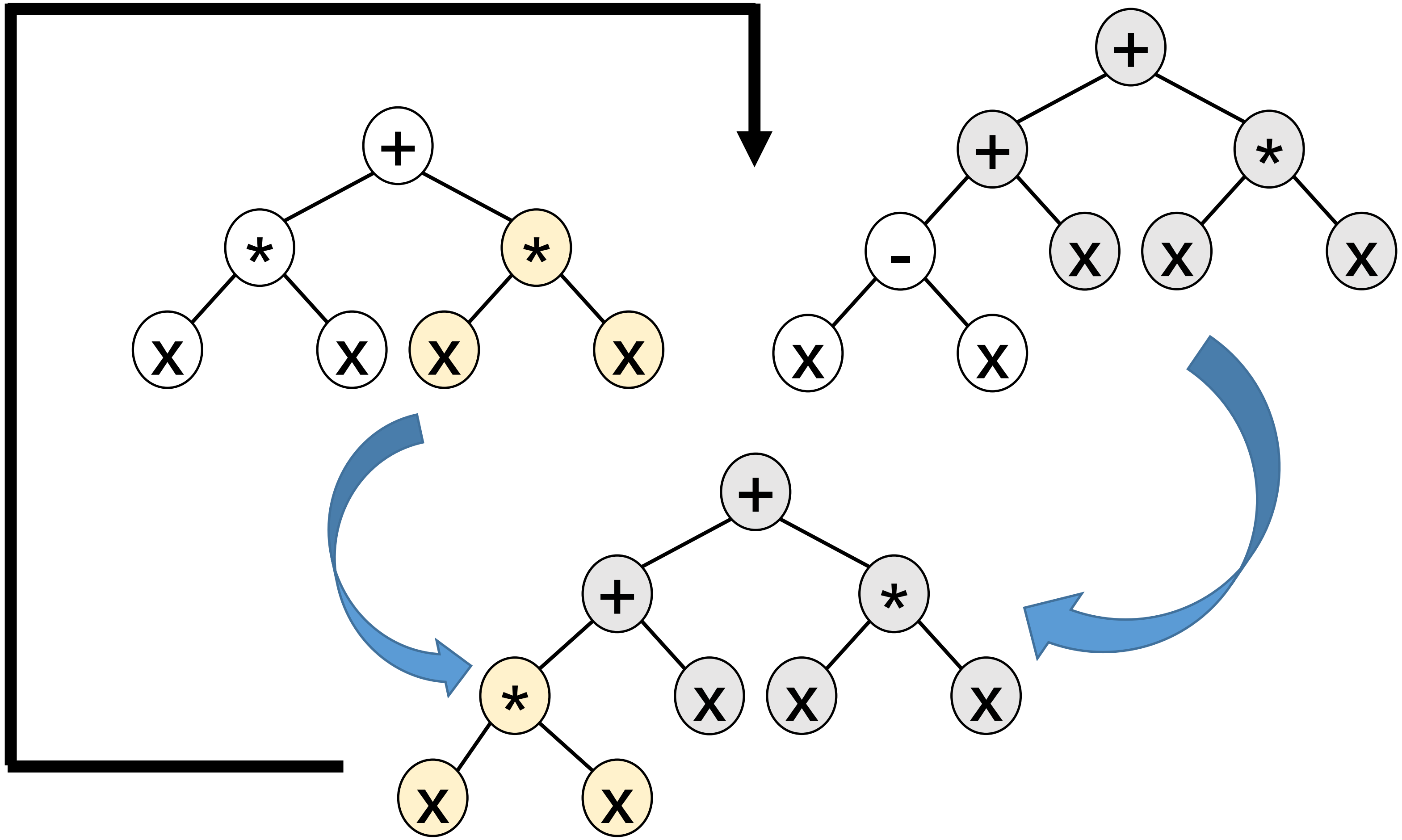
- First proposed by Rechenberg and Schwefel in mid 1960s for numerical optimisation.
- **Real-valued vectors** are used to represent individuals.
- They are closer to **Larmackian evolution**.
- They do have recombination.
- They use ***self-adaptive mutations***.

# Sketch of the simple ES

| | |
|---|---|
| Representation | Real-valued vectors |
| Recombination | Discrete or intermediary |
| Mutation | Gaussian perturbation |
| Parent selection | Uniform random |
| Survivor selection | Deterministic elitist replacement by $(\mu, \lambda)$ or $(\mu + \lambda)$ |
| Speciality | Self-adaptation of mutation step sizes |

- $\mu$: parents size
- $\lambda$: offspring size

# Genetic Programming (GP)

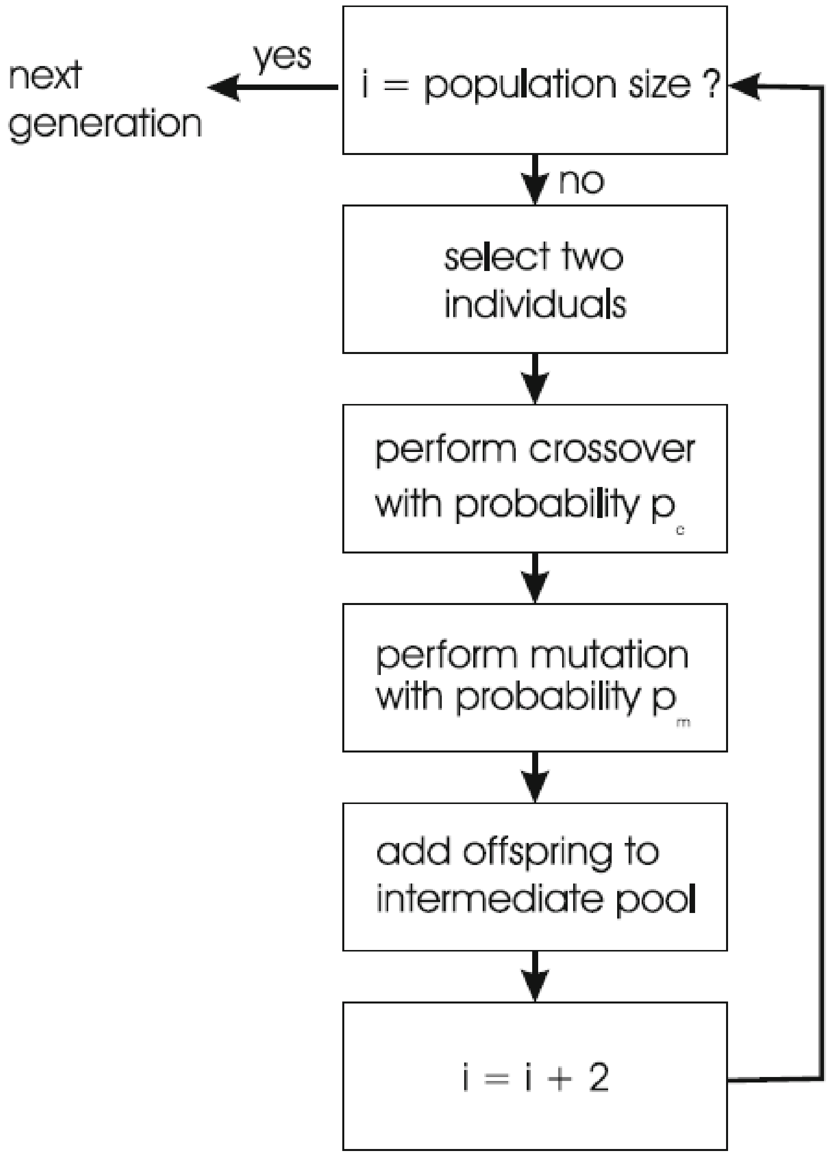


# Genetic Programming (GP)

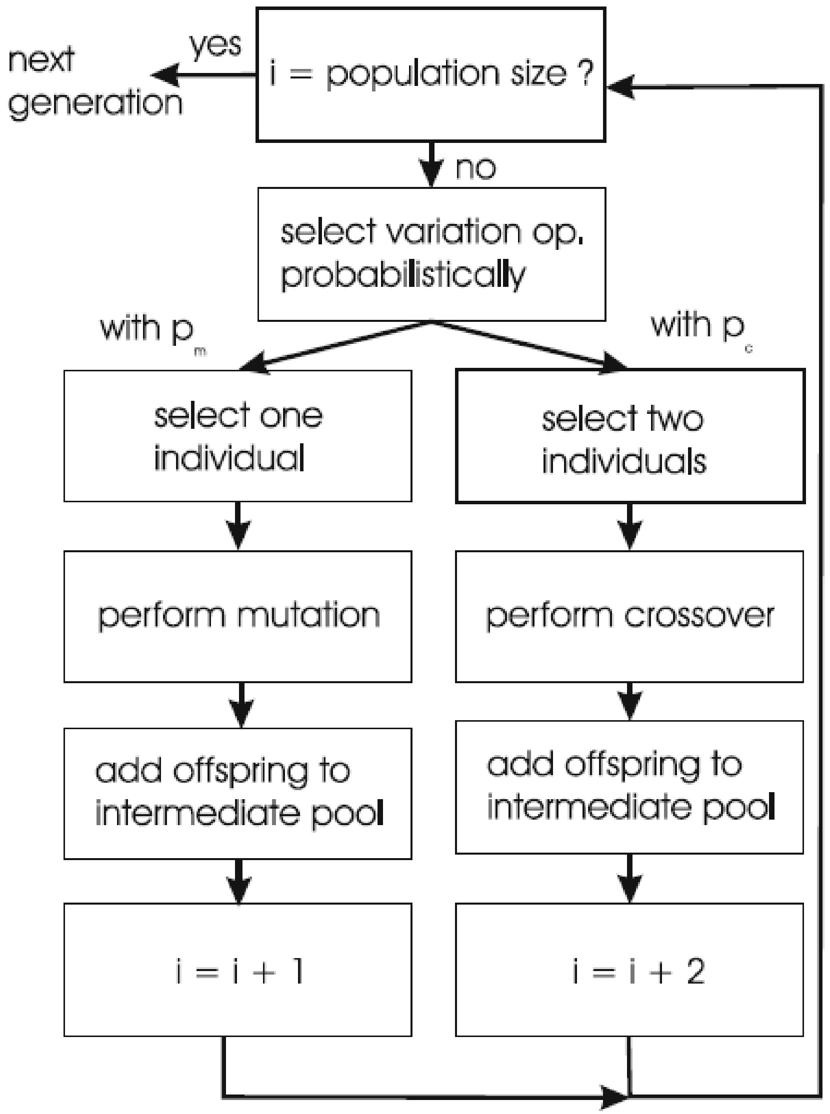
- First used by de Garis to indicate the evolution of artificial neural networks, but used by Koza to indicate the evolution of computer programs.

- **Trees** (especially Lisp expression trees) are often used to represent individuals.

- Both ***crossover*** and ***mutation*** are used.




GA loop




GP loop

# Preferred Term: Evolutionary Algorithms

- EAs face the same fundamental issues as those classical AI faces, i.e., representation, and search.

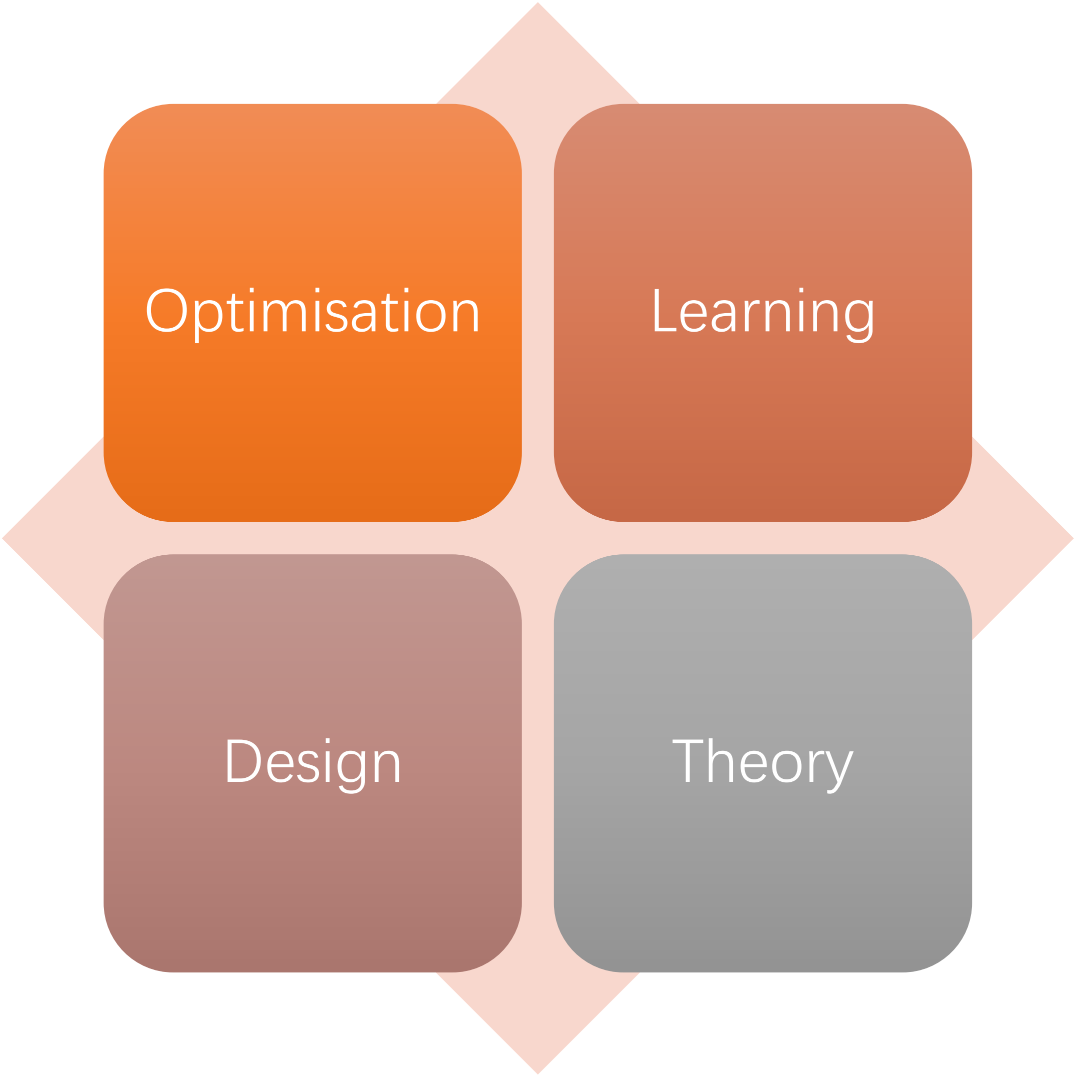
- Although GAs, EP, ES, and GP are different, they are all different variants of **population-based** **generate-and-test** algorithms. They share more similarities than differences!

- A better and more general term to use is evolutionary algorithms (EAs).

# Variations in Operators

- **Crossover/Recombination**: one-point crossover, two-point crossover, uniform crossover, intermediate crossover, etc.
- **Mutation**: bit-flipping, Gaussian mutation, Cauchy mutation, etc.
- **Selection**: roulette wheel selection (fitness proportional selection), tournament selection, rank-based selection (linear and nonlinear), etc.
- **Replacement Strategy**: generational, steady-state (continuous), etc.
- **Specialised Operators**: multi-parent recombination, inversion, order-based crossover, etc.

# Major Areas in EC



# Evolutionary Optimisation

- Numerical (global) optimisation.
- Combinatorial optimisation (of NP-hard problems).
- Mixed optimisation.
- Constrained optimisation.
- Multi-objective optimisation.
- Optimisation in a dynamic environment (with a dynamic fitness function).

# Evolutionary Learning

Evolutionary learning can be used in supervised, unsupervised and reinforcement learning.

- Learning classifier systems (Rule-based systems).
- Evolutionary artificial neural networks.
- Evolutionary fuzzy logic systems.
- Co-evolutionary learning.

# Evolutionary Design

EC techniques are particularly good at exploring unconventional designs which are very difficult to obtain by hand.

- Evolutionary design of artificial neural networks.
- Evolutionary design of electronic circuits.
- Evolvable hardware.
- Evolutionary design of (building) architectures.

# Some Applications of EA

- High-speed train head design (Japan)

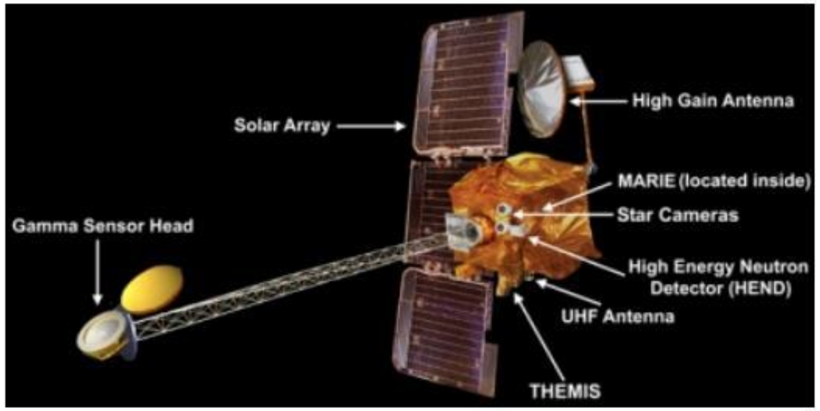
Series 700 Designed by human

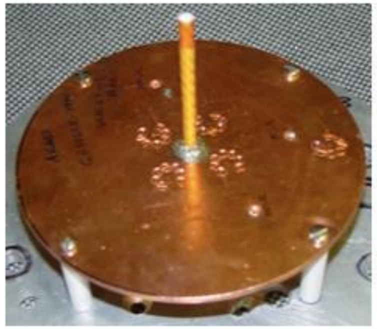
Series N700 Designed by EA

- Save 19% energy…30 increase in the output…

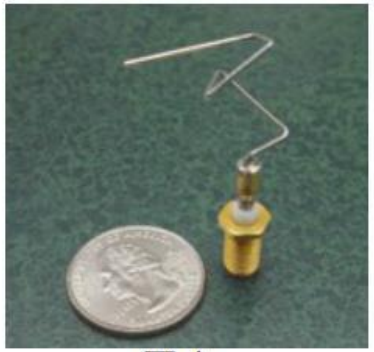
# Some Applications of EA

- X-Band Antenna Design (NASA, US)



Human

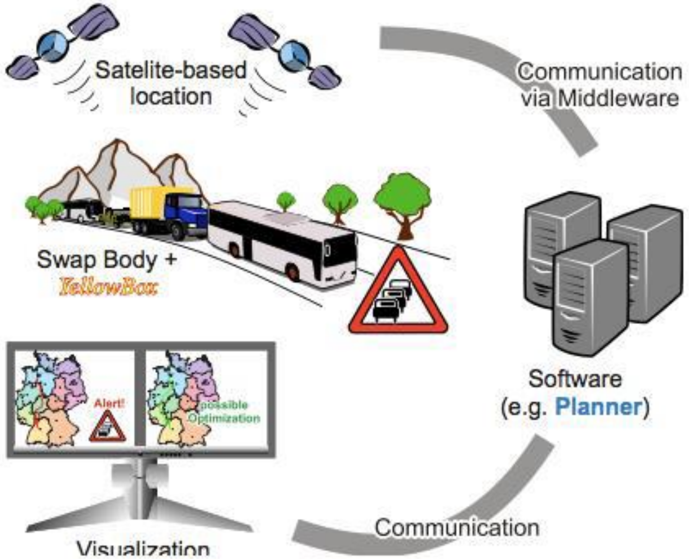


EA

- Increase efficiency from 38% to 93%

# Some Applications of EA

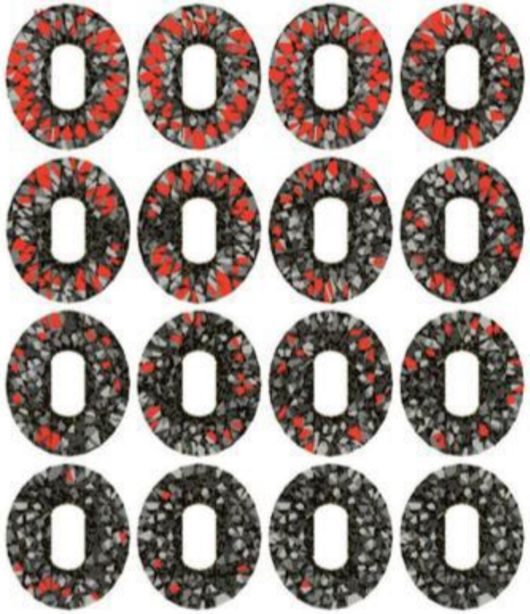
- Transportation Planning System (DHL, Germany)




- Save 9% of the transportation costs.

# Some Applications of EA

- Birds Nest (China & Switzerland)

- The irregular ordering of the beams poses an insoluble problem for the then-current CAD tools.

# Summary

- Evolutionary algorithms can be regarded as population-based generate-and-test algorithms.
- Evolutionary computation techniques can be used in optimisation, learning and design.
- Evolutionary computation techniques are flexible and robust.
- Evolutionary computation techniques are definitely useful tools in your toolbox, but there are problems for which other techniques might be more suitable.