Lecture 1 - Introduction to Evolutionary Computation

Yuhui Shi

CSE, SUSTech

Outline of This Lecture

- Why Natural Computation?
- What is Evolutionary Computation?
- Different Types of Evolutionary Algorithms
- Major Areas in Evolutionary Computation
- Summary of this Lecture
- Reading List

Outline of This Lecture

- **Why Natural Computation?**
- What is Evolutionary Computation?
- Different Types of Evolutionary Algorithms
- Major Areas in Evolutionary Computation
- Summary of this Lecture
- Reading List

Current Computing Systems

- Rigid (死板的) and inflexible (不灵活)
- Brittle (脆弱的)
- 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)
- Boring
- Doesn't learn and generalise
- Not autonomous
- Slow
- ...

Brittle (脆弱性)

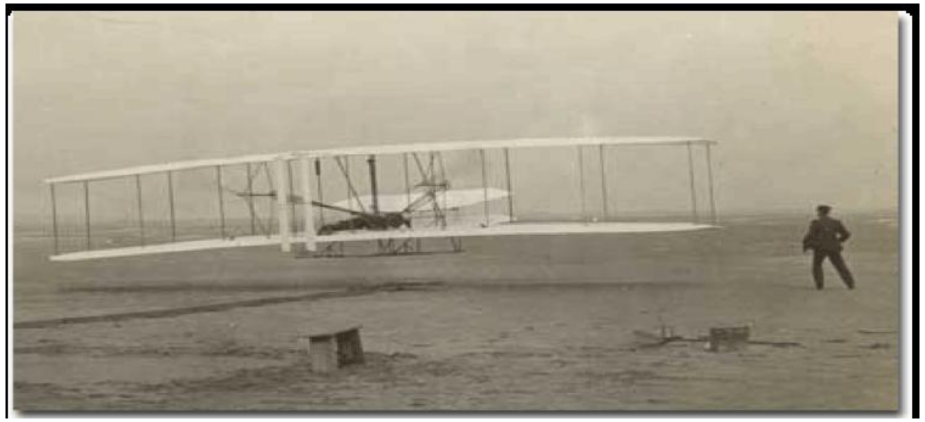
Figure 1: Image source: http://icdn5.digitaltrends.com/image/spilled-liquid-on-laptop-625x400.jpg

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)

Inspiration, Not Copying
Nature Inspired Computation



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)

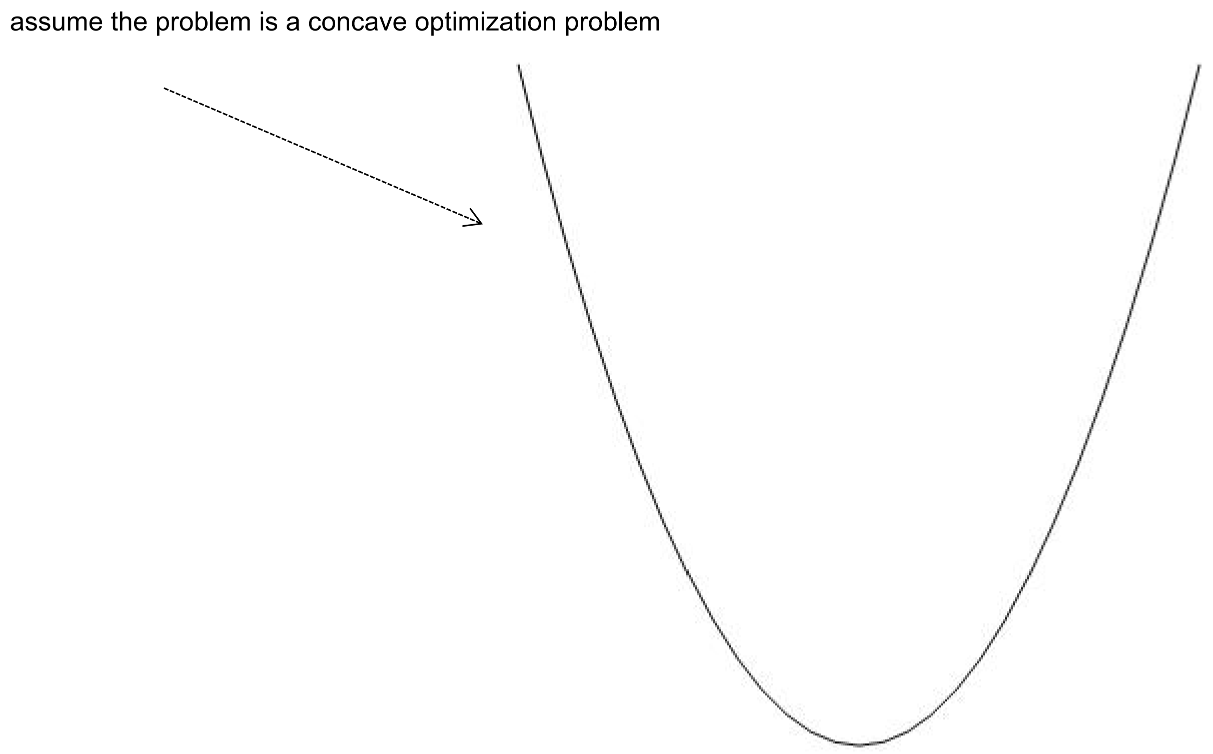
Nature Inspired Computing Techniques (受自然启发的计算技术)

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)
Swarm techniques (群体智能)	Social insects (群居昆虫)
Quantum computing (量子计算)	Quantum physics (量子物理)

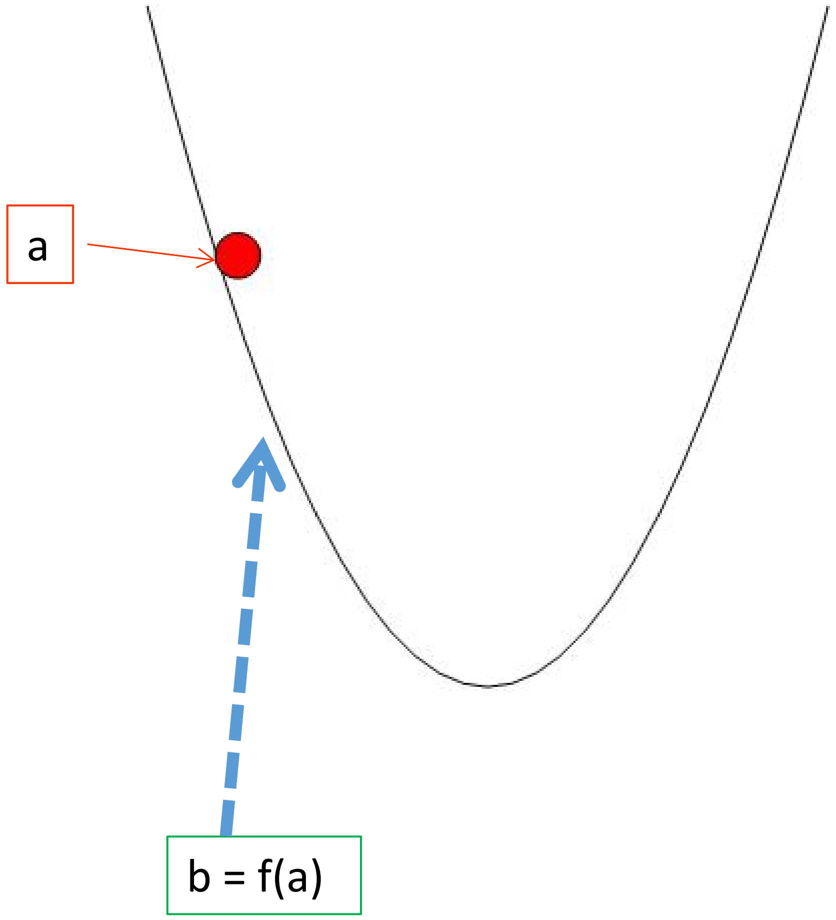
Outline of This Lecture

- Why Natural Computation?
- **What is Evolutionary Computation?**
- Different Types of Evolutionary Algorithms
- Major Areas in Evolutionary Computation
- Summary of this Lecture
- Reading List

Why Evolutionary Computation - Intutively



Why Evolutionary Computation - Intutively



Why Evolutionary Computation - Intutively

Method of steepest descent

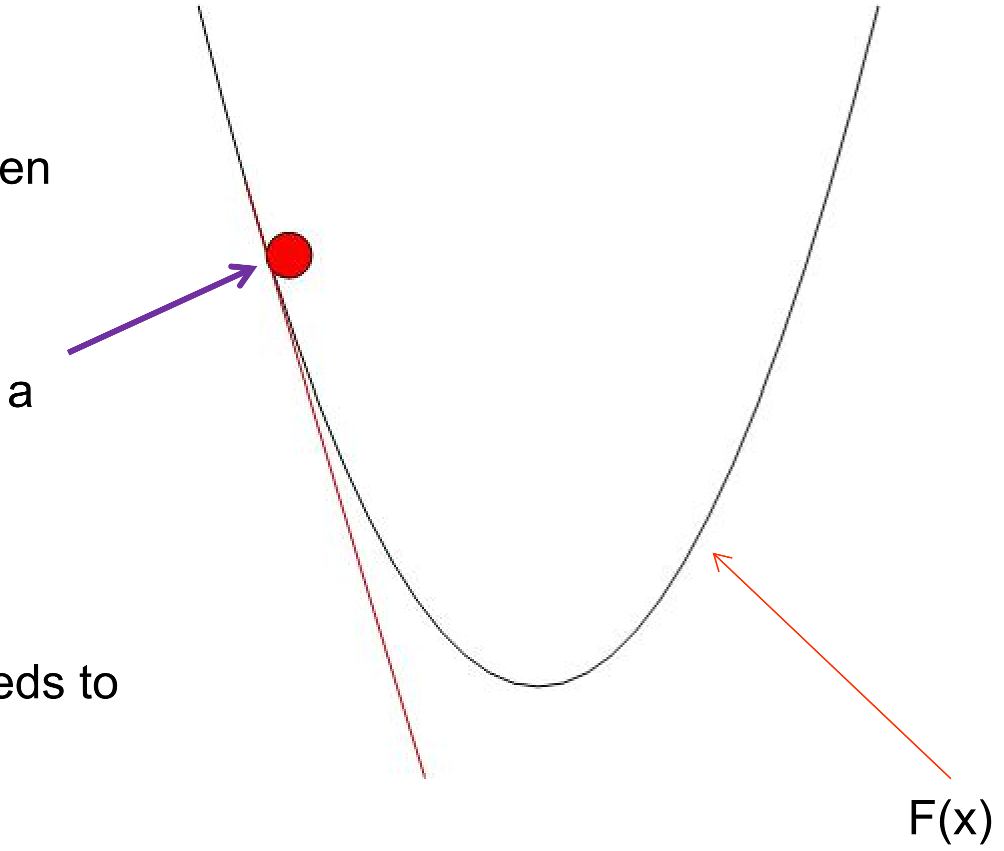
one takes steps proportional to the *negative* of the gradient

$$\underline{\mathbf{b} = \mathbf{a} - \gamma \nabla F(\mathbf{a})}$$

f(a)

For $\gamma \to 0$ a small enough number, then

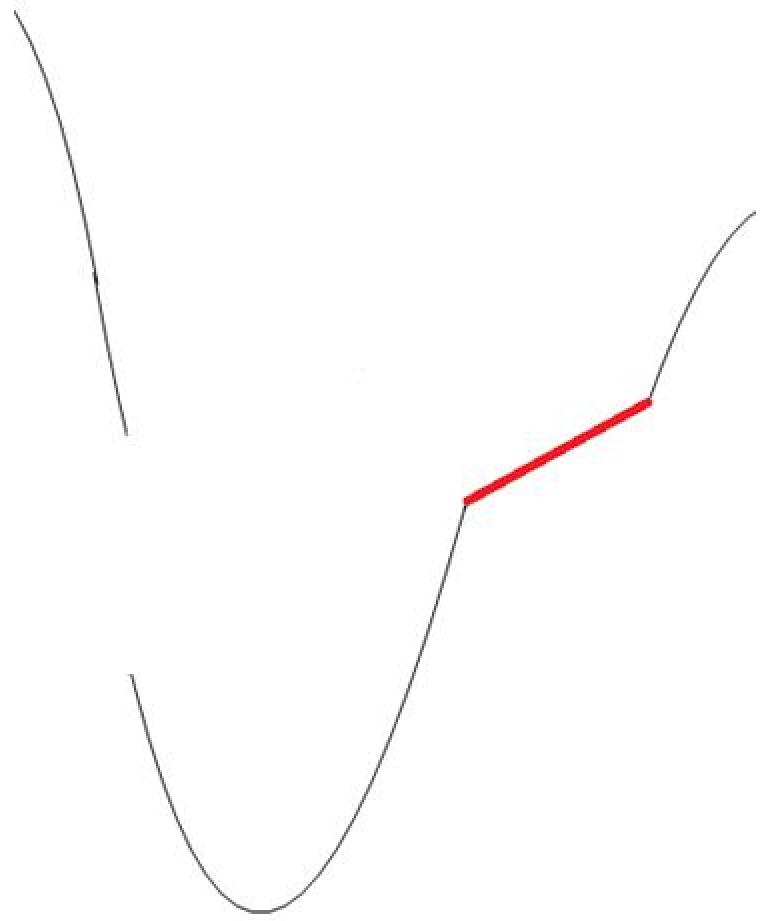
$$F(\mathbf{a}) \geq F(\mathbf{b})$$



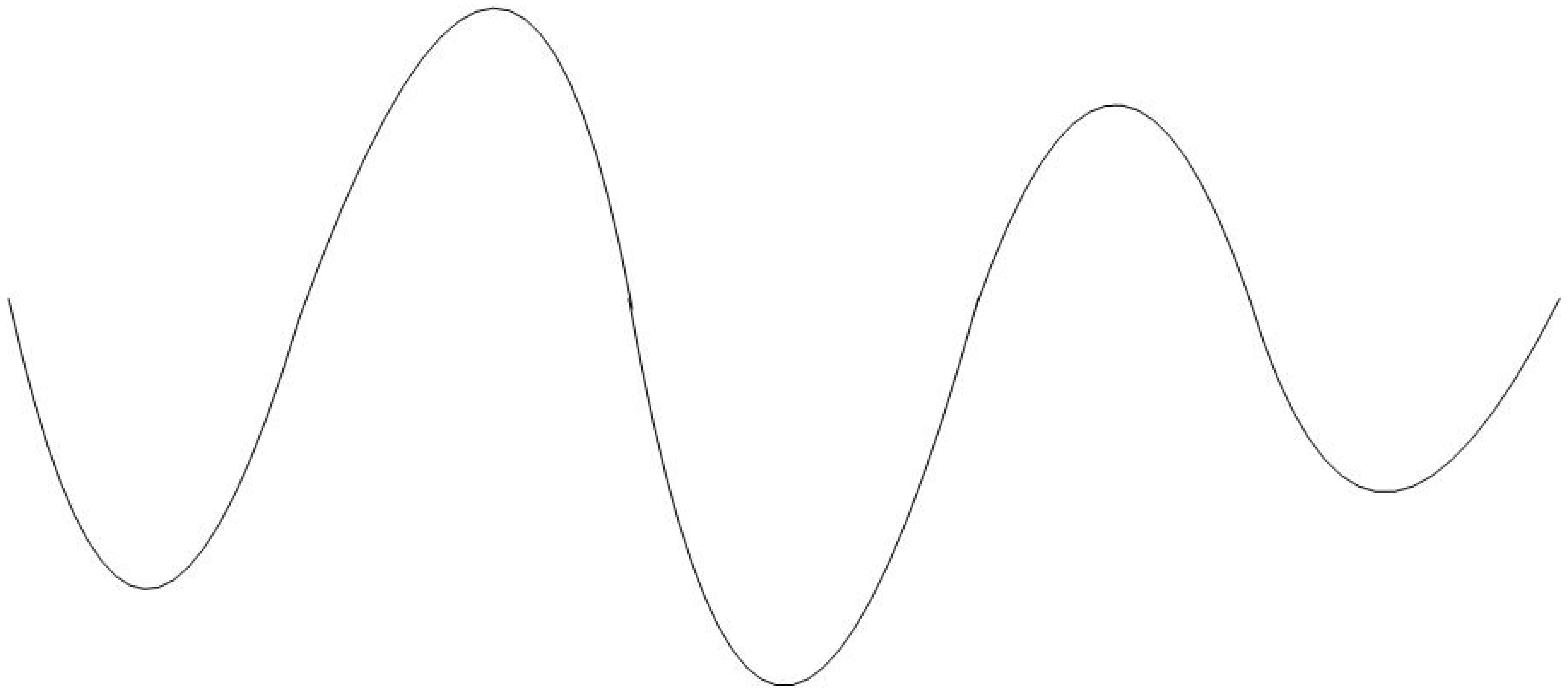
Mapping f: a → b;

condition for obtaining f(.): the F(.) needs to be continuous and differentiable

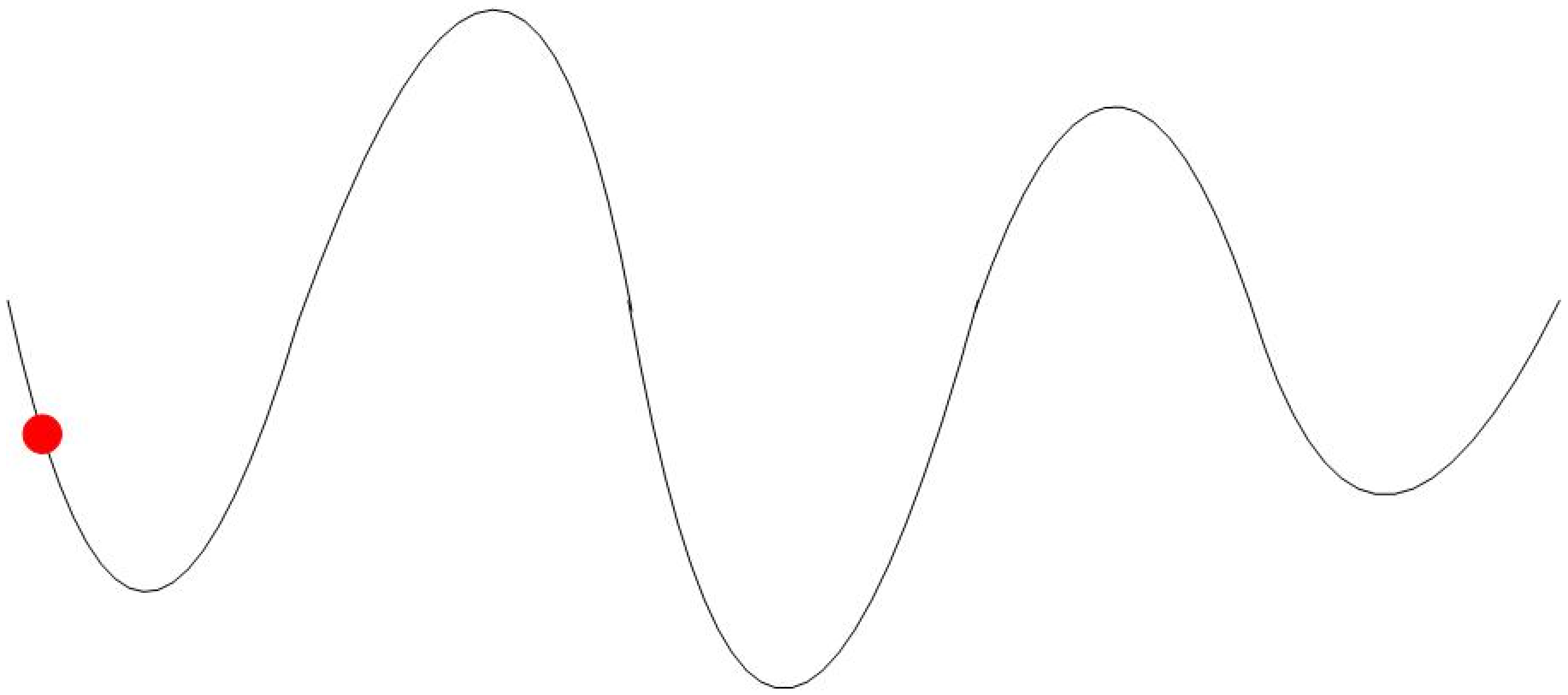
Why Evolutionary Computation - Intutively



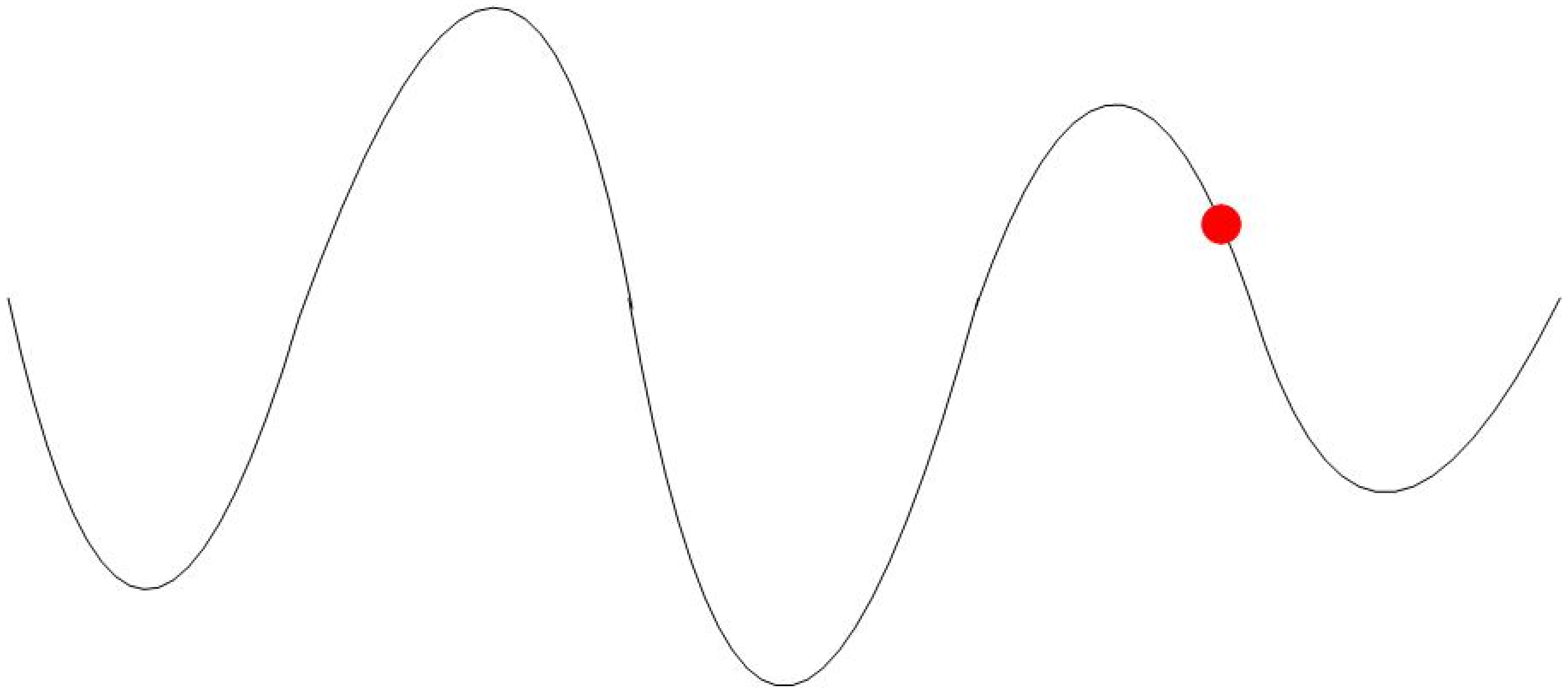
Why Evolutionary Computation - Intutively



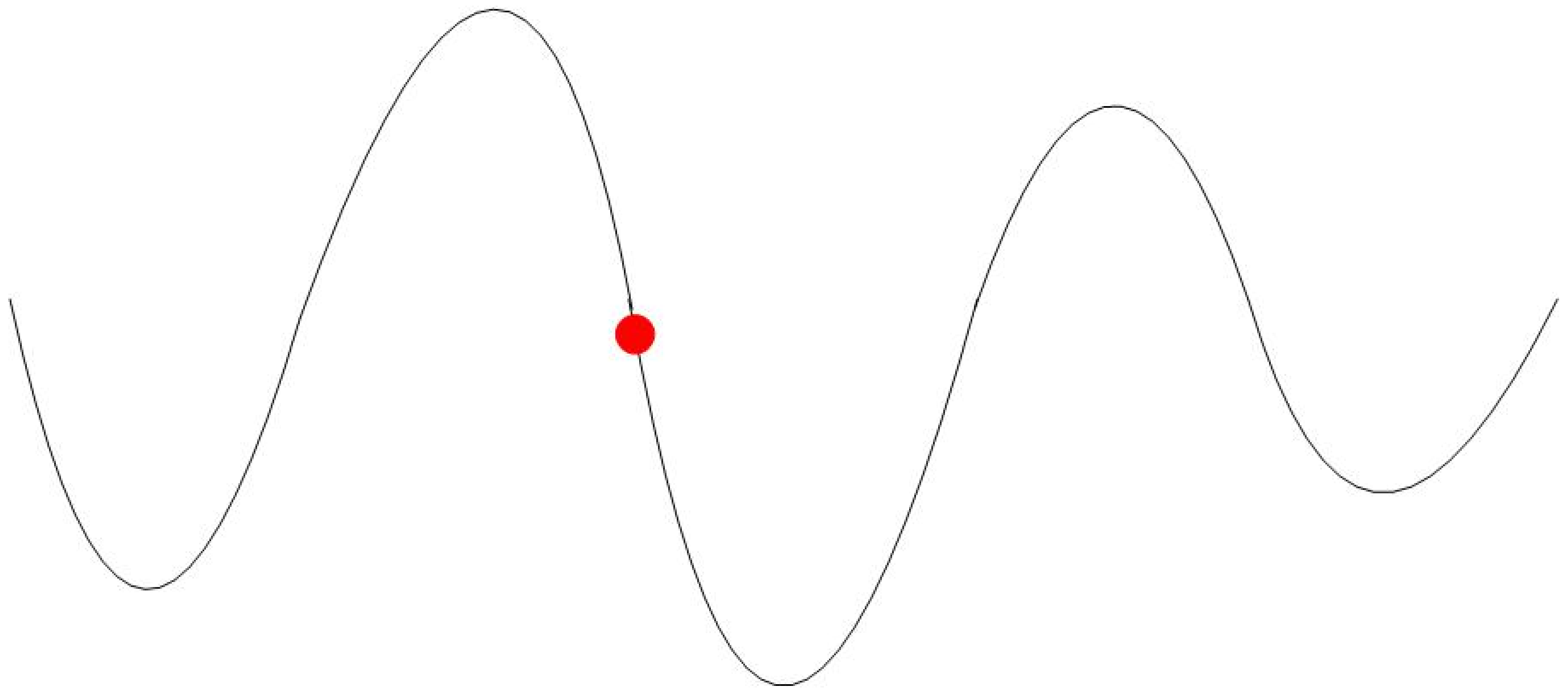
Why Evolutionary Computation - Intutively



Why Evolutionary Computation - Intutively



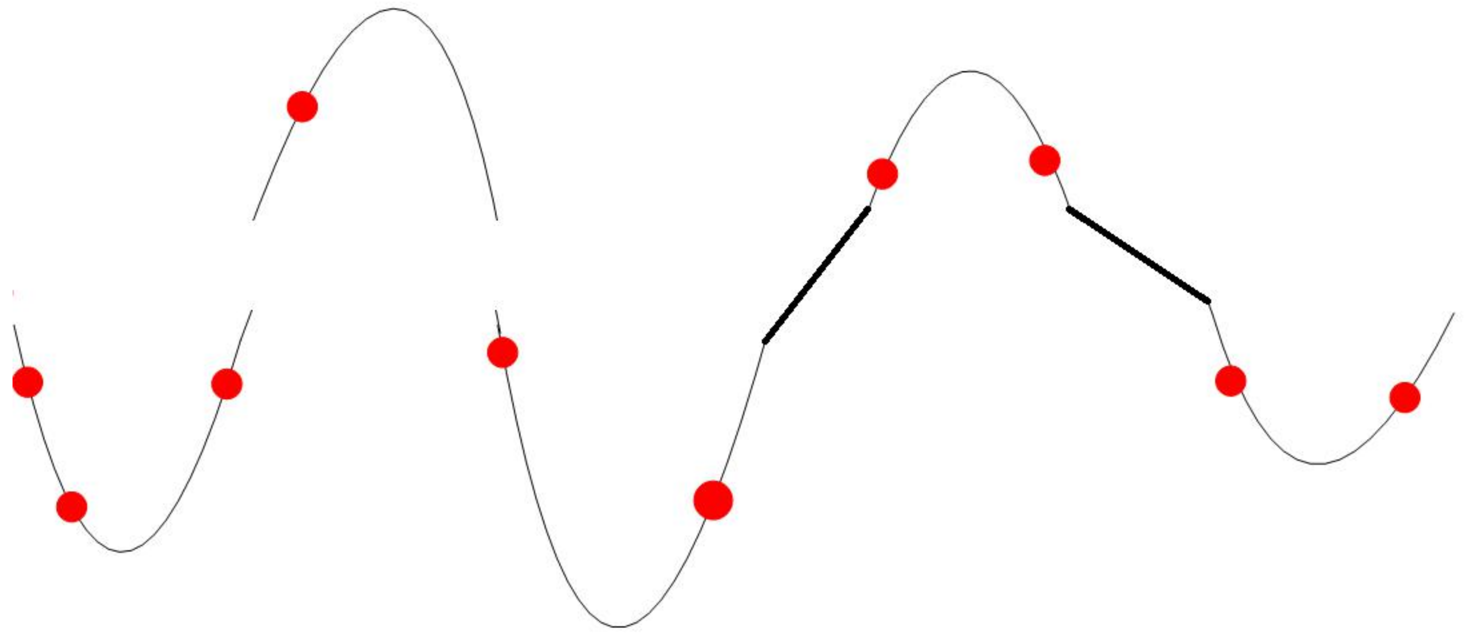
Why Evolutionary Computation - Intutively



Why Evolutionary Computation - Intutively

Not know where should move to?

Avoid Misleading → Add noise → exploration vs. exploitation



Different methods use different mapping functions ← Population-based methods

↑

$f(x_1,x_2,\ldots,x_{10}) \rightarrow x_1,x_2,\ldots,x_{10}$ ⟶ Design the mapping function f(.)

Why Evolutionary Computation - Intutively

Existing Population-Based Algorithms (evolutionary computation)

- Genetic algorithms
- Evolutionary programming
- Evolutionary Strategy
- Genetic Programming
- Swarm Intelligence
  - ✓ Ant Colony Optimization algorithms
  - ✓ Bacterial Foraging algorithms
  - ✓ **Brain Storm Optimization algorithms**
  - ✓ Fish School Optimization algorithms
  - ✓ **Particle Swarm Optimization algorithms**
  - ✓ etc.

Evolutionary Computation (演化计算)

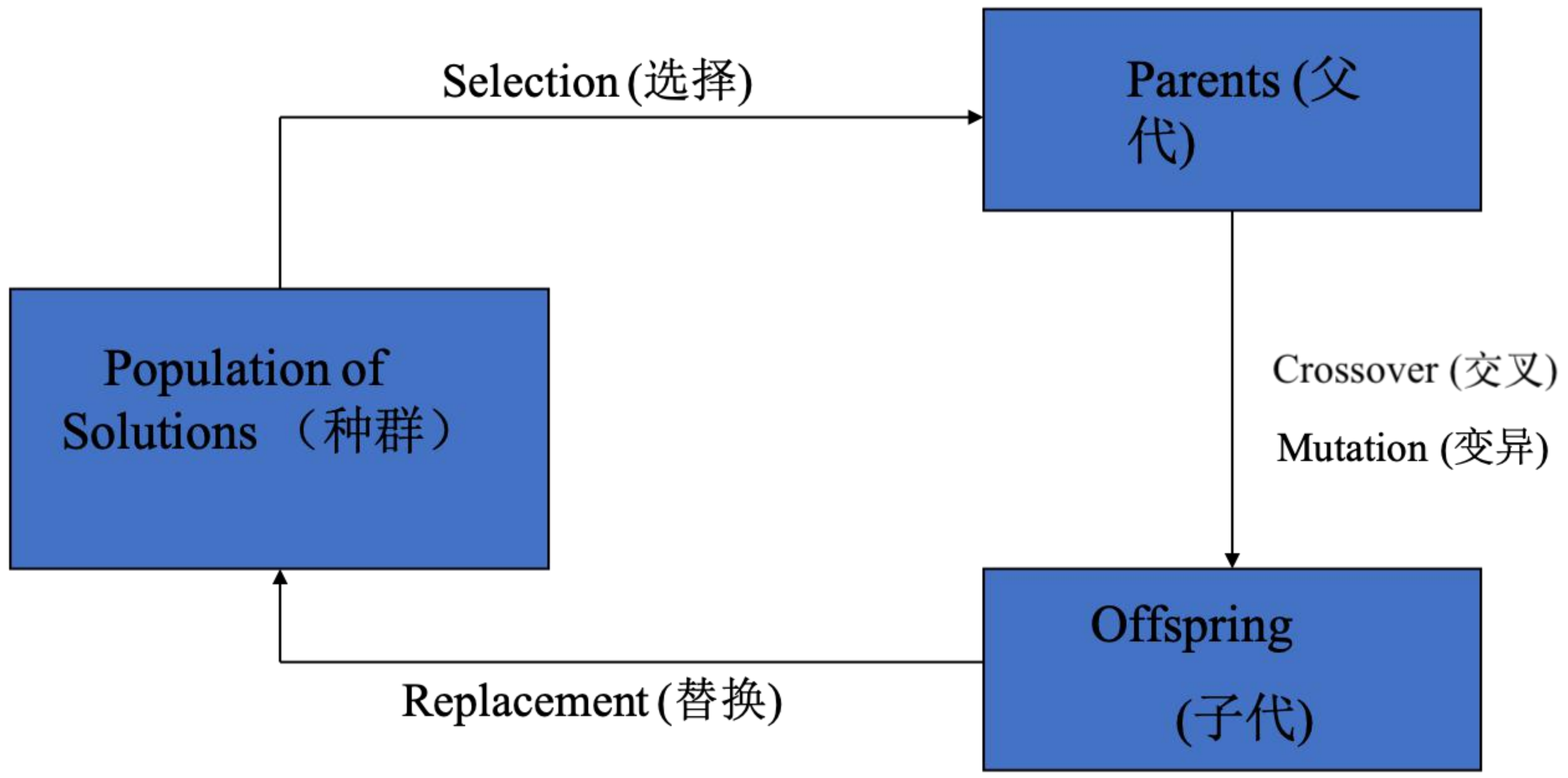


Figure 2: Survival of the fittest (适者生存).

What Is Evolutionary Computation

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

EA 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.

While drawing analogy to biology may be sexy, it is important to understand the underlying links between "new" AI and "old" AI

A Simple Evolutionary Algorithm (EA)

```
1. Generate the initial population $P(0)$ at random;
2. Set $i \leftarrow 0$; // Generation counter
3. REPEAT
    a. Evaluate the fitness of each individual in $P(i)$;
    b. Select parents from $P(i)$ based on their fitness in $P(i)$;
    c. Generate offspring from the parents using crossover and mutation to form
       $P(i + 1)$;
    d. $i \leftarrow i + 1$;
4. UNTIL halting criteria are satisfied
```

Evolutionary Computation
- Conceptually An Important Means to AI

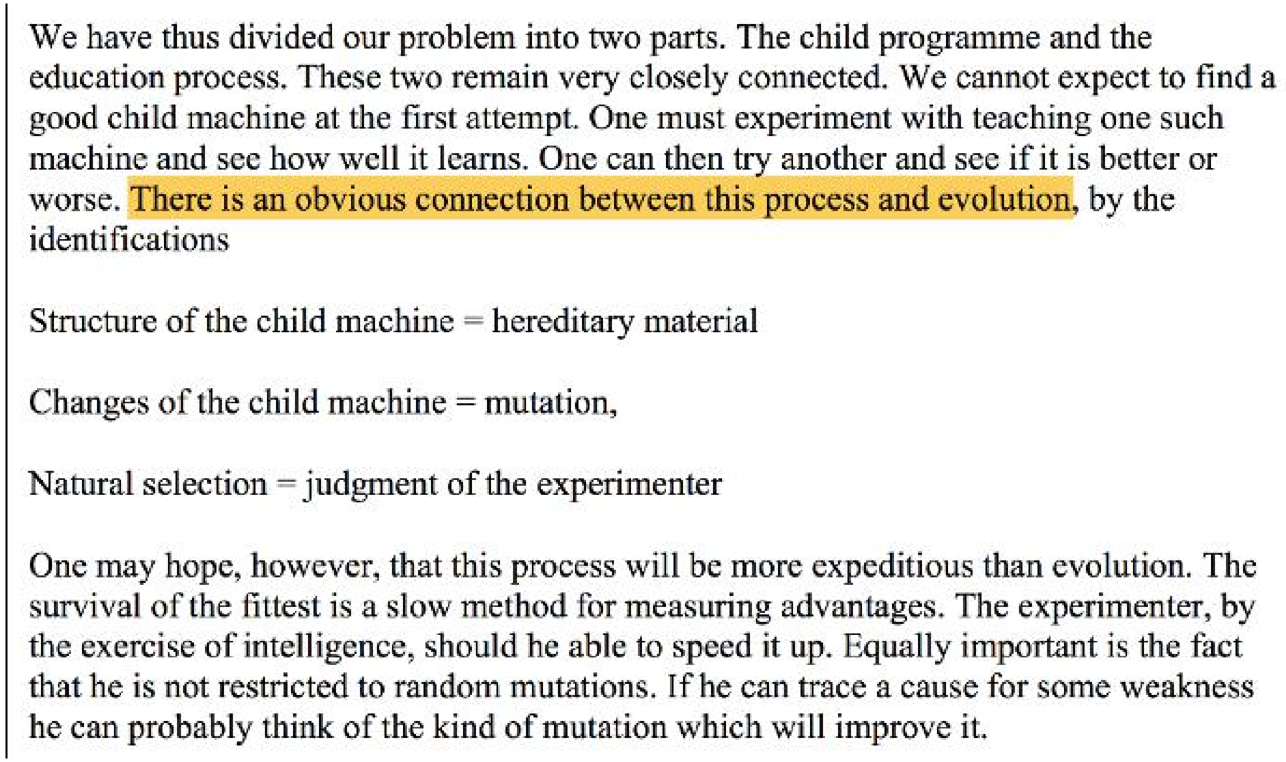
Turing's conceive: realising intelligence with evolution

Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain. Presumably the child brain is something like a notebook as one buys it from the stationer's. Rather little mechanism, and lots of blank sheets. (Mechanism and writing are from our point of view almost synonymous.) Our hope is that there is so little mechanism in the child brain that something like it can be easily programmed. The amount of work in the education we can assume, as a first approximation, to be much the same as for the human child.

[10] A. M. Turing, “Computing machinery and intelligence.” Mind, 49:433-460, 1950.

Evolutionary Computation - Conceptually An Important Means to AI

Turing's conceive: realising intelligence with evolution



We have thus divided our problem into two parts. The child programme and the education process. These two remain very closely connected. We cannot expect to find a good child machine at the first attempt. One must experiment with teaching one such machine and see how well it learns. One can then try another and see if it is better or worse. There is an obvious connection between this process and evolution, by the identifications

Structure of the child machine = hereditary material

Changes of the child machine = mutation,

Natural selection = judgment of the experimenter

One may hope, however, that this process will be more expeditious than evolution. The survival of the fittest is a slow method for measuring advantages. The experimenter, by the exercise of intelligence, should he able to speed it up. Equally important is the fact that he is not restricted to random mutations. If he can trace a cause for some weakness he can probably think of the kind of mutation which will improve it.

[10] A. M. Turing, “Computing machinery and intelligence.” Mind, 49:433-460, 1950.

How Does a Simple EA Work - Example

Let's use the simple EA to maximise the function

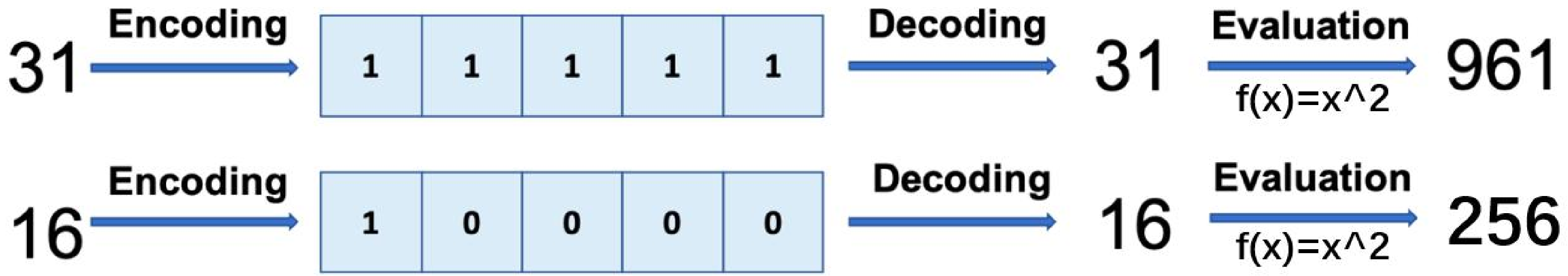
$$f(x) = x^2$$

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

How Does a Simple EA Work - Representation

The first step of EA applications is encoding, i.e., the representation of chromosomes:

- ✓We adopt binary representation for integers.
- ✓Five bits are used to represent integers up to 31.



How Does a Simple EA Work Ⅰ

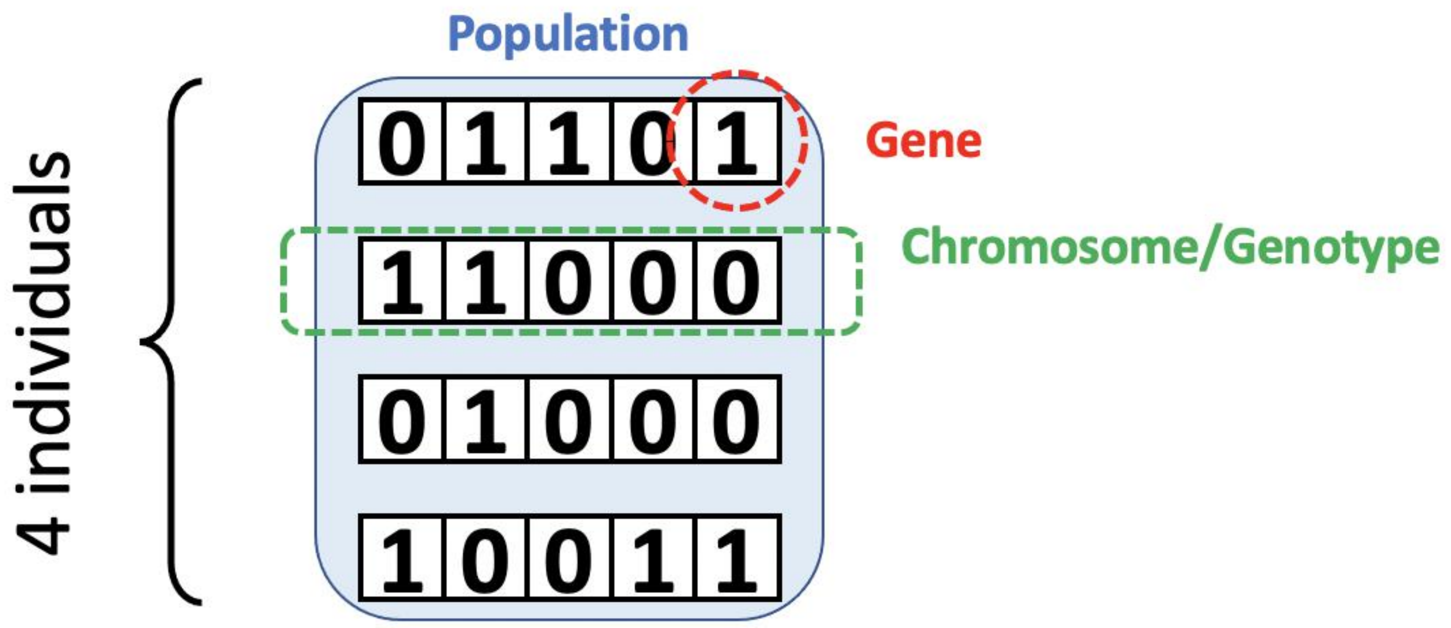
Assume that the population size is 4.

1. Generate initial population at random, e.g.,

   01101, 11000, 01000, 10011.

   These are chromosomes (染色体) or genotypes (基因型).

How Does a Simple EA Work II



How Does a Simple EA Work III

2. Calculate fitness value for each individual.
   a) Decode the individual into an integer (called phenotypes 表现型),

      01101 → 13
      11000 → 24
      01000 → 8
      10011 → 19

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

      01101 → 13 → 169
      11000 → 24 → 576
      01000 → 8 → 64
      10011 → 19 → 361

How Does a Simple EA Work IV

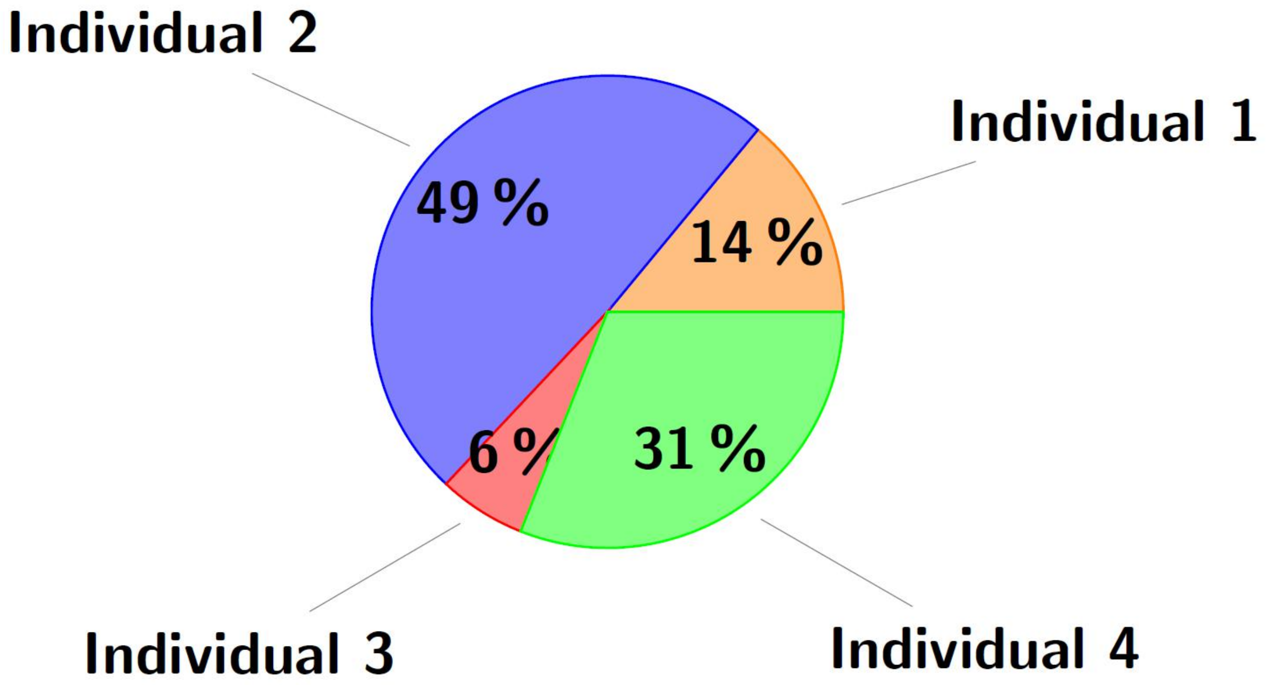
3. Select two individuals for crossover based on their fitness.
   a) If roulette-wheel selection is used, then

$$p_i = \frac{f_i}{\sum_j f_j}.$$

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

$p_1(13) = 169/1170 = 0.14$ $p_2(24) = 576/1170 = 0.49$
$p_3(8) = 64/1170 = 0.06$ $p_4(19) = 361/1170 = 0.31$

How Does a Simple EA Work V



b) Assume we have crossover(01101, 11000) and crossover(10011, 11000). We may obtain offspring 01100 and 11001 from crossover(01101, 11000) by choosing a random crossover point at 4, and obtain 10000 and 11011 from crossover(10011, 11000) by choosing a random crossover point at 2.
Now the intermediate population is 01100, 11001, 10000, 11011.

How Does a Simple EA Work VI

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, 00000, 11011

   The initial population is:

   01101, 11000, 01000, 10011.

5. Goto Step 2 if not stop.

Remarks

- Population-based algorithm.
- Stochastic algorithm.
- No restriction on the fitness or objective function. It can be nondifferentiable, nonsmooth or even discountinuous.
- No need to know the exact form of the objective function. If the objective function is too complex to express explicitly, they can be simulated since EAs only use fitness values.

  → Black-box optimisation.
- There can be many different evolutionary operators and selection mechanisms.

  → We will learn more in future lectures.
- The initial population does not have to be generated at random.

  → We will learn more in future lectures.
- The representation of chromosomes does not have to be binary.

  → We will learn more in future lectures.

Outline of This Lecture

- Why Natural Computation?
- What is Evolutionary Computation?
- **Different Types of Evolutionary Algorithms**
- Major Areas in Evolutionary Computation
- Summary of this Lecture
- Reading List

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)
- Differential Evolution (DE)
- Particle Swarm Optimization (PSO)
- Brain Storm Optimization (BSO)
- ...

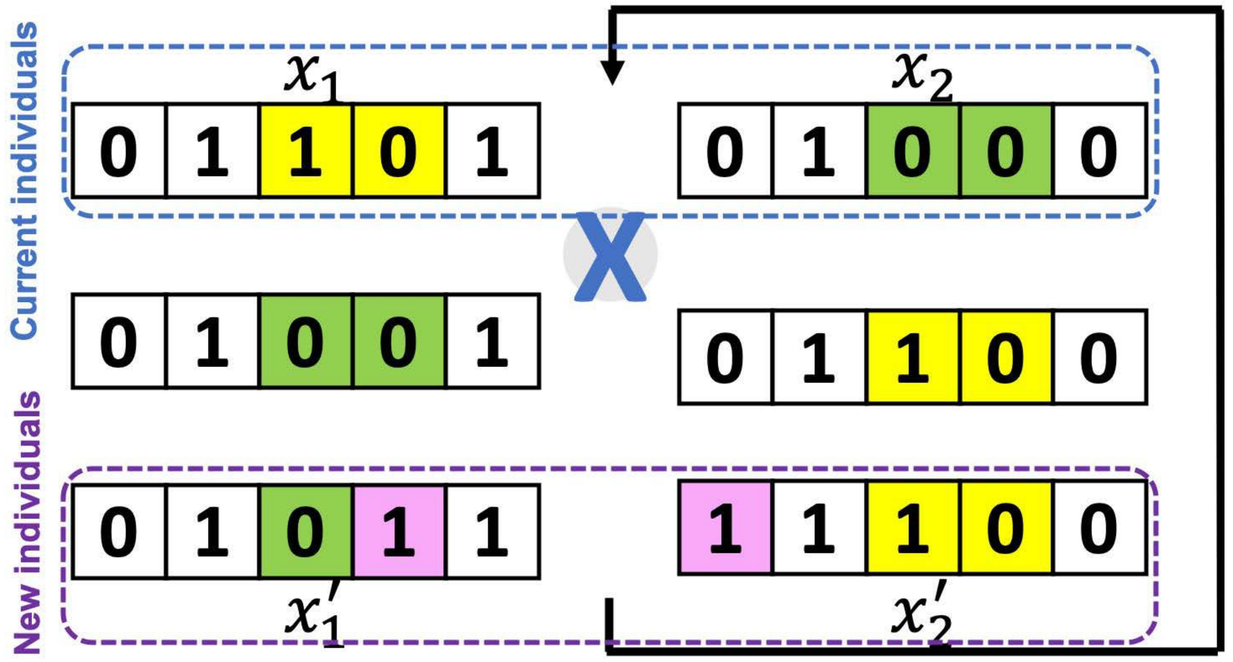
Genetic Algorithms (GAs)

First formulated by Holland and by his students for both numerical optimisation and adaptive system design from mid 1960s to mid 1970s [4]

1. Binary strings have been used extensively as individuals (chromosomes).
2. Simulate Darwinian evolution.
3. Search operators are only applied to the genotypic representation (chromosome) of individuals.
4. Emphasise the role of recombination (crossover). Mutation is only used as a background operator.
5. Often use roulette-wheel selection.

Genetic Algorithms (GAs) - Illustration

Crossover and mutation



Evolutionary Programming (EP)

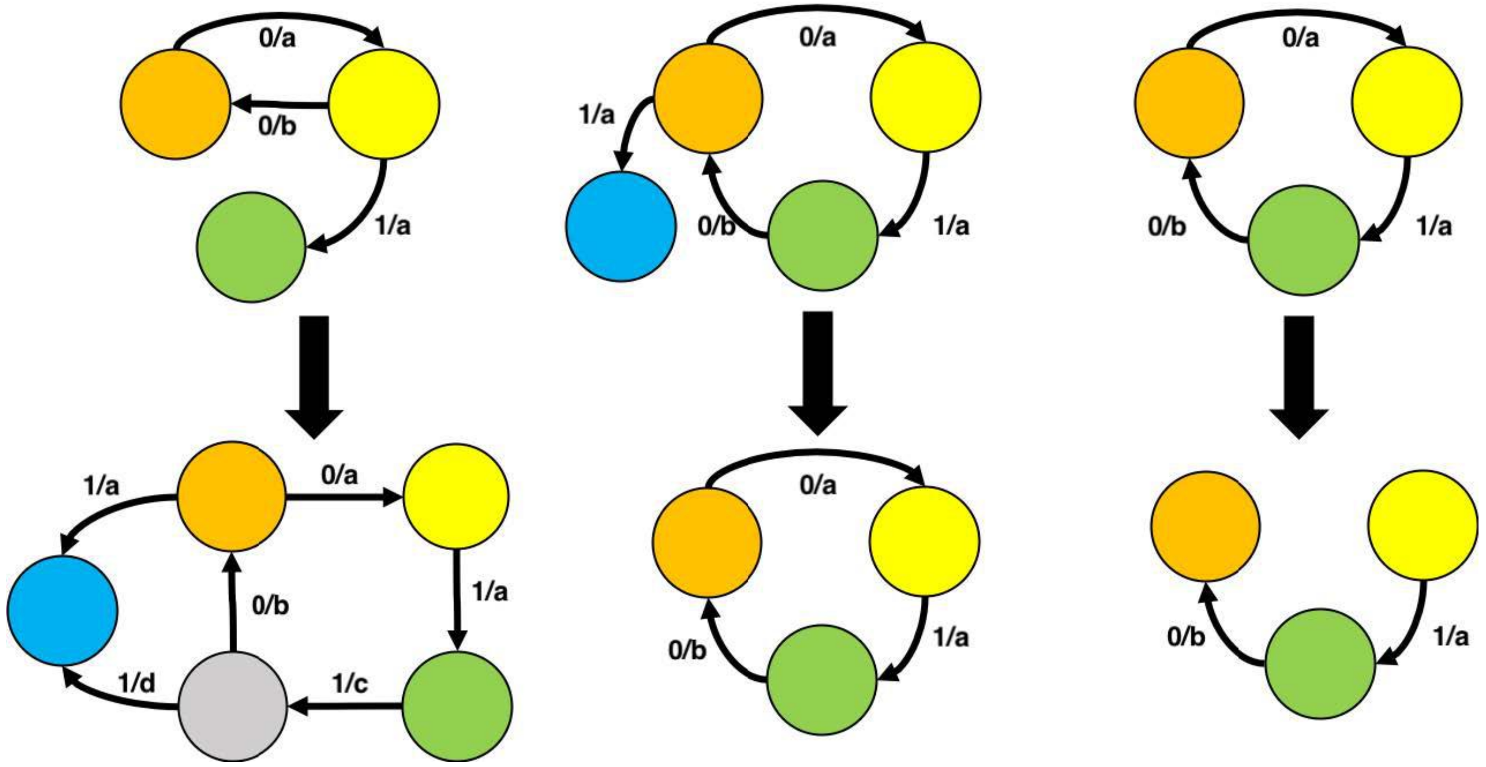
Lamarckism, a theory of evolution based on the principle that physical changes in organisms during their lifetime—such as greater development of an organ or a part through increased use—could be transmitted to their offspring.

First proposed by Fogel for simulating intelligence in 1960s [2,3]

1. Finite-state machines (FSMs) were used to represent individuals, although real-valued vectors have always been used in numerical optimisation.
2. It is closer to Lamarckian evolution.
3. Search operators (mutations only) are applied to the phenotypic representation of individuals.
4. No encoding, not necessarily using binary strings.
5. It does not use any recombination.
6. Self-adaptive mutation is used.

Evolutionary Programming (EP) - Illustration

Mutation



Evolution Strategies (ES)

First proposed by Rechenberg and Schwefel in mid 1960s for numerical optimisation [7,8]

1. Real-valued vectors are used to represent individuals.
2. They are closer to Larmackian evolution.
3. They do have recombination.
4. They use self-adaptive mutations.

Evolution Strategies (ES) - Illustration

$$\begin{tikzcd}
{[x_0^{(1)}, x_1^{(1)}, \ldots, x_{d-1}^{(1)}]} \arrow[dd] & & {[x_0^{(2)}, x_1^{(2)}, \ldots, x_{d-1}^{(2)}]} \arrow[dd] \\
 & \mathbf{X} & \\
{[x'^{(1)}_0, x'^{(1)}_1, \ldots, x'^{(1)}_{d-1}]} & & {[x'^{(2)}_0, x'^{(2)}_1, \ldots, x'^{(2)}_{d-1}]}
\end{tikzcd}$$

Notations:

- Each vector is a candidate solution.
- *d* denotes the dimension of the solution.
- $\forall i \in \{0, 1, . . . , d - 1\}$ and $\forall j \in \{1, 2\}$,
  - $x^{(j)}_i$ is the coordinate $(i + 1)$ of the parent *j*,
  - $x'^{(j)}_i$ is the coordinate $(i + 1)$ of the offspring *j*.

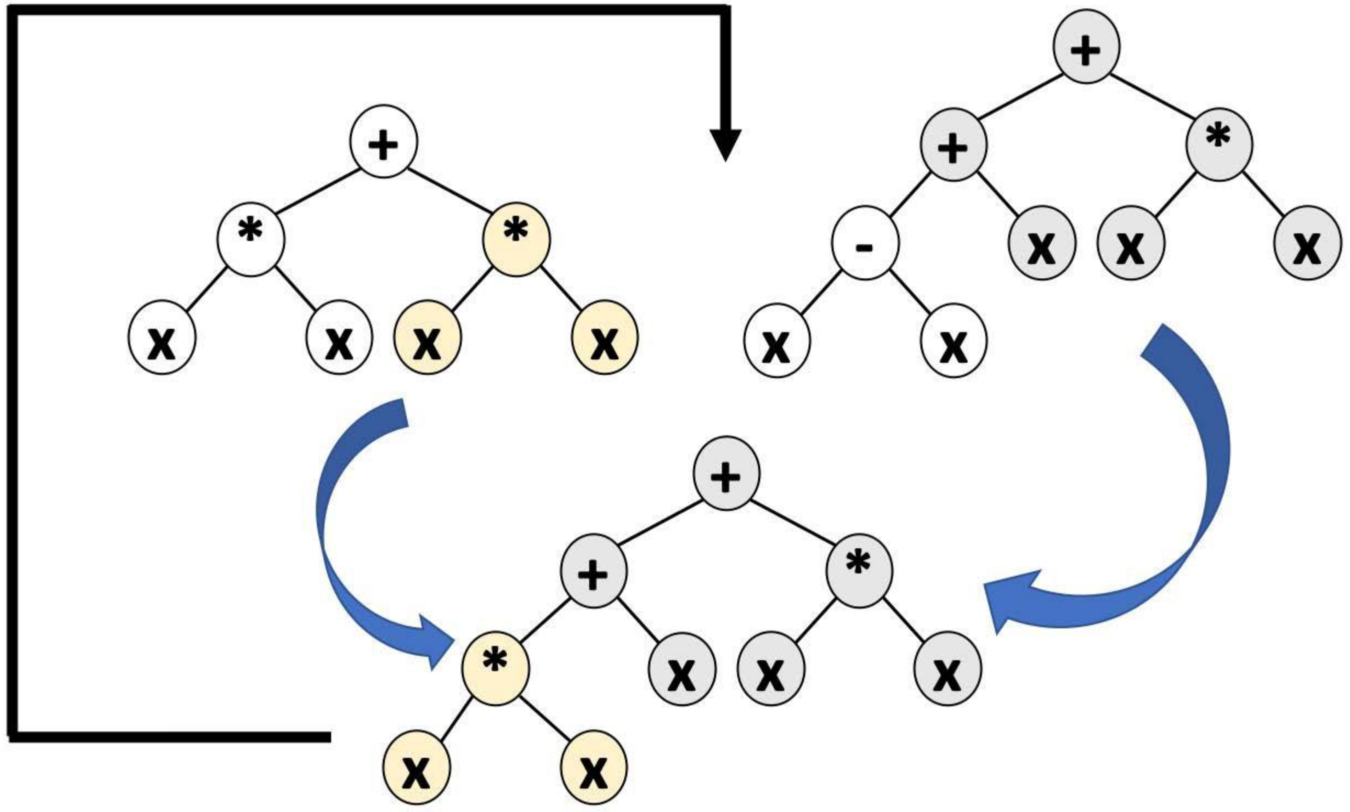
Genetic Programming (GP)

First used by de Garis to indicate the evolution of artificial neural networks, but used by Koza to indicate the application of GAs to the evolution of computer programs [6]

1. Trees (especially Lisp expression trees) are often used to represent individuals.
2. Both crossover and mutation are used.

Genetic Programming (GP) - Illustration

Crossover



Differential Evolution (DE)

- Continuous search space
- Gradient-free
- Ill-conditioning

DE variants:

- DE/best/1: $p'_i = p_{best} + F(p_b - p_c)$
- DE/best/2: $p'_i = p_{best} + F(p_b - p_c) + F(p_d - p_e)$
- DE/rand/1: $p'_i = p_a + F(p_b - p_c)$
- DE/rand/2: $p'_i = p_a + F(p_b - p_c) + F(p_d - p_e)$

where p_{best} is the best point in the current population, p_a, p_b, p_c, p_d and p_e are distinct points randomly chosen in the current population.

Differential Evolution (DE) - Generalised Framework

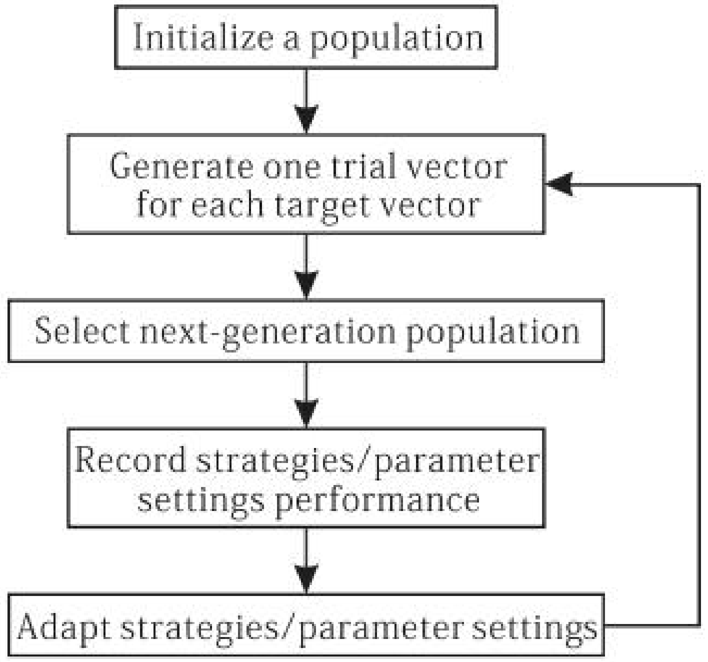

Initialize a population
Generate one trial vector for each target vector
Select next-generation population
Record strategies/parameter settings performance
Adapt strategies/parameter settings

Figure 3: Screenshot of Figure 1.(a) in “X. Lu, K. Tang, B. Sendhoff and X. Yao, A New Self-adaptation Scheme for Differential Evolution, Neurocomputing, 146:2-16, 2014”.

Particle Swarm Optimisation (PSO)

- Proposed by Kennedy and Eberhart in 1995 [5]
- No crossover & mutation defined through a vector addition
- Consider an individual as a point in space with
    - ✓a position x and
    - ✓a velocity v: used to determine a new position (and a new velocity)
- Differences compared to DE: every candidate solution $x \in R^d$ carries its own perturbation vector $v \in R^d$

Preferred Term: Evolutionary Algorithms

- EAs face the same fundamental issues as those classical AI faces, i.e., representation, and search.
- Although GAs, EP, ES, GP, DE, PSO, etc., 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).

Variation Operators and Selection Schemes

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

Key points about operators and selections?

Outline of This Lecture

- Why Natural Computation?
- What is Evolutionary Computation?
- Different Types of Evolutionary Algorithms
- **Major Areas in Evolutionary Computation**
- Summary of this Lecture
- Reading List

Major Areas in Evolutionary Computation

- Optimisation
- Learning
- Design
- Theory

Evolutionary Optimisation

- Numerical (global) optimisation.
- Combinatorial optimisation (of NP-hard problems).
- Mixed optimisation.
- Constrained optimisation.
- Multi-objective optimisation.
- Optimisation in a dynamic and/or uncertain environment.

What Can Evolutionary Computation Bring to Us?
-Intelligent Optimisation as An Example

- Optimisation lies in the foundation of AI.
- Optimisation is to choose the best solution from a set of candidates as quickly as possible.

$$\underbrace{x^*}_{\textbf{The optimal solution}} = \operatorname*{argmin}_{x \in \mathcal{S}} \underbrace{f}_{\textbf{The problem}} (x)$$

where

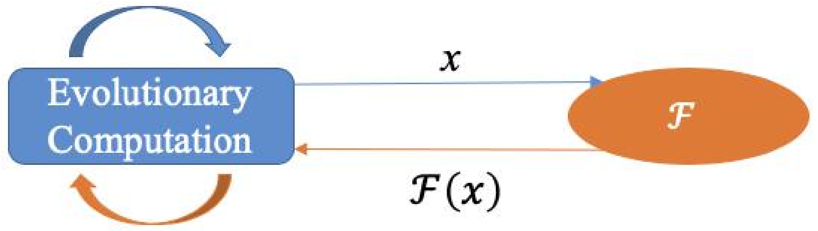
✓S is solution space and d is dimensionality,

✓f : $S \rightarrow R$ and $S \subseteq X^d$.

- Evolutionary Computation is featured to realise intelligent optimisation.

The Features of Evolutionary Computation

- Grey-box Model. Domain knowledge is not a must, but a boost.
  - ✓No need for the mathematical definitions of problems.
  - ✓Open to various types of search space.
    - ➢Continuous optimisation.
    - ➢Combinatorial optimisation.
    - ➢Hybrid optimisation.
- Population-based Randomised Iterative Search.
  - ✓Resistant to data noise.
  - ✓Resistant to local optima.
  - ✓Providing multiple solutions.
    - ➢Multi-objective optimisation.
    - ➢Finding multiple solutions.



Evolutionary Learning

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

1. Learning classifier systems (rule-based systems).
2. Evolutionary artificial neural networks.
3. Evolutionary fuzzy logic systems.
4. Co-evolutionary learning.
5. Automatic modularisation of machine learning systems by speciation and niching.

Evolutionary Design

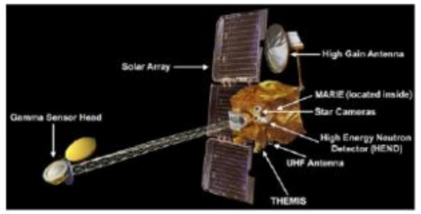
EC techniques are particularly good at exploring unconventional designs which are very difficult to obtain by hand.

1. Evolutionary design of artificial neural networks.
2. Evolutionary design of electronic circuits.
3. Evolvable hardware.
4. Interactive creative design using evolutionary approaches.

Evolutionary Computation Theory

It explains how, when and why EAs work.

Some Successful Examples of Evolutionary Computation



Aerospace

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

Logistic

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

Architecture

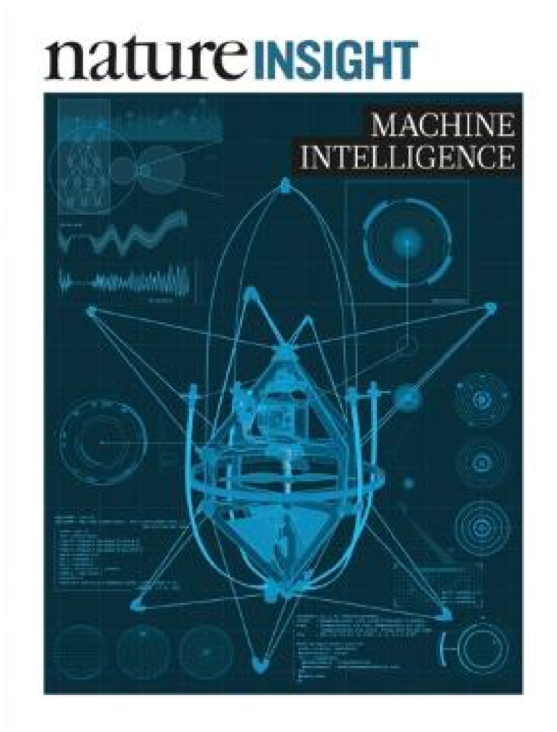
Ludger Hovestadt. Beyond the Grid - Architecture and Information Technology. Applications of a Digital Architectonic. Birkhäuser Basel / Boston 2009.

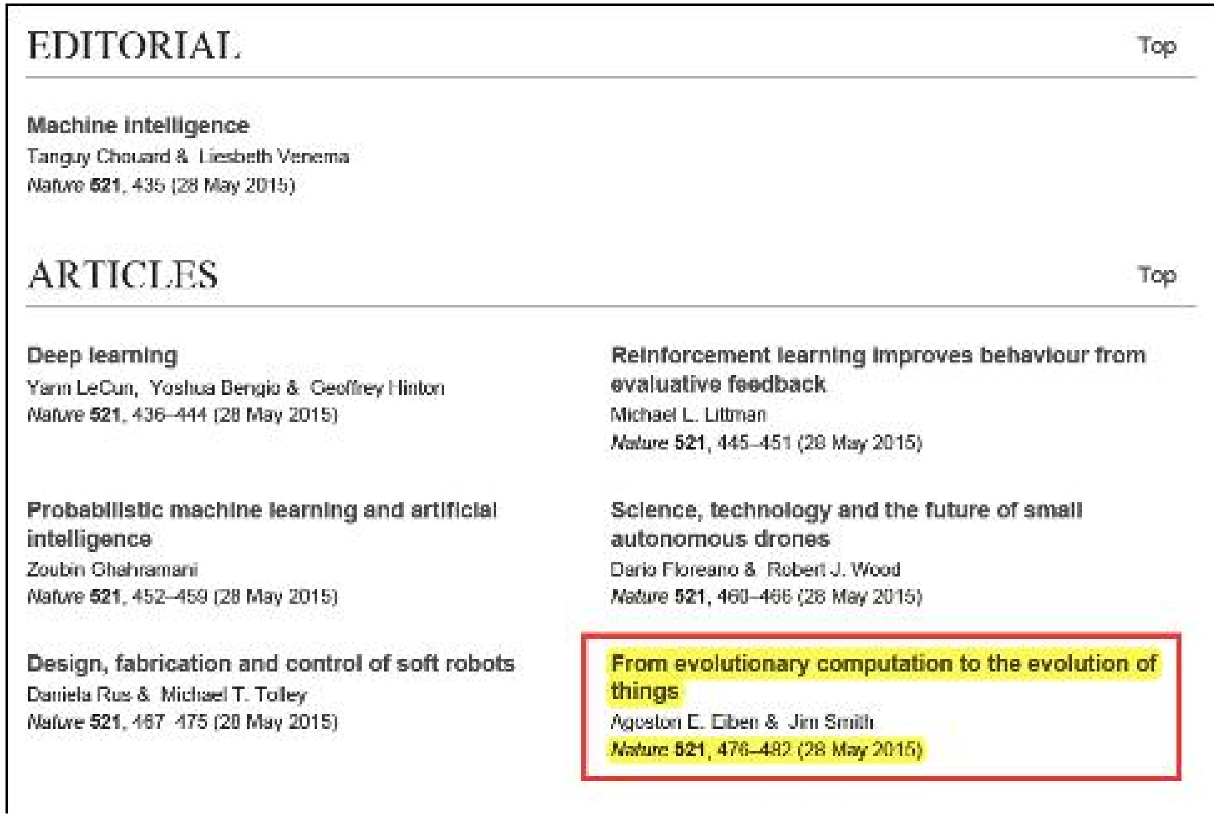
Robotics

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

Become an Important Branch of AI

- EC has been listed as one out of six representative techniques of machine intelligence by Nature in 2015.




EDITORIAL Top

Machine intelligence
Tanguy Chouard & Liesbeth Venema
Nature **521**, 435 (28 May 2015)

ARTICLES Top

Deep learning
Yann LeCun, Yoshua Bengio & Geoffrey Hinton
Nature **521**, 436–444 (28 May 2015)

Reinforcement learning improves behaviour from evaluative feedback
Michael L. Littman
Nature **521**, 445–451 (28 May 2015)

Probabilistic machine learning and artificial intelligence
Zoubin Ghahramani
Nature **521**, 452–459 (28 May 2015)

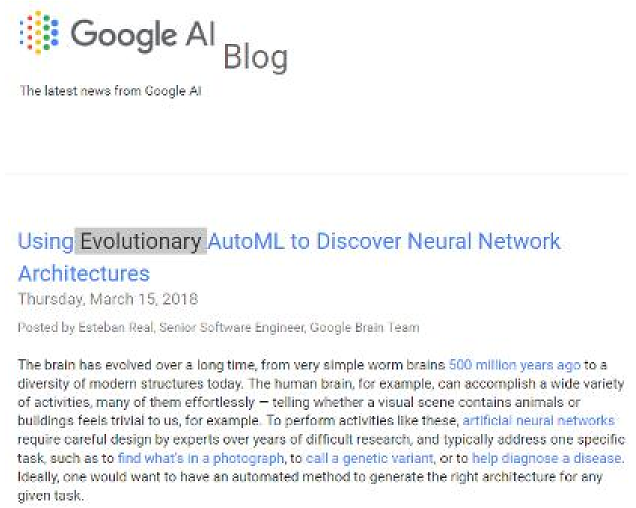
Science, technology and the future of small autonomous drones
Dario Floreano & Robert J. Wood
Nature **521**, 460–466 (28 May 2015)

Design, fabrication and control of soft robots
Daniela Rus & Michael T. Tolley
Nature **521**, 467–475 (28 May 2015)

From evolutionary computation to the evolution of things
Agoston E. Eiben & Jim Smith
Nature **521**, 476–482 (28 May 2015)

Attentions from Government And Leading IT Companies

- EC has been listed in the Commerce Control List of US Department of Commerce at Nov. 2018, placed the 2nd technique in AI field. (From “美国商务部受出口管制的代表性新兴技术清单征求意见稿” *)
- Recently, leading IT companies have put a lot on EC.


Google AI Blog

The latest news from Google AI

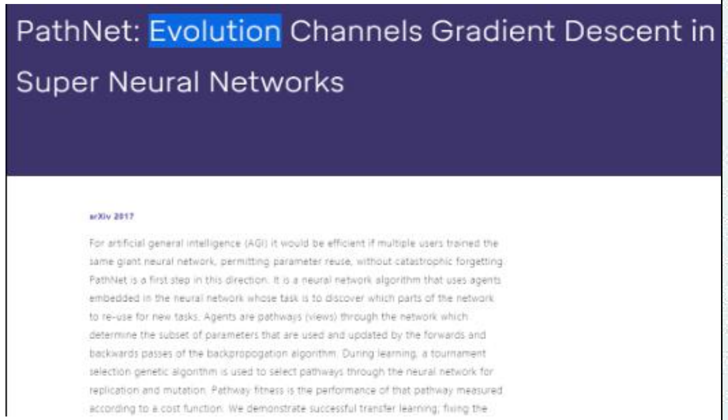
Using Evolutionary AutoML to Discover Neural Network Architectures

Thursday, March 15, 2018

Posted by Esteban Real, Senior Software Engineer, Google Brain Team

The brain has evolved over a long time, from very simple worm brains 500 million years ago to a diversity of modern structures today. The human brain, for example, can accomplish a wide variety of activities, many of them effortlessly – telling whether a visual scene contains animals or buildings feels trivial to us, for example. To perform activities like these, artificial neural networks require careful design by experts over years of difficult research, and typically address one specific task, such as to find what's in a photograph, to call a genetic variant, or to help diagnose a disease. Ideally, one would want to have an automated method to generate the right architecture for any given task.

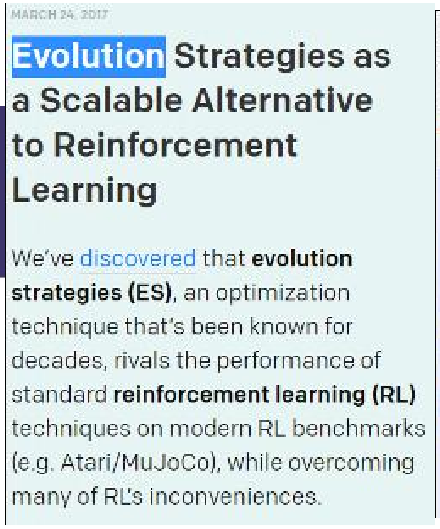
(1) Google AI


PathNet: Evolution Channels Gradient Descent in Super Neural Networks

arXiv 2017

For artificial general intelligence (AGI) it would be efficient if multiple users trained the same giant neural network, permitting parameter reuse, without catastrophic forgetting. PathNet is a first step in this direction. It is a neural network algorithm that uses agents embedded in the neural network whose task is to discover which parts of the network to re-use for new tasks. Agents are pathways (views) through the network which determine the subset of parameters that are used and updated by the forwards and backwards passes of the backpropogation algorithm. During learning, a tournament selection genetic algorithm is used to select pathways through the neural network for replication and mutation. Pathway fitness is the performance of that pathway measured according to a cost function. We demonstrate successful transfer learning; fixing the

(2) DeepMind


MARCH 24, 2017

Evolution Strategies as a Scalable Alternative to Reinforcement Learning

We've discovered that **evolution strategies (ES)**, an optimization technique that's been known for decades, rivals the performance of standard **reinforcement learning (RL)** techniques on modern RL benchmarks (e.g. Atari/MuJoCo), while overcoming many of RL's inconveniences.

(3) Open AI

Facebook's evolutionary search for crashing software bugs

Ars gets the first look at Facebook's fancy new dynamic analysis tool.

SEBASTIAN ANTHONY - 8/22/2017, 10:26 PM

(4) Facebook

(*) https://www.govinfo.gov/content/pkg/FR-2018-11-19/pdf/2018-25221.pdf
(1) https://ai.googleblog.com/2018/03/using-evolutionary-automl-to-discover.html
(2) https://deepmind.com/research/publications/pathnet-evolution-channels-gradient-descent-super-neural-networks/
(3) https://blog.openai.com/evolution-strategies/#content
(4) https://arstechnica.com/information-technology/2017/08/facebook-dynamic-analysis-software-sapienz/

Outline of This Lecture

Summary

1. EC is a field of study that includes EAs and other areas. EAs include many types of algorithms.
2. EAs can be regarded as population-based generate-and-test algorithms.
3. EC techniques can be used in optimisation, learning and design.
4. EC techniques are flexible and robust.
5. EC techniques are definitely useful tools in your toolbox, but there are problems for which other techniques might be more suitable.

Outline of This Lecture

- Why Natural Computation?
- What is Evolutionary Computation?
- Different Types of Evolutionary Algorithms
- Major Areas in Evolutionary Computation
- Summary of this Lecture
- **Reading List**

Essential Reading for This Lecture

1. D. B. Fogel (ed.), Evolutionary Computation: The Fossil Record, IEEE Press, Piscataway, NJ, USA, 1998. (ISBN-10: 0780334817, ISBN-13:978-0780334816)
   - It is essential that you read the introduction and comments, if you do not havetime to read all the classical papers there.
   - It is extremely useful to know the history and origin. Not only do we want to learn brilliant ideas, we also need to learn how they were first conceived and generated.
2. X. Yao, Evolutionary computation: A gentle introduction, In Evolutionary Optimization, R. Sarker, M. Mohammadian and X. Yao (eds.), Chapter 2, pp.27-53, Kluwer Academic Publishers, Boston, 2002. (ISBN 0-7923-7654-4)

Other References I

1. Hugo De Garis. “Genetic programming: Building artificial nervous systems using genetically programmed neural network modules”. In: Machine Learning Proceedings 1990. Elsevier, 1990, pp. 132–139.
2. Lawrence J Fogel. “Autonomous automata”. In: Industrial Research 4 (1962), pp. 14–19.
3. Lawrence J Fogel, Alvin J Owens, and Michael J Walsh. “Artificial intelligence through simulated evolution”. In: (1966).
4. John H Holland. “Genetic algorithms”. In: Scientific American 267.1 (1992), pp. 66–73.
5. J. Kennedy and R. Eberhart. “Particle Swarm Optimization”. In: Proceedings of IEEE International Conference on Neural Networks. 1995, 1942–1948.
6. John R Koza. “Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems”. In: 34 (1990).

Other References II

7. Ingo Rechenberg. “Cybernetic solution path of an experimental problem”. In: Royal Aircraft Establishment Library Translation 1122 (1965).
8. H-P Schwefel. “Kybernetische Evolution als Strategie der experimentellen Forschung in der Stromungstechnik”. In: Diploma thesis, Technical Univ. of Berlin (1965).
9. Rainer Storn. “On the usage of differential evolution for function optimization”. In: Fuzzy Information Processing Society, 1996. NAFIPS. 1996 Biennial Conference of the North American. IEEE. 1996, pp. 519–523.
10. Alan M Turing. “Computing machinery and intelligence”. In: Mind LIX.49 (1950), pp. 433–460.