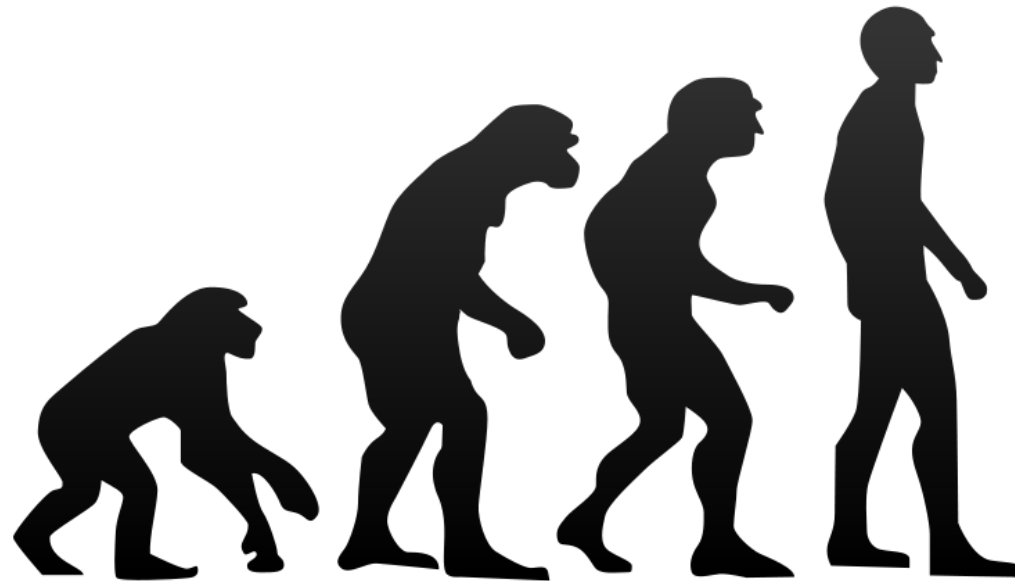# Metaheuristics I

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

Comparison
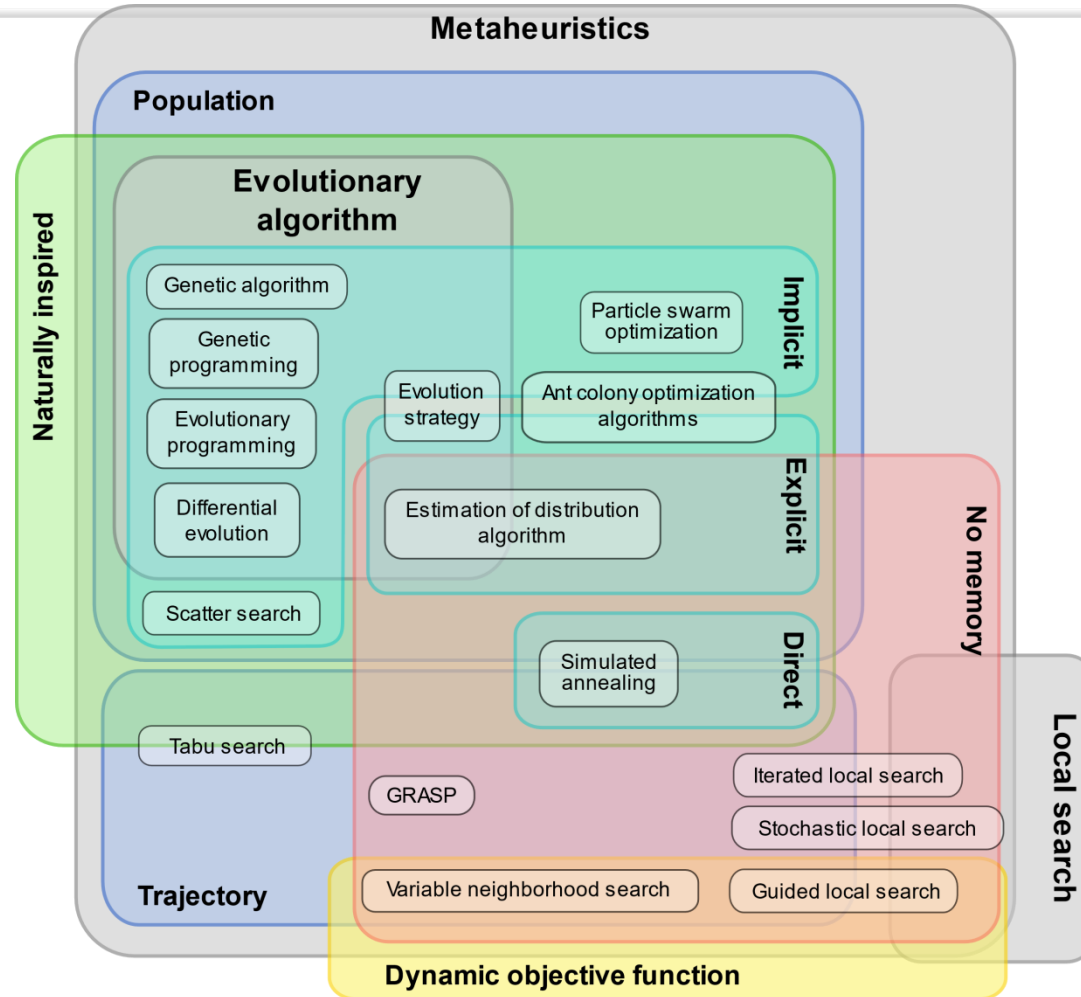
Heuristic

- Problem-specific

Metaheuristic

- Problem-independent
- It's heuristic about heuristics.

Metaheuristics

- We are not able to cover all of them in only 4 hours

- We will focus on a major family: **Evolutionary Computation**

Figure created by Canercandan (https://en.wikipedia.org/wiki/File:Metaheuristics_classification.svg).

Outline

- Why Evolutionary Computation?

- What is Evolutionary Computation?

- Different Evolutionary Algorithms

- Major Areas in Evolutionary Computation

- Summary

Why Evolutionary Computation?

Nature inspiration

Frustration with Computing Systems
（传统计算系统存在的问题）

Conventional computing systems are：

（传统计算系统有以下缺点：）

- Inflexible（不灵活）

- **Brittle**（脆弱）

- **Non-adaptive**（自适应性差）

- Poor at coping with noise（抗噪声能力低）

- **Reliant on human intervention**（太依赖于人）

- **...**

Brittle (脆弱性)

http://icdn5.digitaltrends.com/image/spilled-liquid-on-laptop-625x400.jpg

Natural Systems (自然系统)

In contrast, natural systems are often:
(对比而言，自然系统通常是:)

- **Very resilient (恢复力强)**
- **Exceptionally well adapted (自适应性极强)**
- **Always learning (总在学习)**
- **Unsupervised (无需监督的)**
- **Creative & Ingenious**
  （充满创造性&奇妙性）

Inspired by Nature（受自然启发）

Why bother (为何要向大自然借鉴)?

- **Nature, through evolution, has solved some extraordinarily complex problems (自然善于解决复杂问题)**
- **It is a highly efficient system (高效率)**
  - **By necessity (必需的)!**
- **They are (mostly) intuitively simple（通常很简单）**
- **They give (mostly) comprehensible answers（通常给出可解释的答案）**
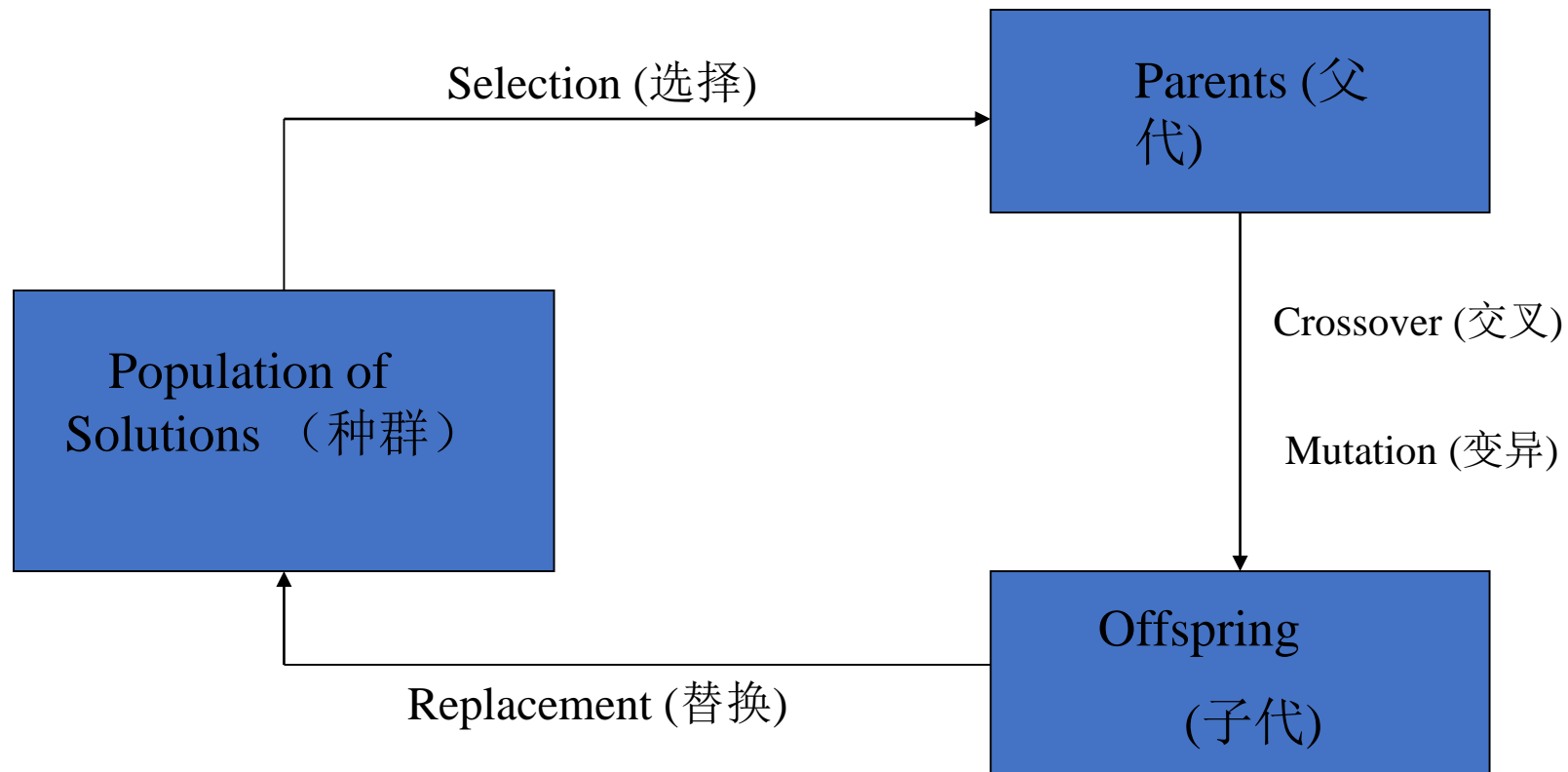
Nature Inspired Computing Techniques
(受自然启发的计算技术)

Technique（计算技术)	Inspiration (自然界灵感)
Evolutionary Algorithms (演化算法)	**Biological process of evolution (生物进化过程)**
Artificial Neural Networks (人工神经网络)	**Neurons in the brain (大脑神经元)**
Agent-based techniques (多智能体系统)	**Human social interaction (人类社交活动)**
Ant / Swarm techniques (蚁群技术)	**Social insects (群居昆虫)**
Molecular Computing (分子计算)	**Chemistry reaction (化学反应)**

What is Evolutionary Computation?

Main steps of a simple Evolutionary Algorithm

Illustrative example

Evolutionary Algorithms (演化算法)

Selection (选择) → Parents (父代)

Population of Solutions （种群）

Crossover (交叉)

Mutation (变异)

Replacement (替换)

Offspring (子代)

适者生存

"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 optimization, 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: Illustration of steps
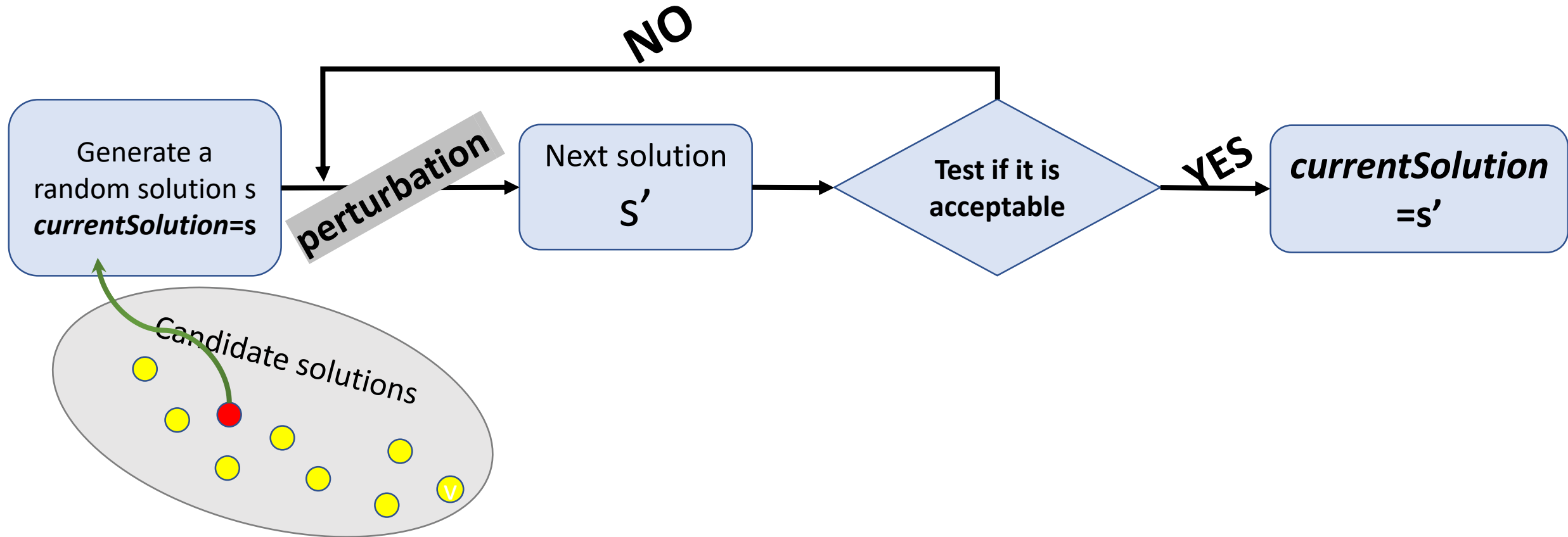
Generate-and-Test: Description of steps

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 unsatisfactory, stop otherwise.

EAs as Population-based Generate-and-Test

- **Generate**: Mutate and/or recombine individuals in a population.
- **Test**: Select the next generation from the parents and offspring.
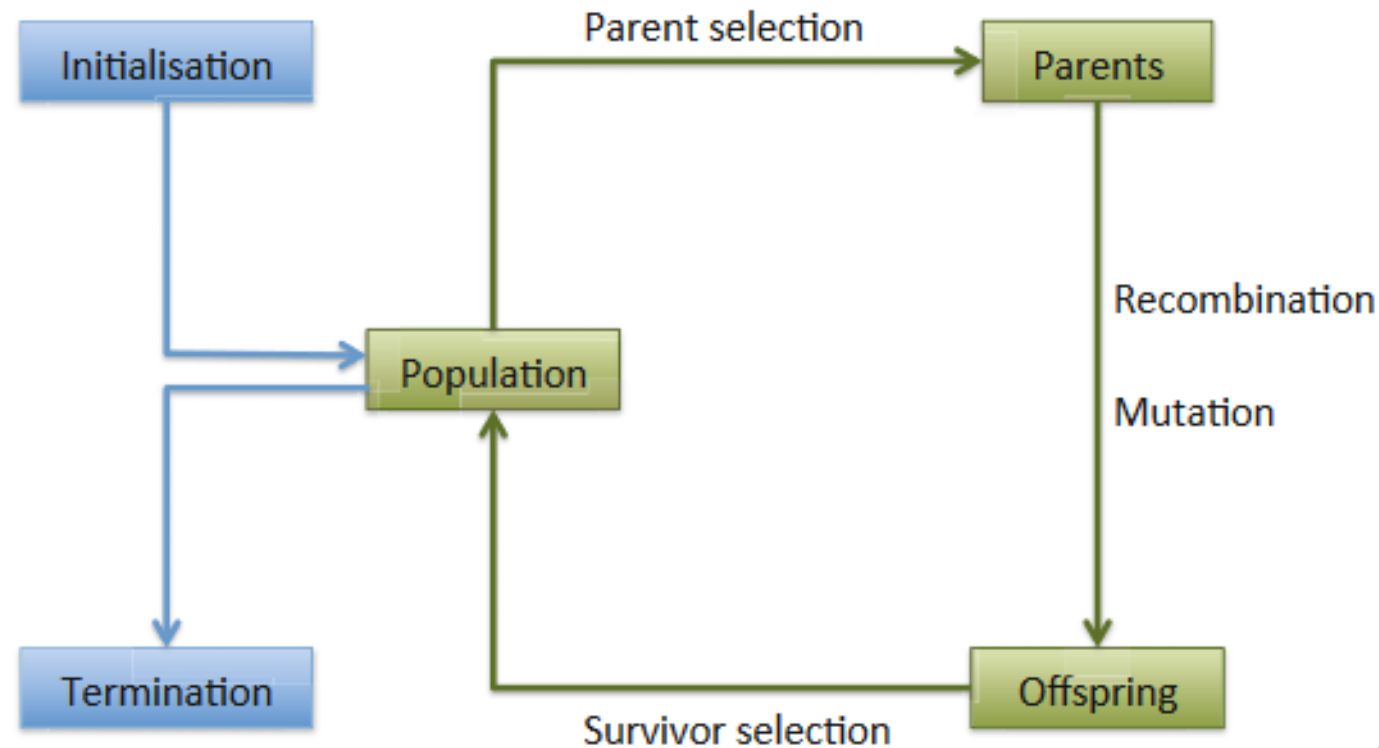
Figure from [2].

A Simple Evolutionary Algorithm (EA)

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

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

3   **WHILE** halting criteria are not satisfied

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

3.2 Select parents form $P(i)$ based on their fitness in $P(i)$

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

3.4 $i \leftarrow i + 1$

So how does this simple EA work?

[Example] How Does the Simple EA Work?

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

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

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

- Population size = 4 $\Leftrightarrow$ 4 individuals/chromosomes
- So, what is an individual or chromosome?

[Example] Encoding and decoding

- **Representation**: 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:

1	1	1	1	1

31 →(Encoding)→ [1 1 1 1 1] →(Decoding)→ 31 →(Evaluation)→ 961

1	0	0	0	0

16 →(Encoding)→ [1 0 0 0 0] →(Decoding)→ 16 →(Evaluation)→ 256
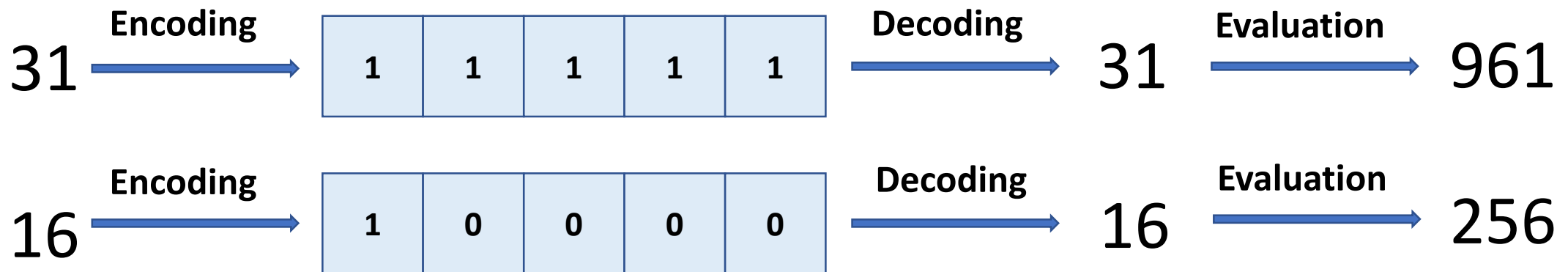
[Example] EA: Step 1

1. **Initialisation**: Generate the initial population at random, e.g., 01101, 11000, 01000, 10011. These are *chromosomes* or *genotypes*.
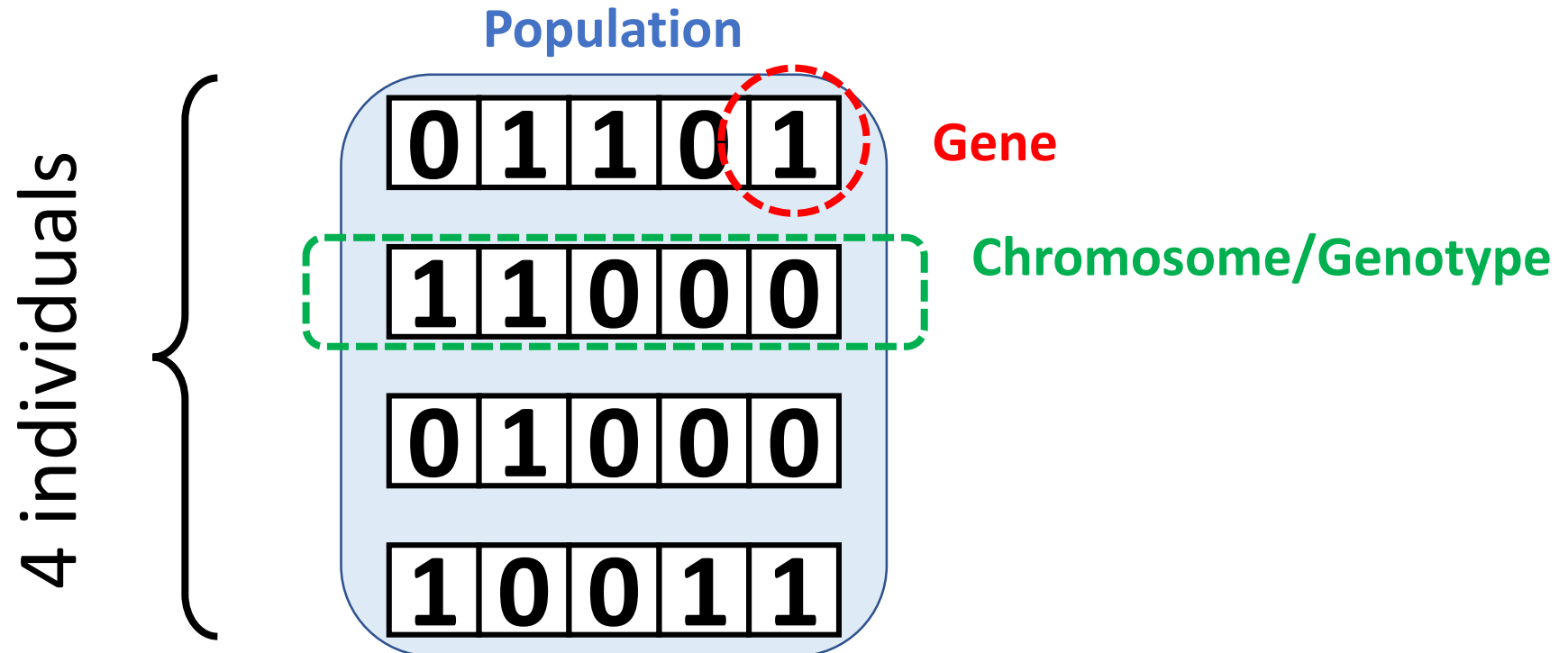
[Example] EA: Step 2

2. **Evaluation**: 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 $f(x) = x^2$:

$$f(13) = 169, f(24) = 576, f(8) = 64, f(19) = 361.$$

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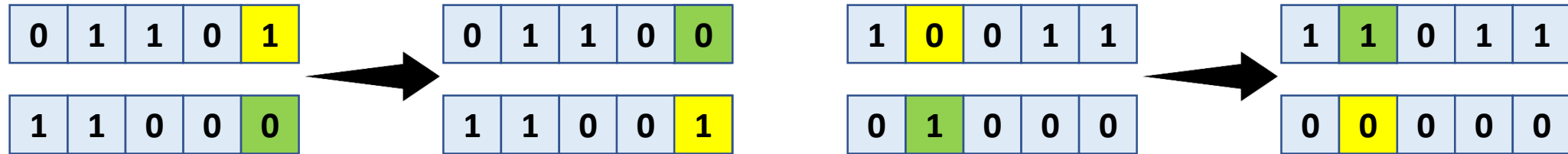
Now the intermediate population is 01100, 11001, 11011, 00000.

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:

$$01101, 11001, 10000, 11011$$

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

Genetic Algorithms (GAs)

Evolutionary Programming (EP)

Evolution Strategies (ES)

Genetic Programming (GP)

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)

- First formulated by Holland [6] for adaptive search and by his students for optimization 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.

Genetic Algorithms (GAs)
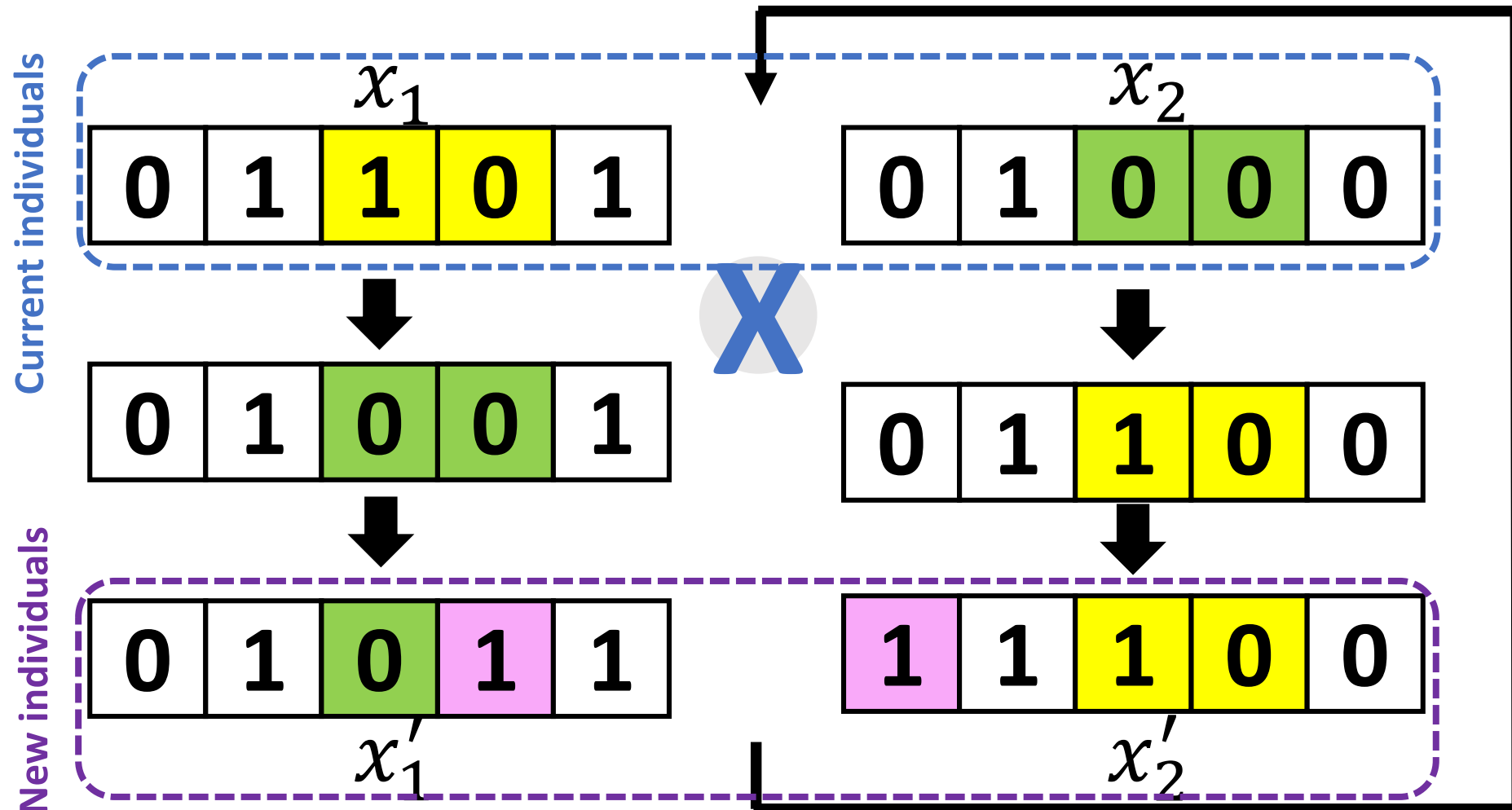
Evolutionary Programming (EP)

- First proposed by Fogel *et al.*[7] 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 optimization.

- It is closer to Lamarckian (拉马克) evolution.

- Search operators (mutations only) are applied to the *phenotypic* representation of individuals.

- It does *not* use any recombination.

Evolutionary Programming (EP)
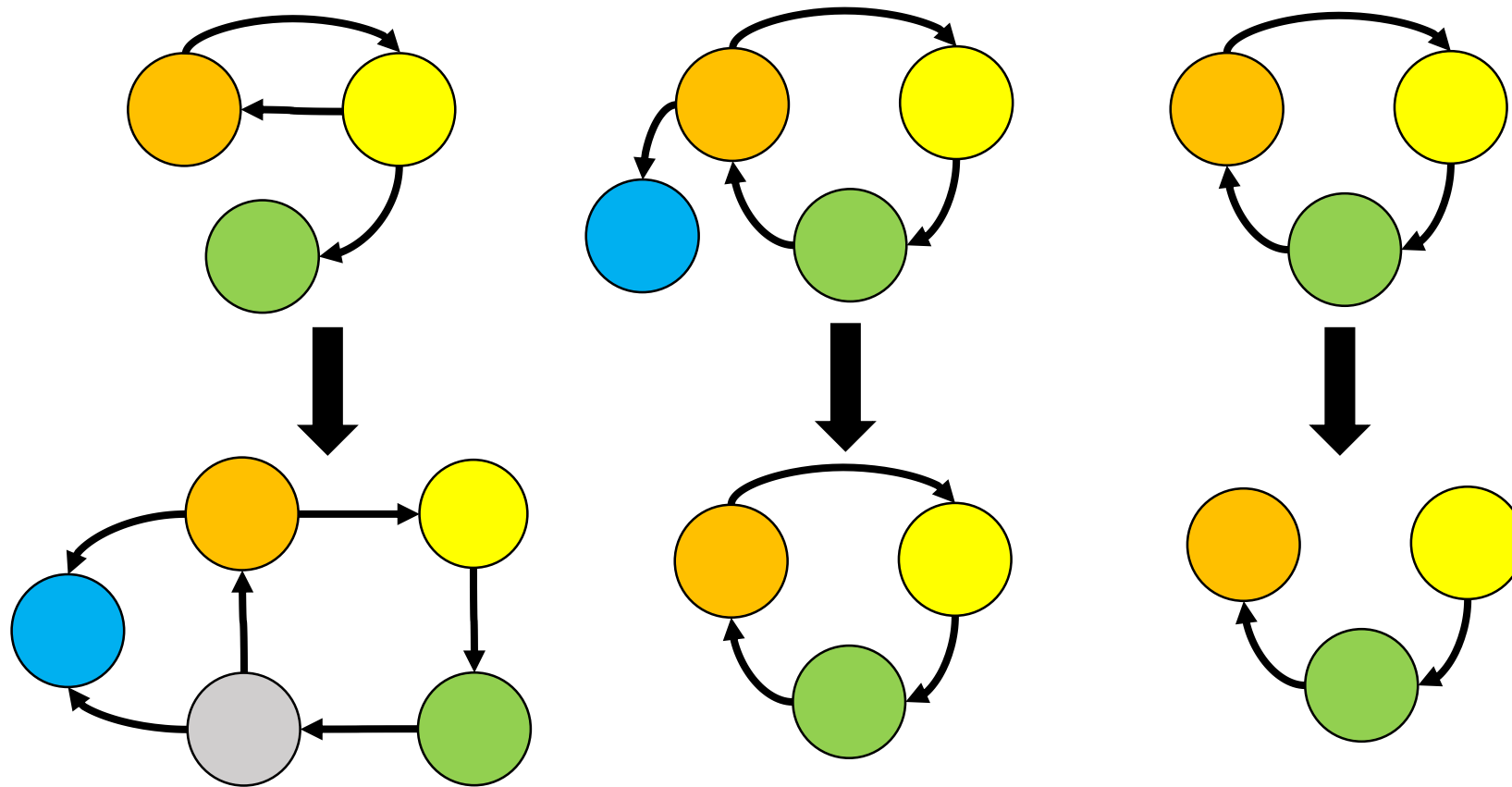
Evolution Strategies (ES)

- First proposed by Rechenberg and Schwefel in mid 1960s for numerical optimization.

- Real-valued vectors are used to represent individuals.

- They are closer to Larmackian (拉马克) evolution.

- They do have recombination.

- They use *self-adaptive mutations*.
  - Mutation is performed by adding a normally distributed random value to each component.

Evolution Strategies (ES)

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

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

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

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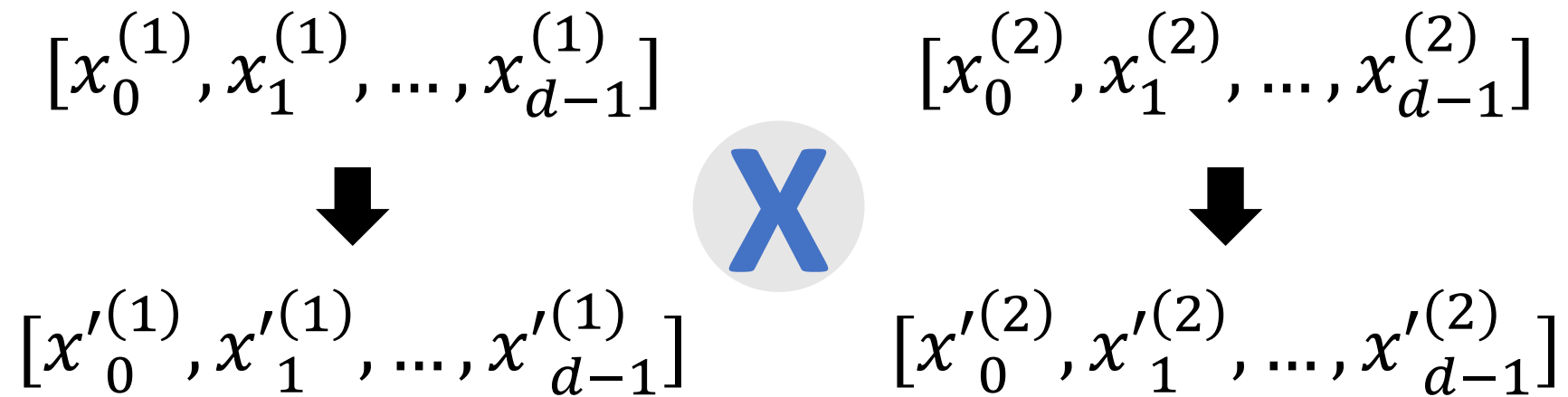
Genetic Programming (GP)

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

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

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

Genetic Programming (GP)
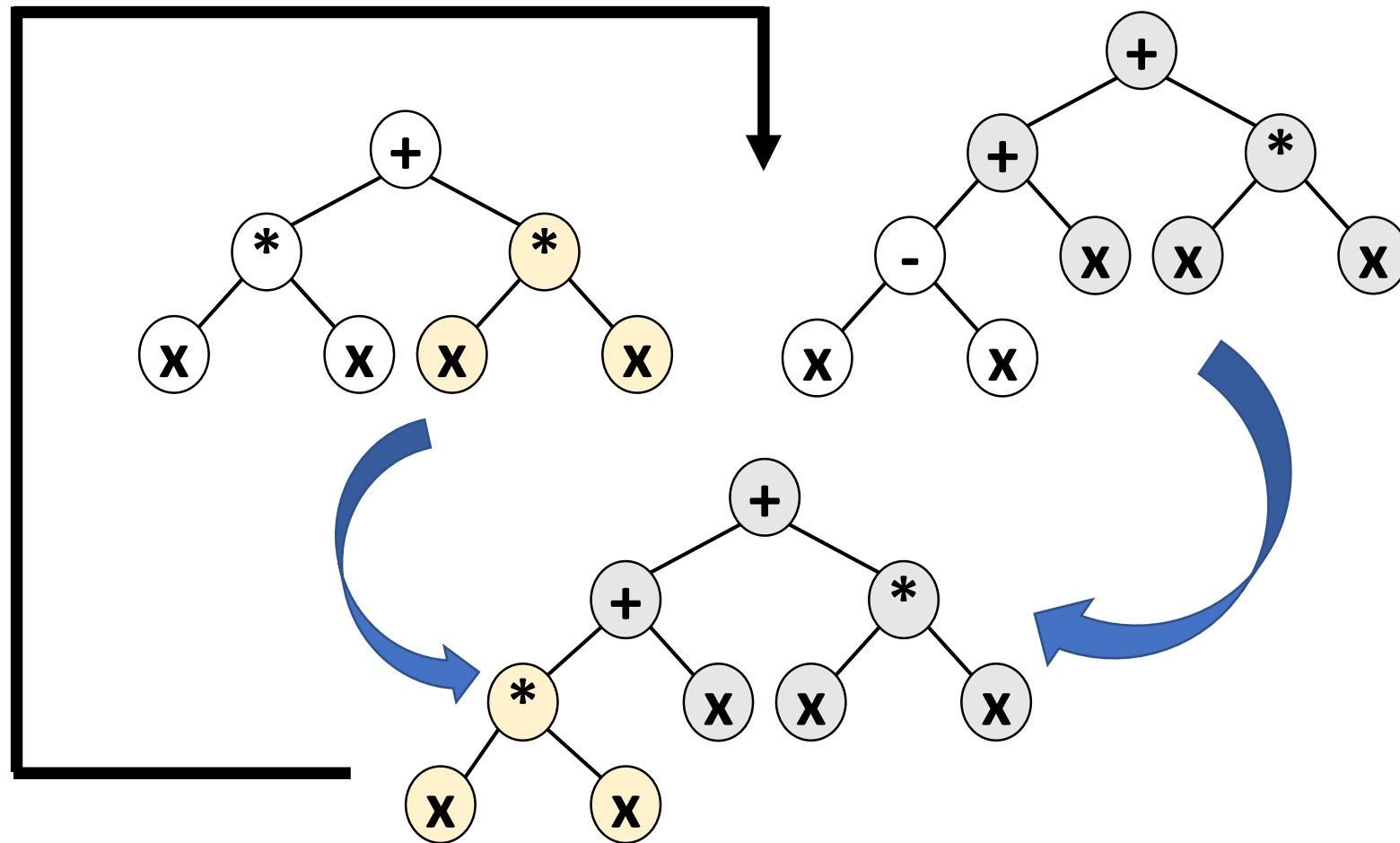
Preferred Term: Evolutionary Algorithms

- 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).

Major Areas in Evolutionary Computation

Optimization

Learning

Creative Design

Theory
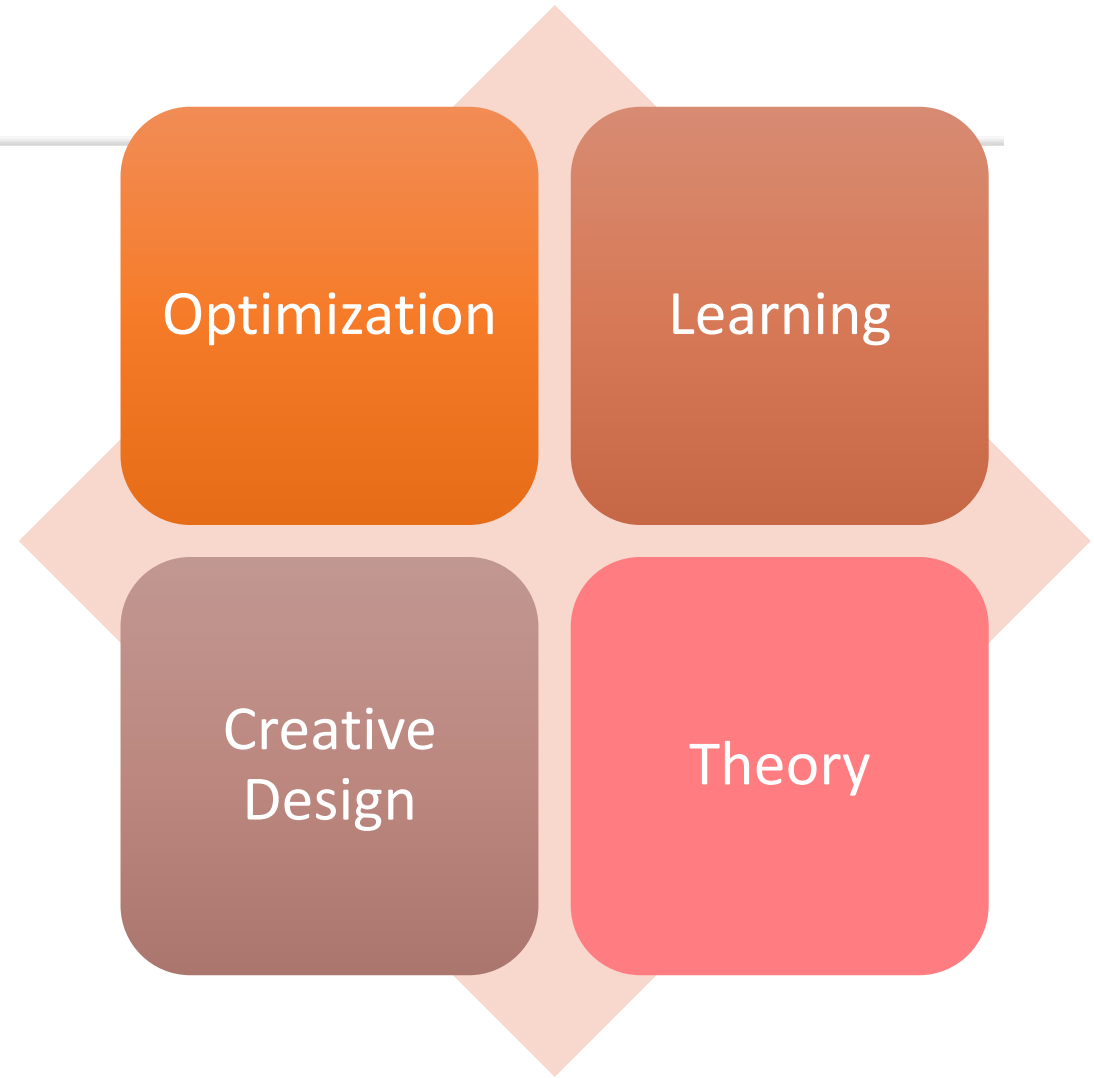
Evolutionary Optimization

- Numerical (global) optimization.

- Combinatorial optimization (of NP-hard problems).

- Mixed optimization.

- Constrained optimization.

- Multiobjective optimization.

- Optimization in a dynamic environment (with a dynamic fitness function).

Gritter Routing

- **Optimization Problem:**
  - **Multiple trucks, multiple depots,**
  - **different capacities**
  - **Routing constraints**
  - *Dynamic environment*

H. Handa, L. Chapman and Xin Yao, ``Robust route optimisation for gritting/salting trucks: A CERCIA experience,'' IEEE Computational Intelligence Magazine, 1(1):6-9, February 2006.

Optimized Routing: An Example

演化算法已被应用于国民经济的重大领域
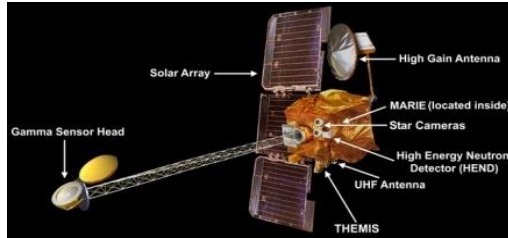
航天：天线设计（NASA，奥德赛号飞行器UHF天线）

G. S. Hornby et al., Automated antenna design with evolutionary algorithms.
American Institute of Aeronautics and Astronautics, 2006.

物流：物流配送系统（DHL公司The in.west System）

Thomas Weise, Alexander Podlich, Kai Reinhard, Christian Gorldt, and Kurt Geihs (2009):
"Evolutionary Freight Transportation Planning," in Applications of Evolutionary Computing

建筑：大型桁架结构设计（如：鸟巢、水立方）

Ludger Hovestadt. Beyond the Grid - Architecture and Information Technology.
Applications of a Digital Architectonic. Birkhäuser Basel / Boston 2009.

机器人：仿生机器人

Zykov V., Mytilinaios E., Adams B., Lipson H. (2005) "Self-reproducing machines",
Nature Vol. 435 No. 7038, pp. 163-164

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.

- Automatic modularization of machine learning systems by speciation and niching.

奇偶校验问题的前向神经网络：人的设计

Layered bipolar neural network with one hidden layer for the parity-8 problem.
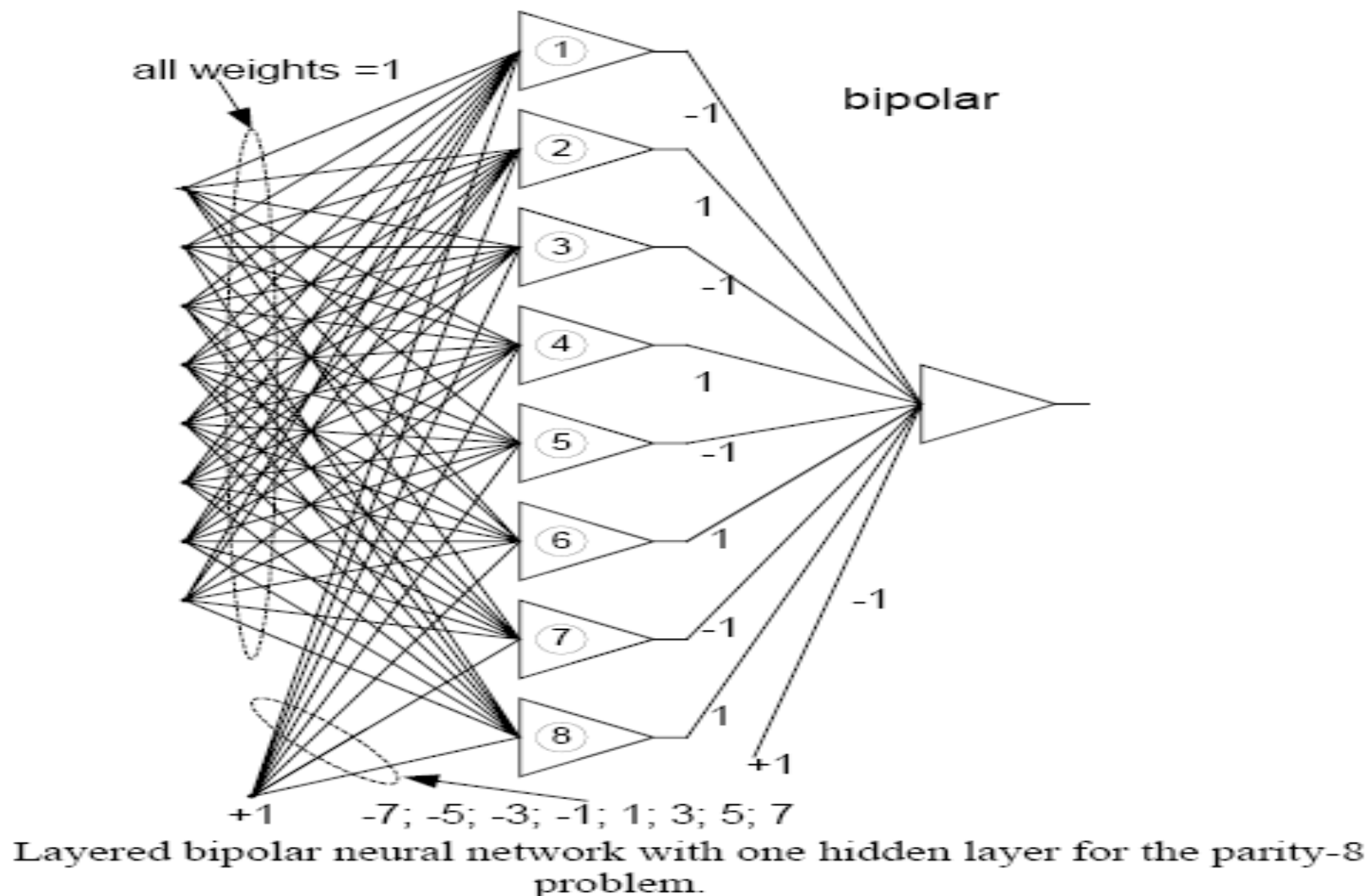
- **B. Wilamowski, D. Hunter, and A. Malinowski, "Solving Parity-N Problems with Feedforward Neural Networks", DOI: 10.1109/IJCNN.2003.1223966, Proc. of IJCNN'03.**

反过来想一想

- **如果不是靠人设计呢?**
- **如果让演化过程去*自动发现*最佳的神经网络,这样的神经网络会是怎样的呢?**

The 9-Parity Problem — Architectures
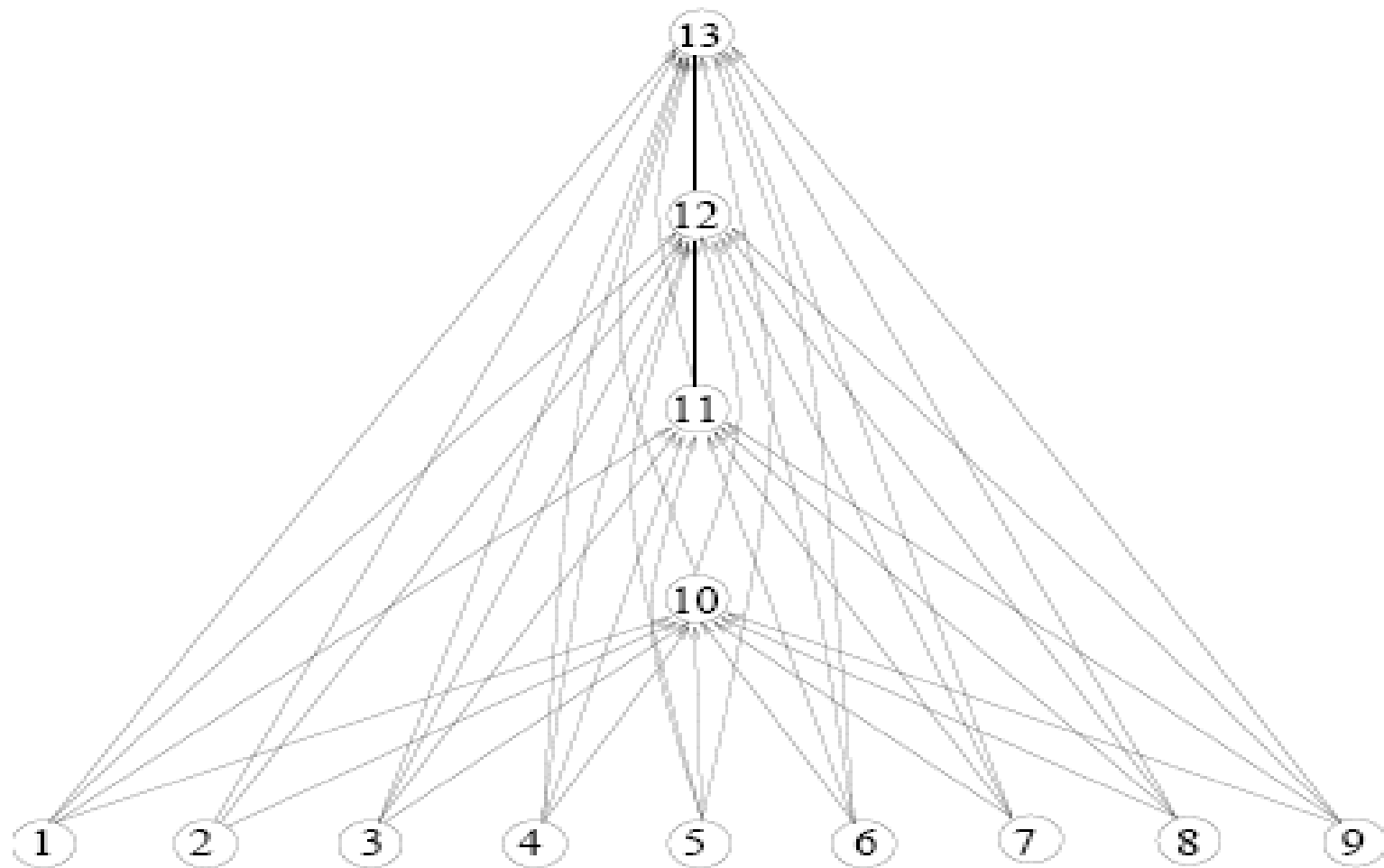
The 9-Parity Problem — Weights and Biases

	T	1	2	3	4	5	6
10	-12.35	-12.19	12.38	-12.19	-12.20	12.23	-12.19
11	-4.12	6.04	0	6.04	6.04	-6.34	6.04
12	7.05	6.78	-7.13	6.78	6.79	-6.96	6.79
13	0	7.89	-7.76	7.89	7.89	-8.04	7.89

	7	8	9	10	11	12
10	12.23	-12.12	-12.20	0	0	0
11	-6.34	-26.16	6.05	0	0	0
12	-6.96	-9.59	6.79	26.70	-16.45	0
13	-8.04	-10.71	7.89	30.71	-18.99	-17.02

二者很不一样哦

- **演化的神经网更紧凑, 神经元个数很少.**
- **演化的网络层数较多, 但捷径（short-cuts）更多.**
- **占用的资源少, 更有效.**

- **X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, 87(9):1423-1447, September 1999. (Won the 2001 IEEE Donald G. Fink Prize Paper Award)**

What can EC do ...

← Formula-1 Racing teams (F1赛车团队)?

Investors and traders (投资者与交易者)?

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.

What can EC do …

…artists and designers (艺术家和设计师)

- **Evolutionary art from Andrew Rowbottom**
- **Genetic art by Peter Kleiweg**
- **Organic art by William Latham**
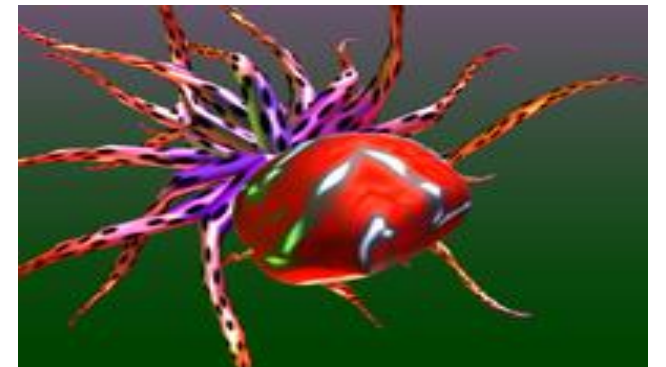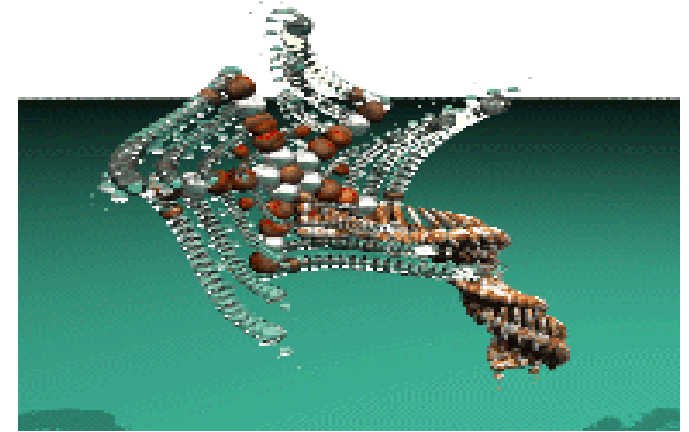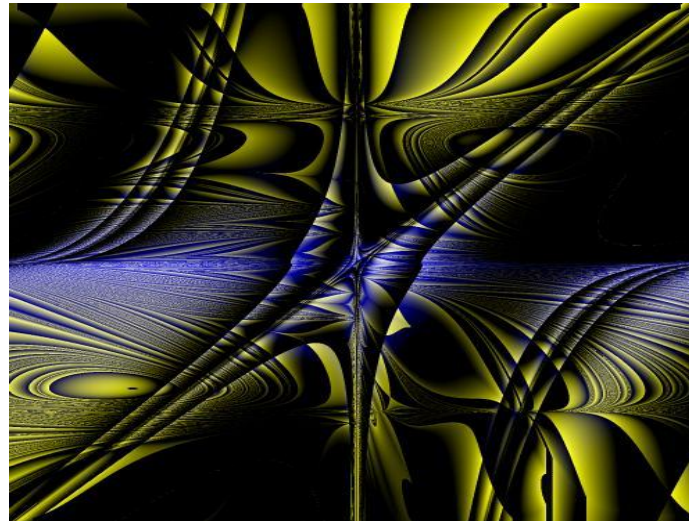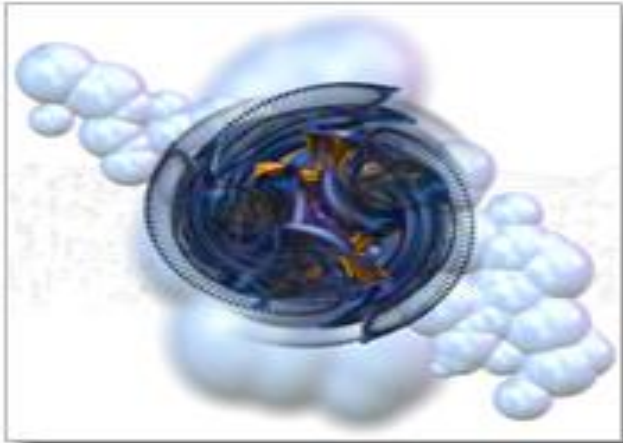
Summary

- Evolutionary algorithms can be regarded as population-based generate-and-test algorithms.

- Evolutionary computation techniques can be used in optimization, learning and creative 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.

Reading Materials

Books

[b1] Eiben, A. E., & Smith, J. E. (2003). *Introduction to evolutionary computing* (Vol. 53). Berlin: springer. (pages 1-48)

Articles

[1] K. Chellapilla, *Combining mutation operators in evolutionary programming*, IEEE Transactions on Evolutionary Computation, 2(3):91-96, Sept. 1998.

[2] Yao, X., Liu, Y., & Lin, G. (1999). *Evolutionary programming made faster.* IEEE Transactions on Evolutionary computation, 3(2), 82-102. K. H.

[3] T. Schnier and X. Yao, *Using Multiple Representations in Evolutionary Algorithms*, Proceedings of the 2000 Congress on Evolutionary Computation, IEEE Press, Piscataway, NJ, USA, July 2000. pp.479-486.

[4] Liang, X. Yao and C. S. Newton, *Adapting self-adaptive parameters in evolutionary algorithms*, Applied Intelligence, 15(3):171-180, November/December 2001.

[5] Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). *Optimization by simulated annealing.* science, 220(4598), 671-680.

[6] Holland, J. H. (1992). *Genetic algorithms.* Scientific American, 267(1), 66-73.

[7] Fogel, D. B., Fogel, L. J., & Atmar, J. W. (1991, November). *Meta-evolutionary programming.* In Signals, systems and computers, 1991. 1991 Conference record of the twenty-fifth Asilomar Conference on (pp. 540-545). IEEE.