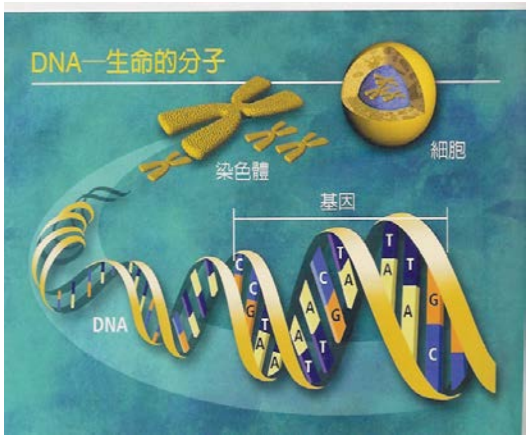
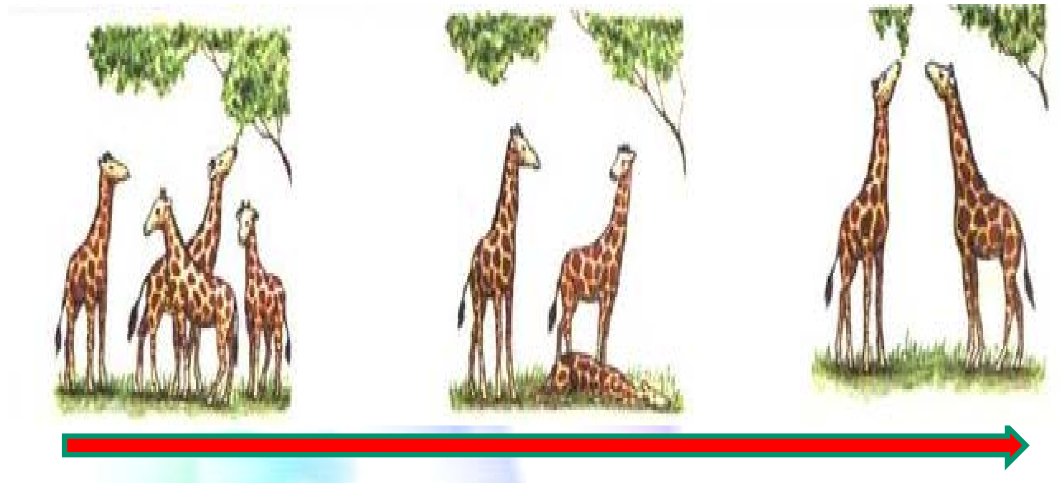


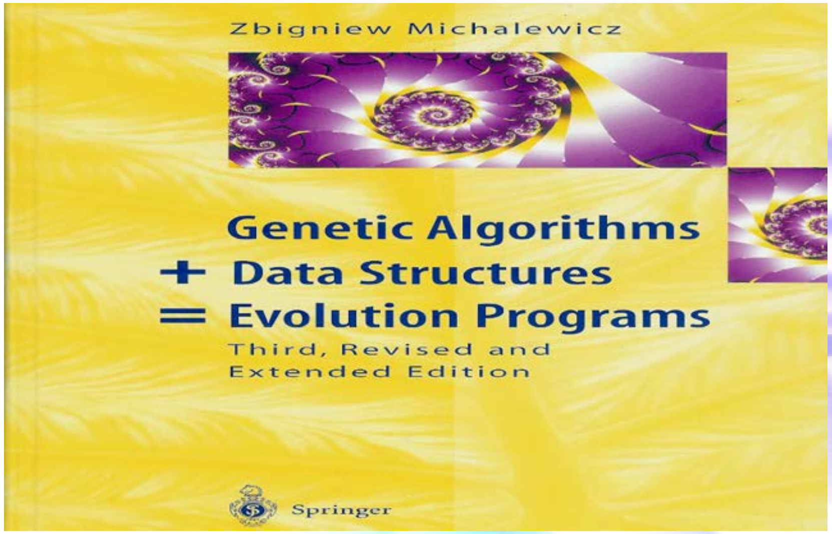
+



Genetic Algorithms
遺傳基因演繹法

Prof. Shu-Mei Guo (郭淑美教授)

Department of Computer Science and Information Engineering
National Cheng-Kung University

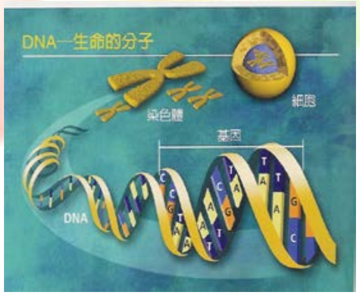


Evolution Programs
進化計算論

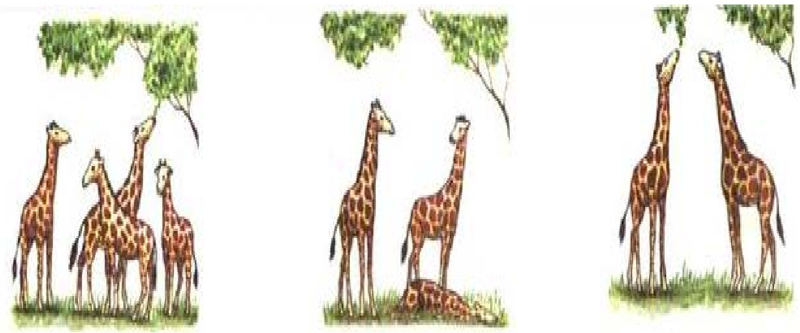
Prof. Shu-Mei Guo (郭淑美教授)
Department of Computer Science
and Information Engineering
National Cheng-Kung University

The Evolution Mechanism

- Evolution Programs(EPs) / Evolutionary Computing(EC) / Evolutionary Computation(EC) /Evolutionary Algorithms(EAs)
  - The principle of *hereditary* and *evolution*
  - A population of possible *solutions*



+



Contents

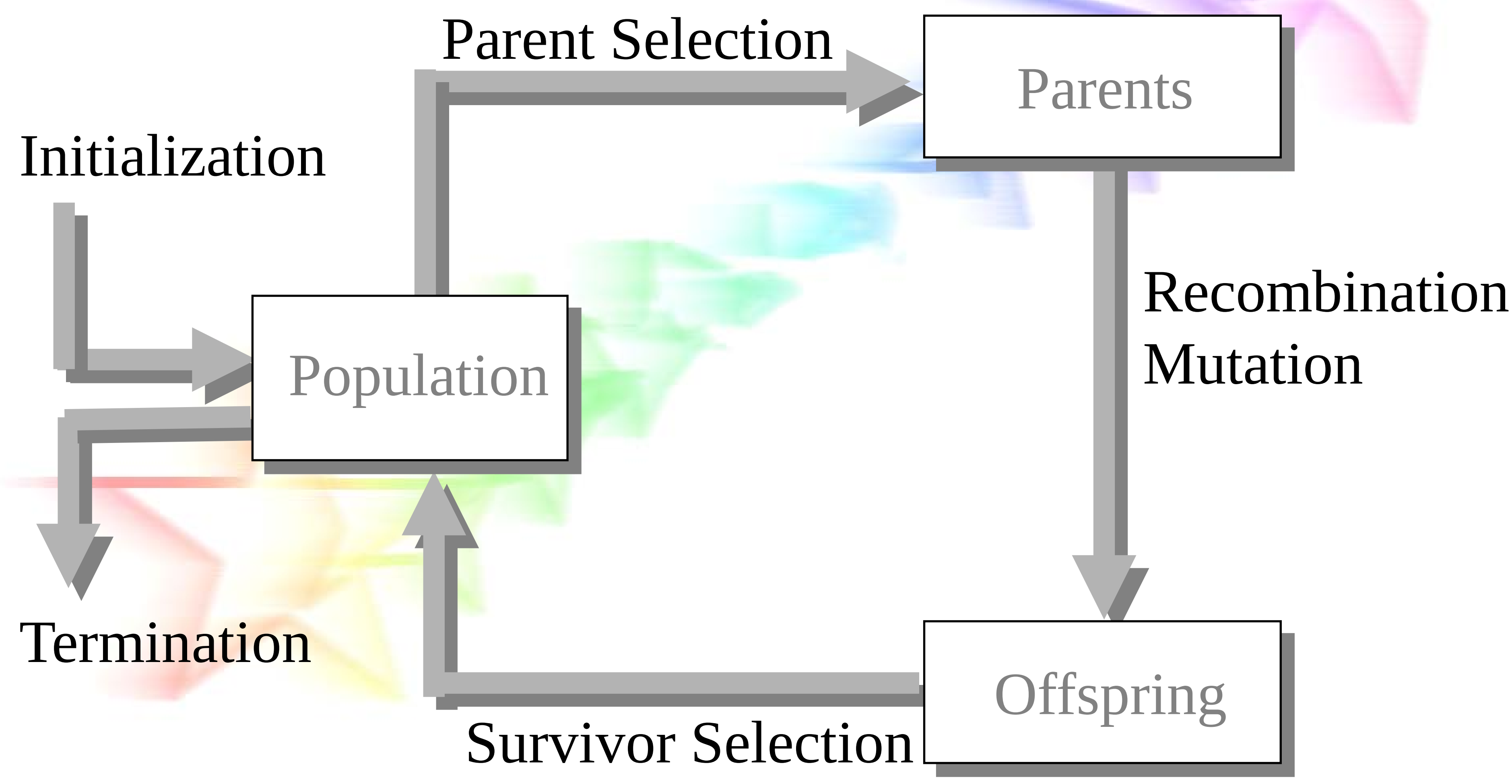
Part I. GENETIC ALGORITHMS

Chapter 1
GAs: What Are They?

Classical Genetic Algorithm

- Darwin's theory of evolution states that the strongest species survive, and thus reproduce themselves, creating more outstanding offspring.
- GA is a parallel, global search and optimal technique based on natural or artificial genetic operations, which include the operators of reproduction, crossover, and mutation.
- GA can effectively solve complex, confliction, mathematically difficult and constrained multiple objective problem.

The Evolutionary Cycle



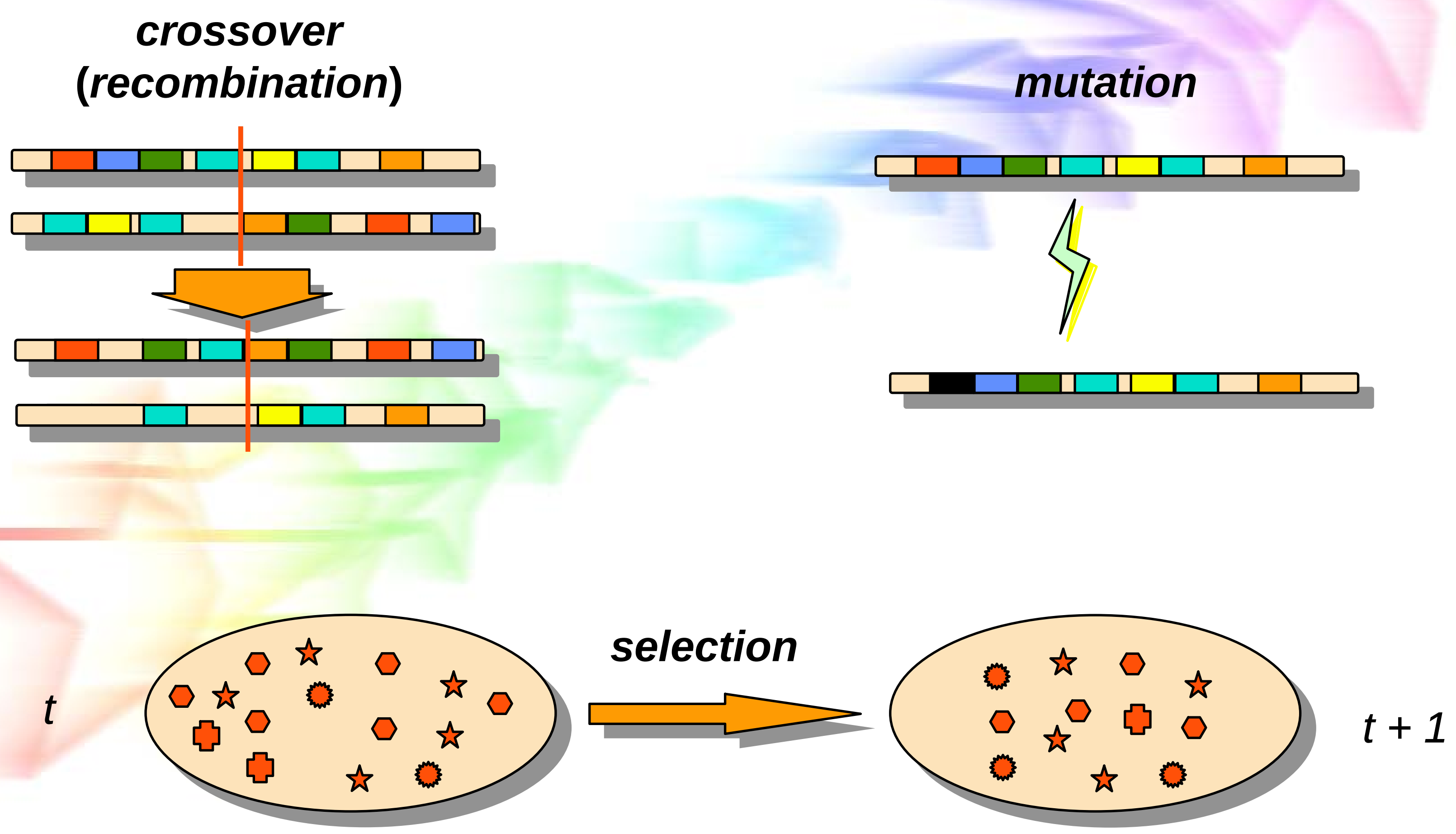
Genetic Algorithm

```
Procedure  Genetic Algorithm
Begin
    t ← 0
    Initialize P(t)
    Evaluate P(t)
    While (not termination-condition) do
    Begin
           t ← t + 1
          Select P(t) from P(t -1)
          Alter P(t)
          Evaluate P(t)
    End
End
```

General Scheme of an GA

- Initialization
  - Population (Individuals)
- Selection
  - Survivor Selection (Evaluation)
- Alter
  - Crossover
  - Mutation
- Termination

Operators

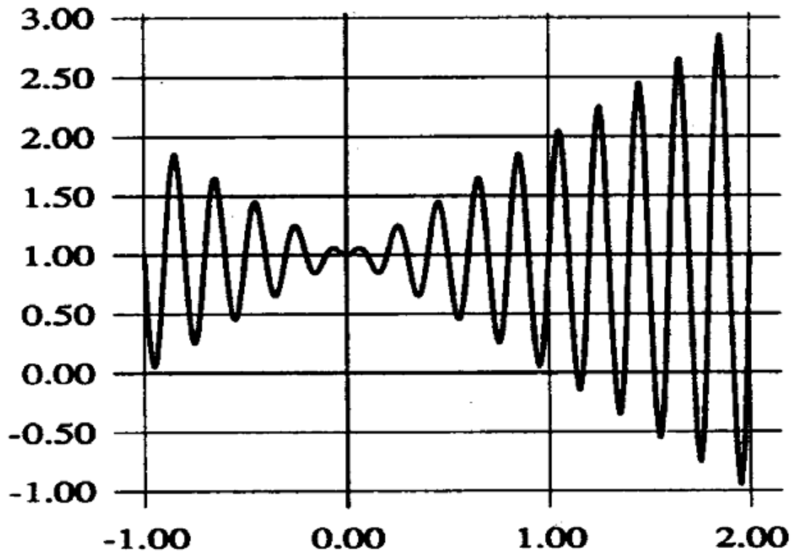


Example

(one variable optimization problem)

$$\max\ f(x) = x \cdot \sin(10\pi \cdot x) + 1$$

$$x \in [-1..2]$$



Results

Generation number	Evaluation function
1	1.441942
6	2.250003
8	2.250283
9	2.250284
10	2.250363
12	2.328077
39	2.344251
40	2.345087
51	2.738930
99	2.849246
137	2.850217
145	2.850227

General Scheme of an GA

1. An **evaluation function** (fitness function)
2. Values for various **parameters** (pop_size, crossover_rate, mutation_rate, …)
3. A genetic **representation** (individual)
4. A way to create an **initial population**
5. Genetic **operators** (crossover, mutation, selection)
6. A way to **terminate** the algorithm
7. GA **algorithm**

General Scheme of an GA

1. An **evaluation function**
   - *Fitness Function :*

2. Values for various **parameters**
   - *population _size =*
   - *Crossover rate =*
   - *Mutation ratio =*

3. A genetic **representation**
4. An i**nitial population**

5. Genetic **operators**
   - *Crossover:*
   - *Mutation:*
   - *Survivor Selection*

6. A way to **terminate** the algorithm

7. GA **algorithm**

The Metaphor

Gene ↔

Genotype
(Chromosome) ↔

Phenotype
(Individual) ↔

Population ↔

Locus(gene) ↔

Fitness ↔

Hillclimbing, Simulated Annealing, and Genetic Algorithms

- Example:

$$\max f(v) = \left| 11 \cdot one(v) - 150 \right|$$

- Comparison:

Hillclimbing

```
Procedure    Iterated  Hillclimber
begin
    t ← 0
    repeat
        local ← FALSE
        select a current string v_c at random
        evaluate v_c
        repeat
            select 30 new strings in the neighborhood of v_c
                by flipping single bits of v_c
            select the string v_n  from the set of new strings
                with the largest value of objective function f
            if  f(v_c) < f(v_n)   then   v_c ← v_n
            else local ← TRUE
        until local
        t ← t + 1
    until  t = max
end
```

Simulated Annealing

```
Procedure    Simulated Annealing
begin
    t ← 0
    Initialize temperature T
    select a current string v_c at random
    evaluate v_c
    repeat
        repeat
            select a new string v_n
                in the neighborhood of v_c
                by flipping single bits of v_c
            if  f(v_c) < f(v_n)
            then  v_c ← v_n
            else
                if  random[0..1) < exp{(f(v_n) - f(v_c))/T}
                then  v_c ← v_n
        until  (termination-condition)
            T ← g(T,t)
            t ← t + 1
    until  (stop-criterion)
end
```

Hillclimbing & Simulated annealing

```
Procedure   Iterated Hillclimber
begin
    t ← 0
    repeat
        local ← FALSE
        select a current string v_c at random
        evaluate v_c
        repeat
            select 30 new strings in the neighborhood
                of v_c by flipping single bits of v_c
            select the string v_n from the set of new
                strings with the largest value of
                objective function f
            if f(v_c) < f(v_n)   then   v_c ← v_n
            else local ← TRUE
        until local
        t ← t + 1
    until t = max
end
```

```
Procedure   Simulated Annealing
begin
    t ← 0
    Initialize temperature T
    select a current string v_c at random
    evaluate v_c
    repeat
        repeat
            select a new string v_n
                in the neighborhood of v_c
                by flipping single bits of v_c
            if f(v_c) < f(v_n)
                then   v_c ← v_n
            else
                if random[0..1) < exp{(f(v_n) - f(v_c))/T}
                    then   v_c ← v_n
        until (termination-condition)
        T ← g(T,t)
        t ← t + 1
    until (stop-criterion)
end
```

Hillclimbing, Simulated Annealing, and Genetic Algorithms

- Comparison:

GA vs. Conventional Optimization

Algorithm performance

- Never draw any conclusion from a single run.
  - Use statistical measures (averages, medians) from a sufficient number of independent runs.
- From the application point of view
  - design perspective:

    Find a very good solution at least once.
  - production perspective:

    Find a good solution at almost every run.
- Remember the WUTIWYG principl:
  - “What you test is what you get” – don’t tune algorithm performance on toy data and expect it to work with real data.