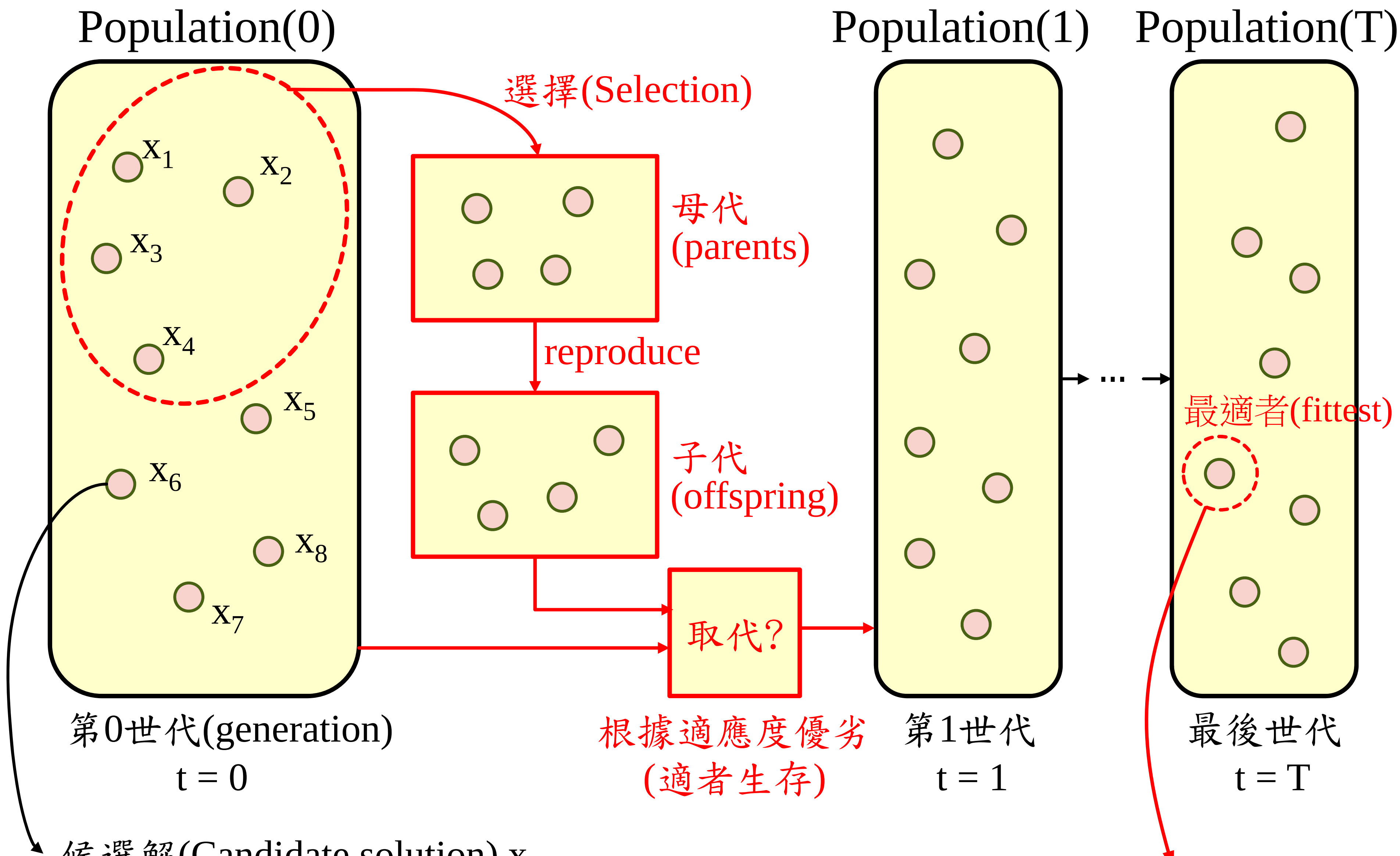
基因演算法
(Genetic Algorithm)
Part 2

基因演算法之示意圖



候選解(Candidate solution) x_i

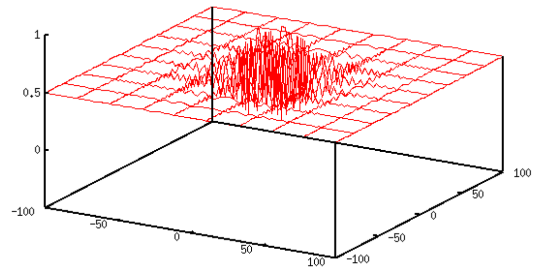
有個適應度(fitness)：$f(x_i)$

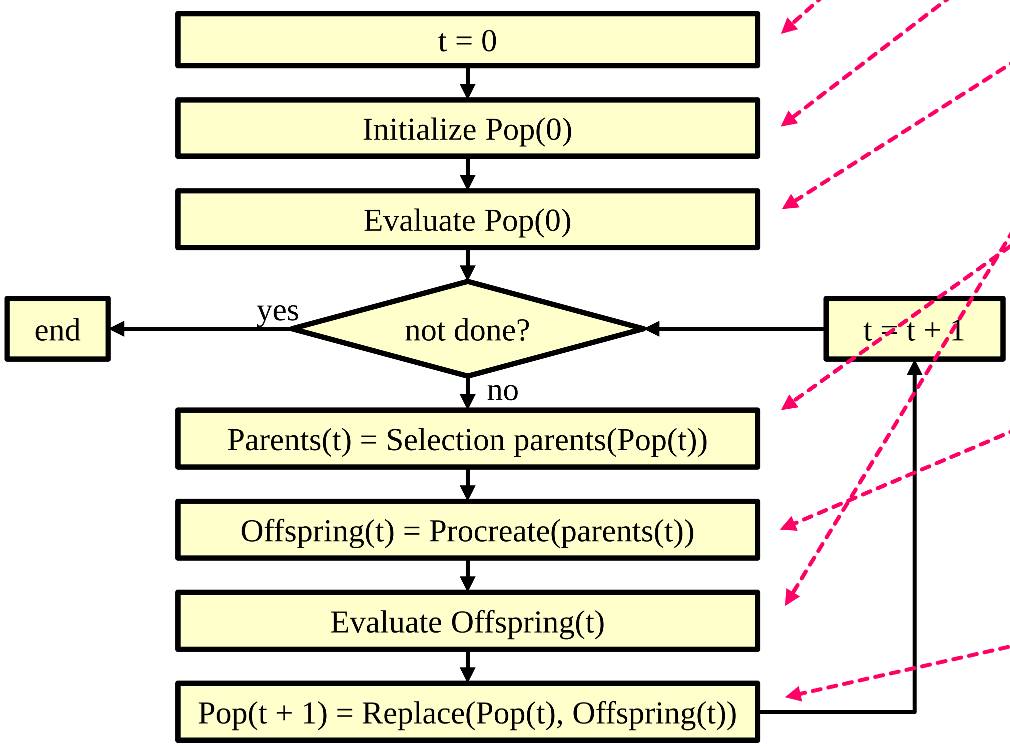
最終解

Review

Max f(x,y)

= 0.5 + (sin(sqrt(x²+y²))² – 0.5) / (1.0 + 0.001(x²+y²))²





1. 編碼：(x_i,y_i)

初始化：(x_i,y_i)隨機從[-100,100] × [-100,100]中取一點

2. 適應度：$fit_i = f(x_i,y_i)$

3. 選擇：隨機從目前的population中選出一對父母(x_{mom}, y_{mon})和(x_{dad}, y_{dad})

4. 繁衍：

$x_{kid} = rnd(x_{mom}, x_{dad}) + N_x(0,\sigma)$

$y_{kid} = rnd(y_{mom}, y_{dad}) + N_y(0,\sigma)$

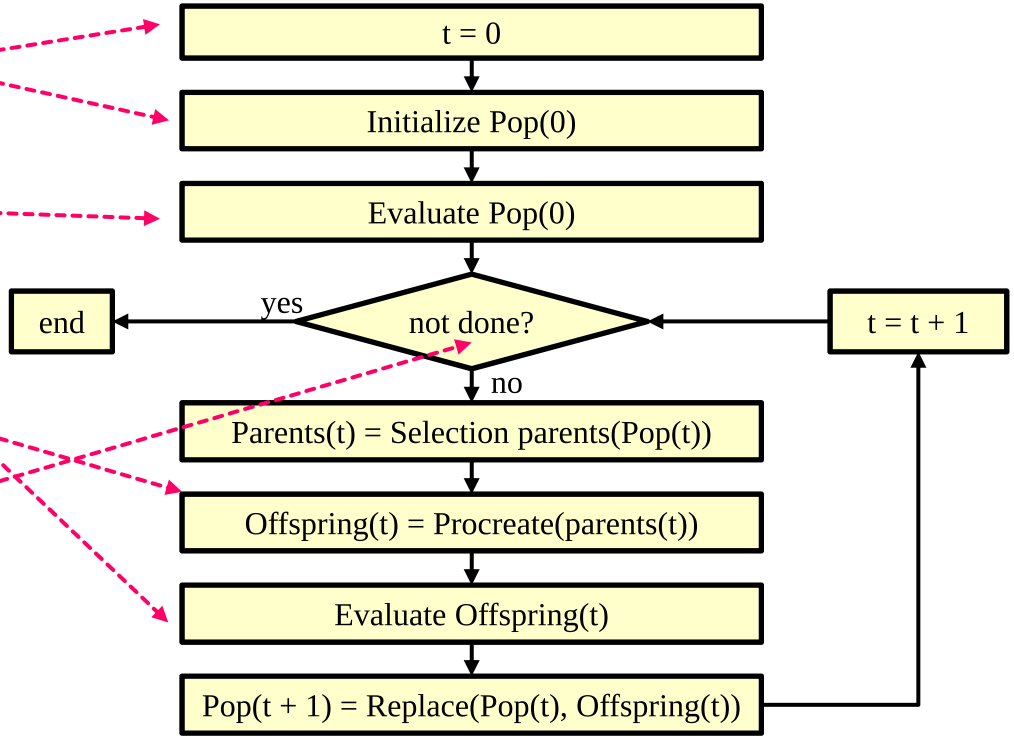
5. 取代：

$Pop(t+1) = \{Pop(t) - \{worst\}\} \cup \{kid\}$

Basic principles

Basic principles

1. Coding or Representation
   - String with all parameters
2. Fitness function
   - Parent selection
3. Reproduction
   - Crossover
   - Mutation
4. Convergence
   - When to stop



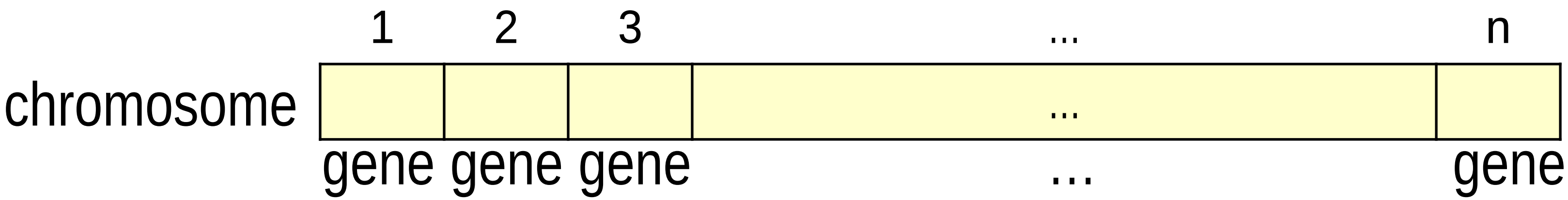
- Link between genetic algorithm and problem:
  - Coding
  - Fitness function

- Reproduction mechanism has no knowledge of the problem to be solved

1. Coding

Coding

- Parameters of the solution (genes) are concatenated to form a string (chromosome)



- All kind of alphabets can be used for a chromosome (numbers, characters),
  but generally a binary alphabet is used
- Order of genes on chromosome can be important
  - Ex1. 若問題有10個0-1的決策變數
  - Ex2. 若問題有5個0-1的決策變數, 5個整數變數
  - Ex3. 若問題的答案是個10個編號的排列

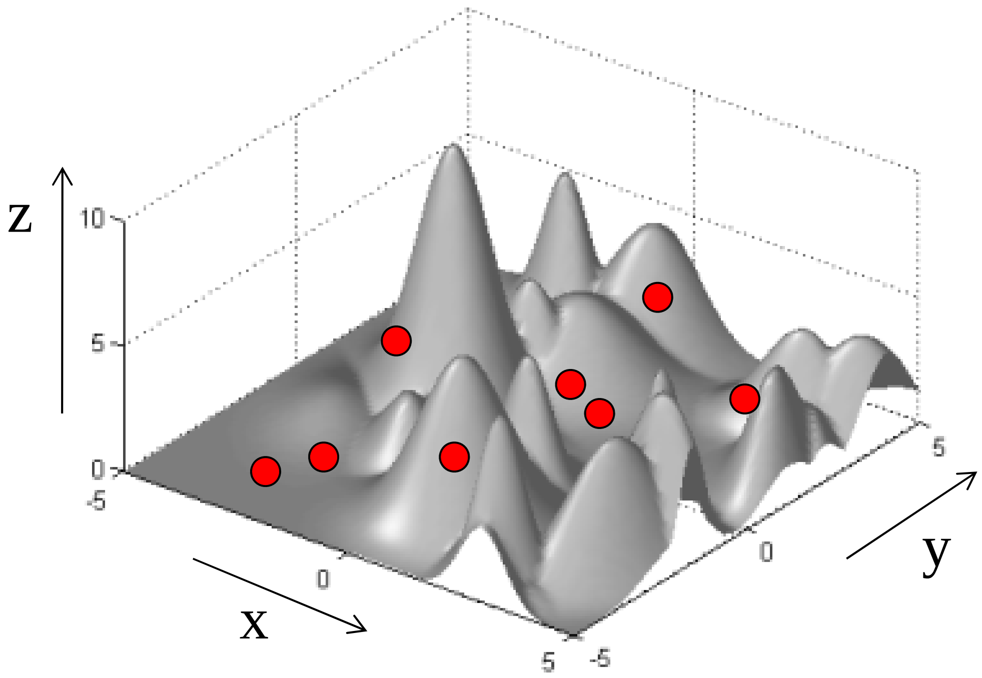
Coding

- Generally many different codings for the parameters of a solution are possible
- Good coding is probably the most important factor for the performance of a GA
- In many cases many possible chromosomes do not code for feasible solutions

2. Fitness Function

Fitness Landscape

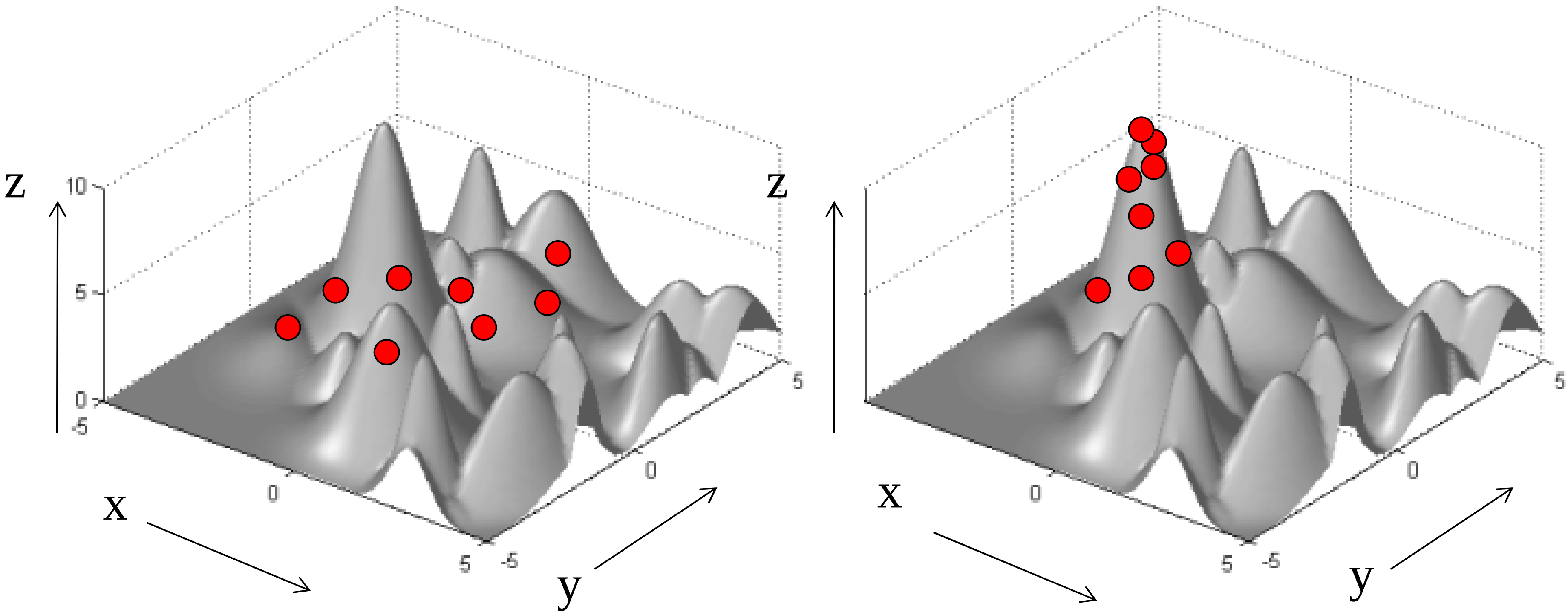
- *n*-dimensional landscape:
  - fitness function is the objective function: $z = f(\vec{x})$
  - $\vec{x} = \{x_1, x_2, \ldots x_n\}$ is the genotype to be optimized
  - peaks: local optima
- 3-D case: intuitive visualization: $z = f(x, y)$



example of an initial population (red dots) on a fitness landscape

Fitness Landscape

- Convergence example:



Fitness Function

Purpose

- Parent selection
- Measure for convergence
- For Steady state: Selection of individuals to die

- Next to coding the most critical part of a GA

2-1: Parent selection

Chance to be selected as parent proportional to fitness

- Roulette wheel (輪盤式選擇法)

To avoid problems with fitness function

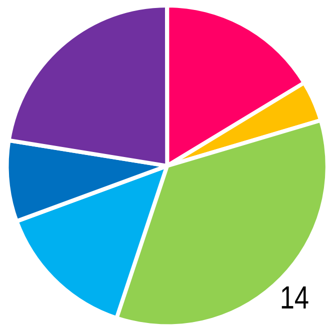
- Tournament (競爭式選擇法)

Not a very important parameter

Roulette wheel

- Sum the fitness of all chromosomes, call it T
- Generate a random number N between 1 and T
- Return chromosome whose fitness added to the running total is equal to or larger than N
- Chance to be selected is exactly proportional to fitness

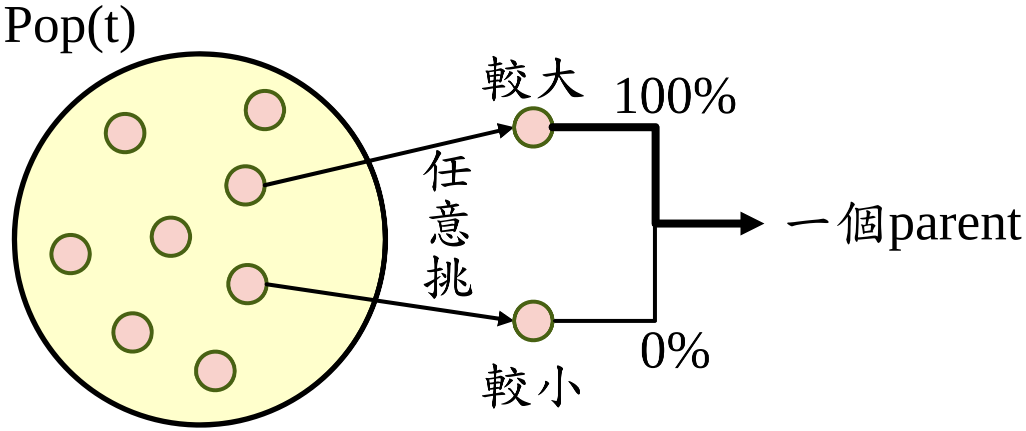
Chromosome :	1	2	3	4	5	6
Fitness:	8	2	17	7	4	11
Running total:	8	10	27	34	38	49
N (1 ≤ N ≤ 49):			23			
Selected:			3			

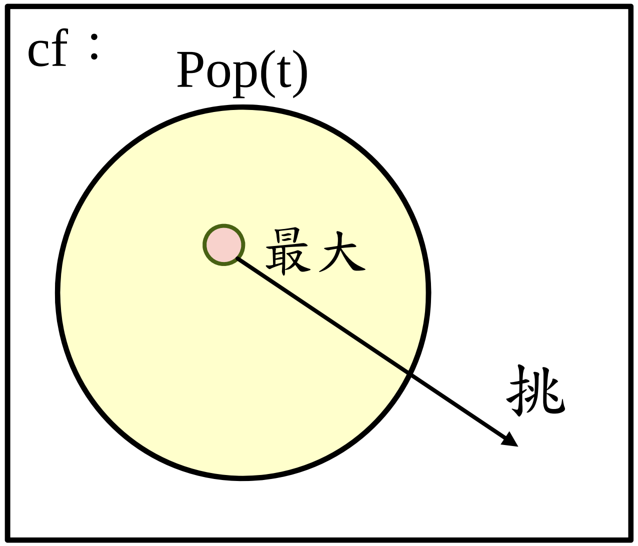


Tournament

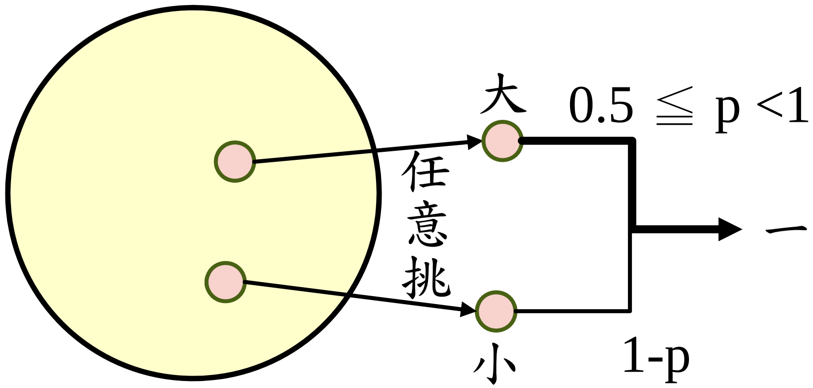
- Binary tournament
  - Two individuals are randomly chosen;
    the fitter of the two is selected as a parent
- Probabilistic binary tournament
  - Two individuals are randomly chosen;
    with a chance p, $0.5<p<1$,
    the fitter of the two is selected as a parent
- Larger tournaments
  - n individuals are randomly chosen;
    the fittest one is selected as a parent
- By changing n and/or p, the GA can be adjusted dynamically

1

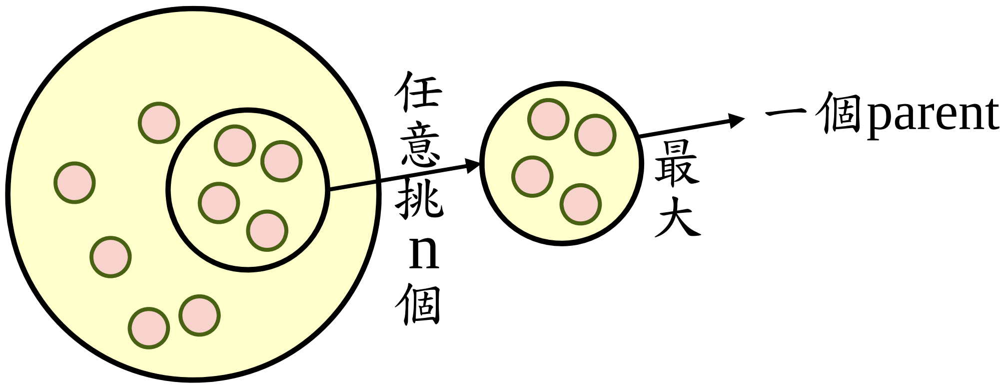

Pop(t)
較大
100%
任意挑
一個parent
0%
較小


cf：
Pop(t)
最大
挑

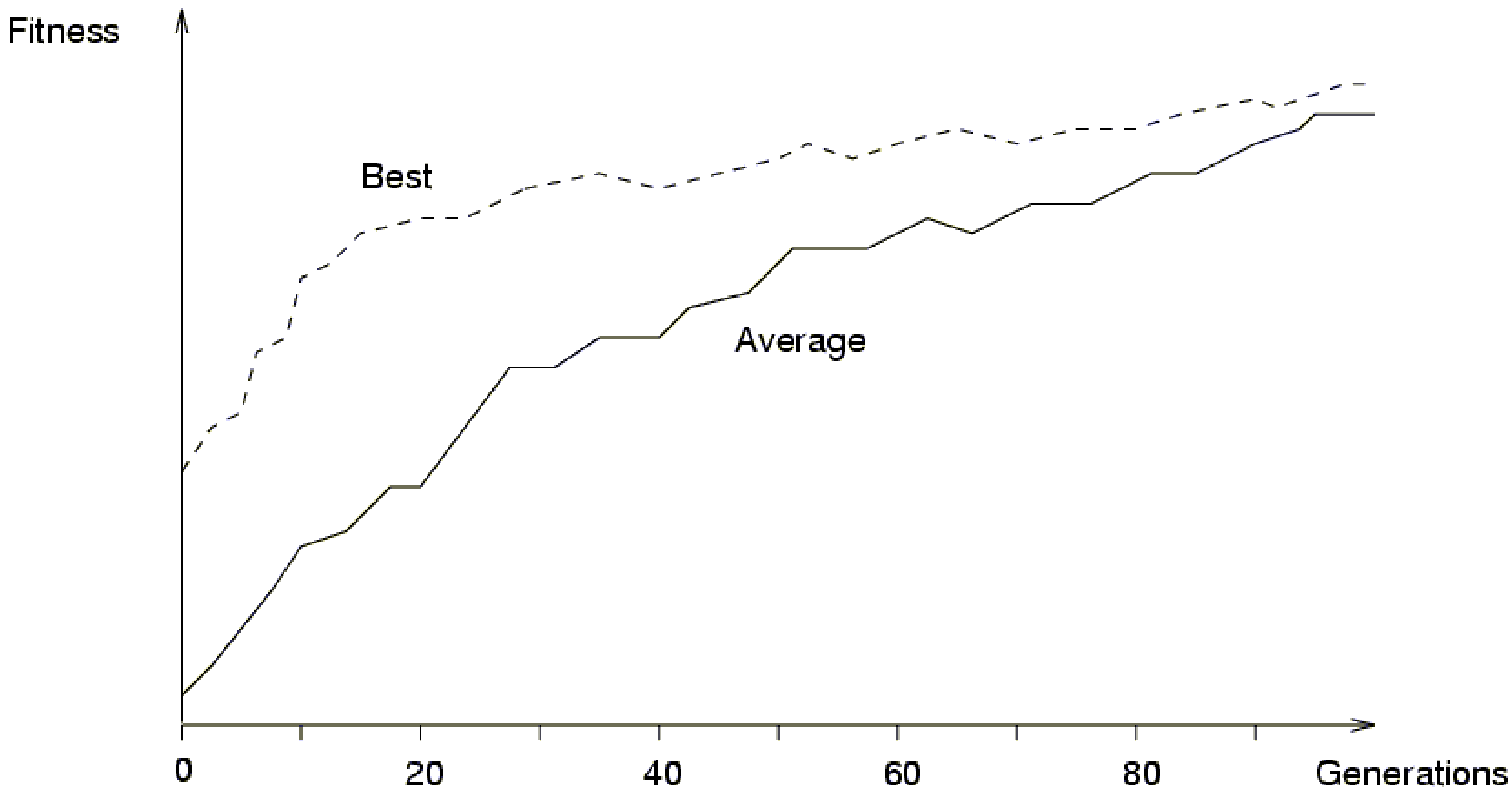
2


大
0.5 ≦ p <1
任意挑
一個parent
小
1-p

3


任意挑n個
一個parent
最大

2-2: Example of convergence



3. Reproduction

Reproduction

- Crossover (交配)
  - Two parents produce two offspring
  - Generally the **chance of crossover** is between 0.6 and 1.0
    - ✓ There is a chance that the chromosomes of the two parents are copied unmodified as offspring
    - ✓ There is a chance that the chromosomes of the two parents are randomly recombined (crossover) to form offspring

- Mutation (突變)
  - There is a chance that a gene of a child is changed randomly
  - Generally the **chance of mutation** is low (e.g. 0.001)

Crossover

- One-point crossover
- Two-point crossover
- Uniform crossover

One-point crossover 1

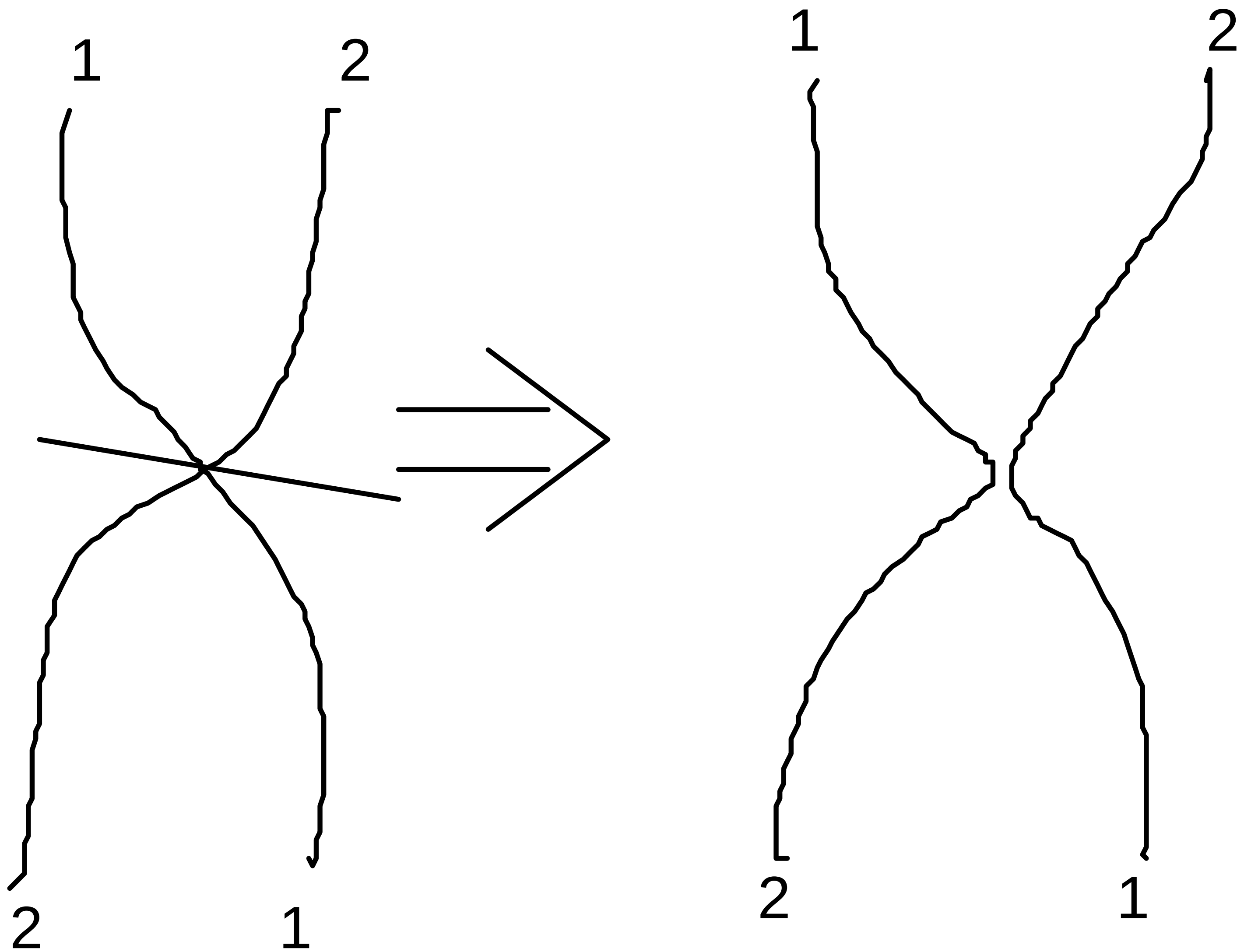
- Randomly one position in the chromosomes is chosen
- Child 1 is head of chromosome of parent 1 with tail of chromosome of parent 2
- Child 2 is head of 2 with tail of 1

Randomly chosen position

Parents: 1010001110 0011010010

Offspring: 1010010010 0011001110

One-point crossover 2



Two-point crossover

- Randomly two positions in the chromosomes are chosen
- Avoids that genes at the head and genes at the tail of a chromosome are always split when recombined

Randomly chosen positions

Parents: 1010001110 0011010010

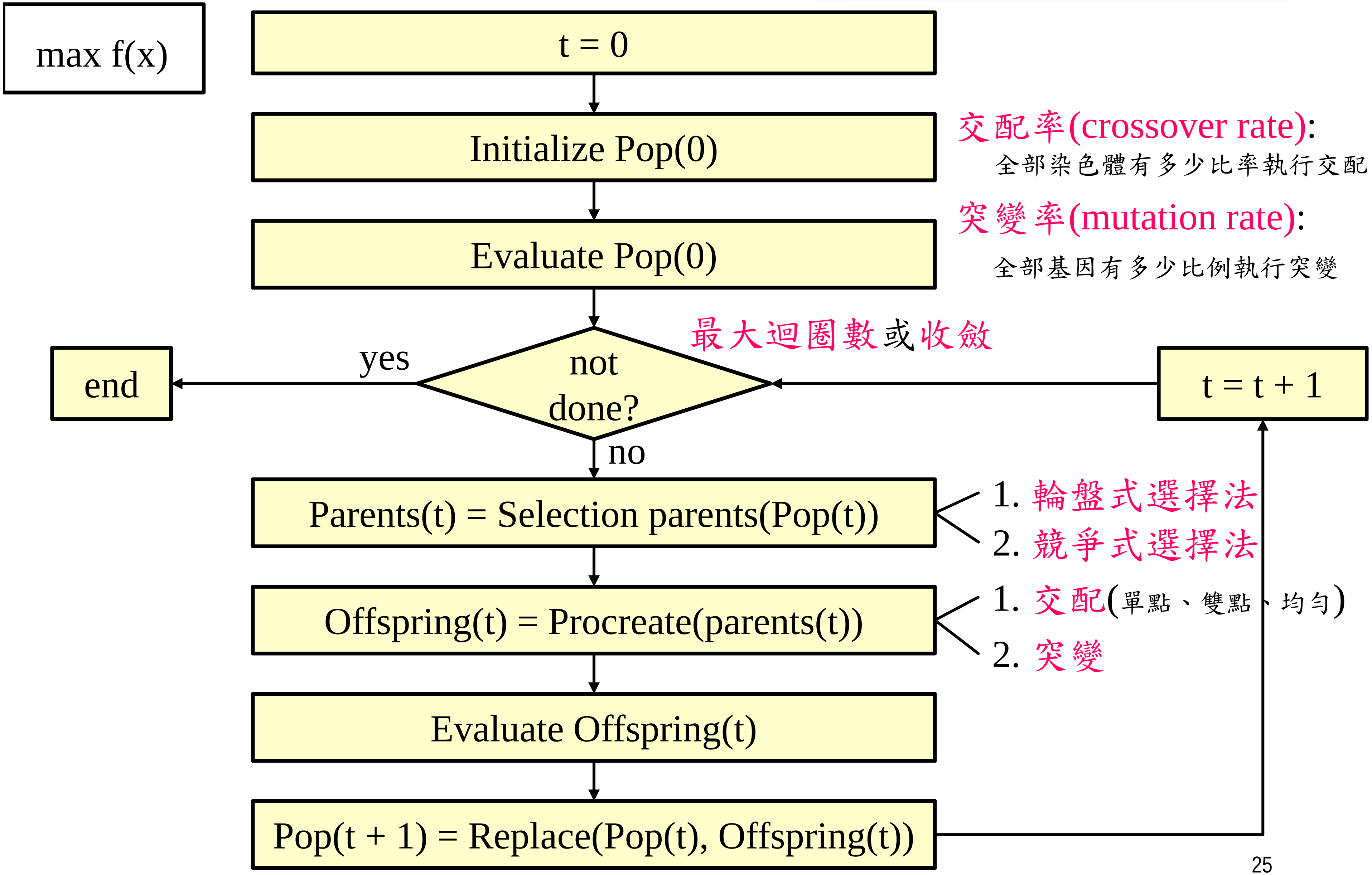
Offspring: 1010010010 0011001110

Uniform crossover

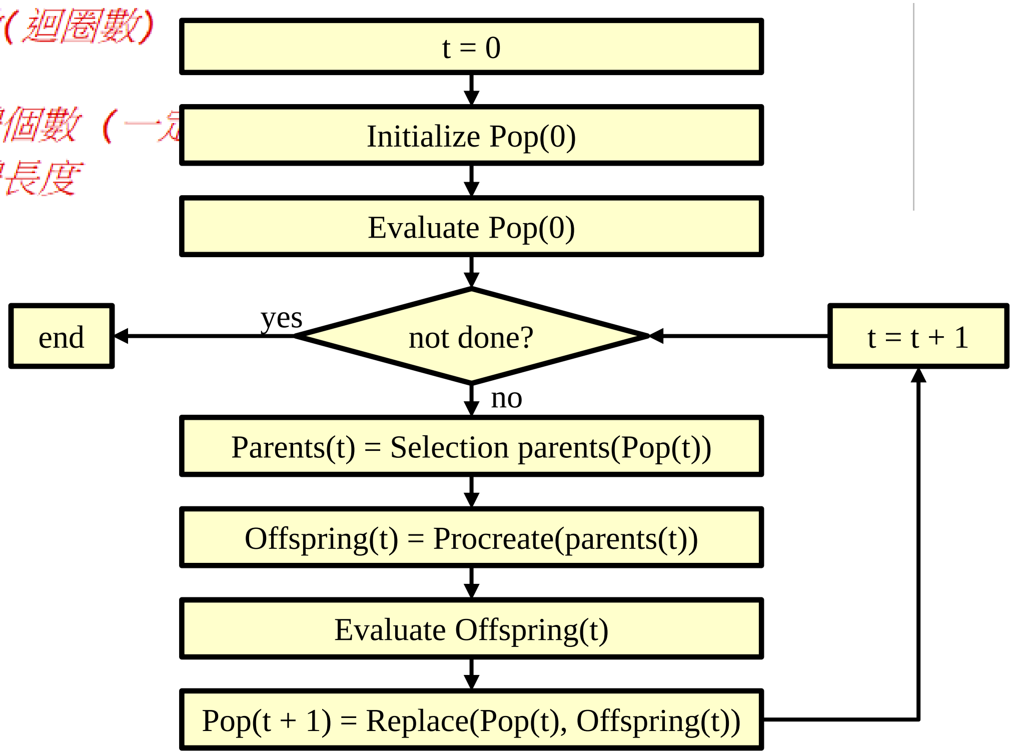
- A random mask is generated
- The mask determines which bits are copied from one parent and which from the other parent
- Bit density in mask determines how much material is taken from the other parent (takeover parameter)

```
Mask:        0110011000     (Randomly generated)
Parents:     1010001110     0011010010

Offspring:   0011001010     1010010110
```

基因演算法流程圖


max f(x)
t = 0
Initialize Pop(0)
Evaluate Pop(0)
not done?
yes
end
no
最大迴圈數或收斂
t = t + 1
Parents(t) = Selection parents(Pop(t))
Offspring(t) = Procreate(parents(t))
Evaluate Offspring(t)
Pop(t + 1) = Replace(Pop(t), Offspring(t))
交配率(crossover rate):
全部染色體有多少比率執行交配
突變率(mutation rate):
全部基因有多少比例執行突變
1. 輪盤式選擇法
2. 競爭式選擇法
1. 交配(單點、雙點、均勻)
2. 突變

```python
NUM_ITERATION = 20            # 世代數(迴圈數)

NUM_CHROME = 20               # 染色體個數 (一
NUM_BIT = 6                   # 染色體長度
```




```python
# ==== 主程式 ====
pop = initPop()             # 初始化 pop
pop_fit = evaluatePop(pop)  # 算 pop 的 fit

for i in range(NUM_ITERATION) :
    parent = selection(pop, pop_fit)            # 挑父母
    offspring = crossover(parent)               # 交配
    mutation(offspring)                         # 突變
    offspring_fit = evaluatePop(offspring)      # 算子代的 fit
    pop, pop_fit = replace(pop, pop_fit, offspring, offspring_fit)    # 取代

    print('iteration %d: x = %s, y = %d'    %(i, pop[0], pop_fit[0]))
```

```python
NUM_ITERATION = 20            # 世代數(迴圈數)

NUM_CHROME = 20               # 染色體個數 (一定要偶數)
NUM_BIT = 6                   # 染色體長度
```

1. 編碼： 6個二元編碼，000000~111111
初始化： 隨機設為6個二元編碼

```python
def initPop():                # 初始化群體
    return np.random.randint(2, size=(NUM_CHROME,NUM_BIT)) # 產生 NUM_CHROME 個二元編碼
```

```python
def fitFunc(x):               # 適應度函數
    # 將[1, 2, ..., NUM_BIT-1]的二元數轉成整數(第0數是符號數)
    fitness = int("".join(str(i) for i in x[1:NUM_BIT]), 2)
    return 1024 - fitness * fitness

def evaluatePop(p):           # 評估群體之適應度
    return [fitFunc(p[i]) for i in range(len(p))]
```

2. 適應度： 轉成10進位x_d，然後計算 $fit = 1024 - x_d^2$

```python
# ==== 主程式 ====
pop = initPop()               # 初始化 pop
pop_fit = evaluatePop(pop)    # 算 pop 的 fit

for i in range(NUM_ITERATION) :
    parent = selection(pop, pop_fit)              # 挑父母
    offspring = crossover(parent)                 # 交配
    mutation(offspring)                           # 突變
    offspring_fit = evaluatePop(offspring)        # 算子代的 fit
    pop, pop_fit = replace(pop, pop_fit, offspring, offspring_fit)    # 取代

    print('iteration %d: x = %s, y = %d'    %(i, pop[0], pop_fit[0]))
```

```python
Pc = 0.5                      # 交配率 (代表共執行Pc*NUM_CHROME/2次交配)
Pm = 0.01                     # 突變率 (代表共要執行Pm*NUM_CHROME*NUM_BIT次突變)

NUM_PARENT = NUM_CHROME                          # 父母的個數
NUM_CROSSOVER = int(Pc * NUM_CHROME / 2)         # 交配的次數
NUM_CROSSOVER_2 = NUM_CROSSOVER*2                # 上數的兩倍
NUM_MUTATION = int(Pm * NUM_CHROME * NUM_BIT)    # 突變的次數
```

0.5 * 20 /2 → 5

0.01 * 20 * 6 → 1

```python
def selection(p, p_fit):   # 用二元競爭式選擇法來挑父母
    a = []
    for i in range(NUM_PARENT):
        [j, k] = np.random.choice(NUM_CHROME, 2, replace=False)  # 任選兩個index
        if p_fit[j] > p_fit[k] :                      # 擇優
            a.append(p[j])
        else:
            a.append(p[k])

    return a
```

3. 選擇：二元競爭式選擇法

```python
def crossover(p):            # 用單點交配來繁衍子代
    a = []
    for i in range(NUM_CROSSOVER) :
        c = np.random.randint(1, NUM_BIT)                   # 隨機找出單點(不包含0)
        [j, k] = np.random.choice(NUM_PARENT, 2, replace=False)  # 任選兩個index

        a.append(np.concatenate((p[j][0: c], p[k][c: NUM_BIT]), axis=0))
        a.append(np.concatenate((p[k][0: c], p[j][c: NUM_BIT]), axis=0))

    return a
```

4. 交配：單點交配

```python
Pc = 0.5                           # 交配率 (代表共執行Pc*NUM_CHROME/2次交配)
Pm = 0.01                          # 突變率 (代表共要執行Pm*NUM_CHROME*NUM_BIT次突變)

NUM_PARENT = NUM_CHROME                           # 父母的個數
NUM_CROSSOVER = int(Pc * NUM_CHROME / 2)          # 交配的次數
NUM_CROSSOVER_2 = NUM_CROSSOVER*2                 # 上數的兩倍
NUM_MUTATION = int(Pm * NUM_CHROME * NUM_BIT)     # 突變的次數
```

5. 突變：任選一染色體的一個基因，01互換

```python
def mutation(p):                   # 突變
    for _ in range(NUM_MUTATION) :
        row = np.random.randint(NUM_CROSSOVER_2)  # 任選一個染色體
        col = np.random.randint(NUM_BIT)          # 任選一個基因

        p[row][col] = (p[row][col] + 1) % 2       # 對應此染色體的此基因01互換
```

```python
def sortChrome(a, a_fit):          # a的根據a_fit由大排到小
    a_index = range(len(a))                          # 產生 0, 1, 2, ..., |a|-1 的 list

    # a_index 根據 a_fit 的大小由大到小連動的排序
    a_fit, a_index = zip(*sorted(zip(a_fit,a_index), reverse=True))

    # 根據 a_index 的次序來回傳 a，並把對應的 fit 回傳
    return [a[i] for i in a_index], a_fit
```

6. 取代：Pop(t+1) = {Pop(t) - {worsts}} ∪ {kids}

```python
def replace(p, p_fit, a, a_fit):
    b = np.concatenate((p,a), axis=0)                # 把本代 p 和子代 a 合併成 b
    b_fit = p_fit + a_fit                            # 把上述兩代的 fitness 合併成
    b, b_fit = sortChrome(b, b_fit)                  # b 和 b_fit 連動的排序

    return b[:NUM_CHROME], list(b_fit[:NUM_CHROME]) # 回傳 NUM_CHROME 個為新的一
```

Exercise

- Implement the two-point crossover operation in the sample code "GA05-GA-basic-2.py".

Randomly chosen positions

Parents: 1010001110 0011010010

Offspring: 1010010010 0011001110

- (Optional)
  Implement the uniform crossover operation.