2024 12 13

Stochastic Search

随机搜寻

Howard Hao-Chun Chuang (莊皓鈞)

Professor
College of Commerce
National Chengchi University

December, 2024
Taipei, Taiwan

用隨機搜尋做隨機跳躍，試圖找最佳解.

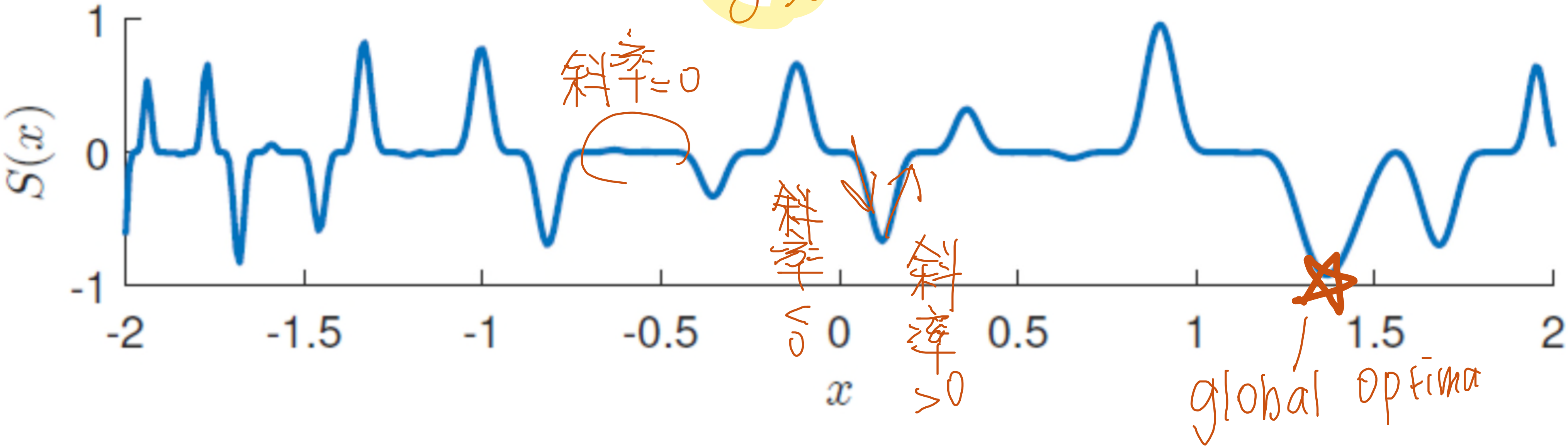
Difficult Optimization

斜率

- **We could easily have multiple local minimums/maximums**

$$S(x) = \begin{cases} -e^{-x^2/100} \sin(13x - x^4)^5 \sin(1 - 3x^2)^2, & \text{if } -2 \leqslant x \leqslant 2, \\ \infty, & \text{otherwise.} \end{cases}$$

$\frac{\partial f(x)}{\partial x} > 0$ 或 < 0 ?



全域最佳解

鍛劍

Simulated Annealing 退火

- *S*(*x*) is an arbitrary function to be minimized, *x* may take values in continuous or discrete set *X*

Simulated annealing is a Monte Carlo technique for minimization that emulates the physical state of atoms in a metal when the metal is heated up and then slowly cooled down. When the cooling is performed very slowly, the atoms settle down to a minimum-energy state. Denoting the state as $\boldsymbol{x}$ and the energy of a state as $S(\boldsymbol{x})$, the probability distribution of the (random) states is described by the *Boltzmann pdf*

$$f(\boldsymbol{x}) \propto \mathrm{e}^{-\frac{S(\boldsymbol{x})}{kT}}, \quad \boldsymbol{x} \in \mathcal{X},$$

#加熱与冷卻
直到找到最小能量狀態,

where k is Boltzmann's constant and T is the temperature.

Simulated Annealing

The idea of simulated annealing is to create a sequence of points $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots$ that are approximately distributed according to pdfs $f_{T_1}(\boldsymbol{x}), f_{T_2}(\boldsymbol{x}), \ldots$, where $T_1, T_2, \ldots$ is a sequence of "temperatures" that decreases (is "cooled") to 0 — known as the *annealing schedule*. If each $\boldsymbol{X}_t$ were sampled *exactly* from f_{T_t}, then $\boldsymbol{X}_t$ would converge to a global minimum of $S(\boldsymbol{x})$ as $T_t \to 0$. However, in practice sampling is *approximate* and convergence to a global minimum is not assured. A generic simulated annealing algorithm is as follows.

temperature

Algorithm 3.4.1: Simulated Annealing

input: Annealing schedule $T_0, T_1, \ldots,$, function S, initial value $\boldsymbol{x}_0$.

output: Approximations to the global minimizer $\boldsymbol{x}^*$ and minimum value $S(\boldsymbol{x}^*)$.

1 Set $\boldsymbol{X}_0 \leftarrow \boldsymbol{x}_0$ and $t \leftarrow 1$.

2 **while** not stopping **do**

3   Approximately simulate $\boldsymbol{X}_t$ from $f_{T_t}(\boldsymbol{x})$.

4     $t \leftarrow t + 1$

5 **return** $\boldsymbol{X}_t, S(\boldsymbol{X}_t)$

離散、連續都試用

Min $S(x)$

Algorithm 3.4.2: Simulated Annealing with a Random Walk Sampler

input: Objective function S, starting state X_0, initial temperature T_0, number of iterations N, symmetric proposal density $q(\mathbf{y} \mid \mathbf{x})$, constant β.

output: Approximate minimizer and minimum value of S.

for $t = 0$ **to** $N - 1$ **do**

  New old

  Simulate a new state $\mathbf{Y}$ from the symmetric proposal $q(\mathbf{y} \mid \mathbf{X}_t)$.

  continuous

  if $S(\mathbf{Y}) < S(\mathbf{X}_t)$ **then**

    $\mathbf{X}_{t+1} \leftarrow \mathbf{Y}$ // accept

  $Y' = \lceil Y \rceil \to$ ceiling

  smallest integer $> Y$

  else // $S(y) > S(x)$

    Draw $U \sim \mathcal{U}(0, 1)$. $(\leq 1, \geq 0)$

    if $U \leqslant e^{-(S(\mathbf{Y}) - S(\mathbf{X}_t))/T_t}$ **then**

      $\mathbf{X}_{t+1} \leftarrow \mathbf{Y}$ 取代 W_{old}

    else

      $\mathbf{X}_{t+1} \leftarrow \mathbf{X}_t$ 放棄，再重跳

  $T_{t+1} \leftarrow \beta T_t$

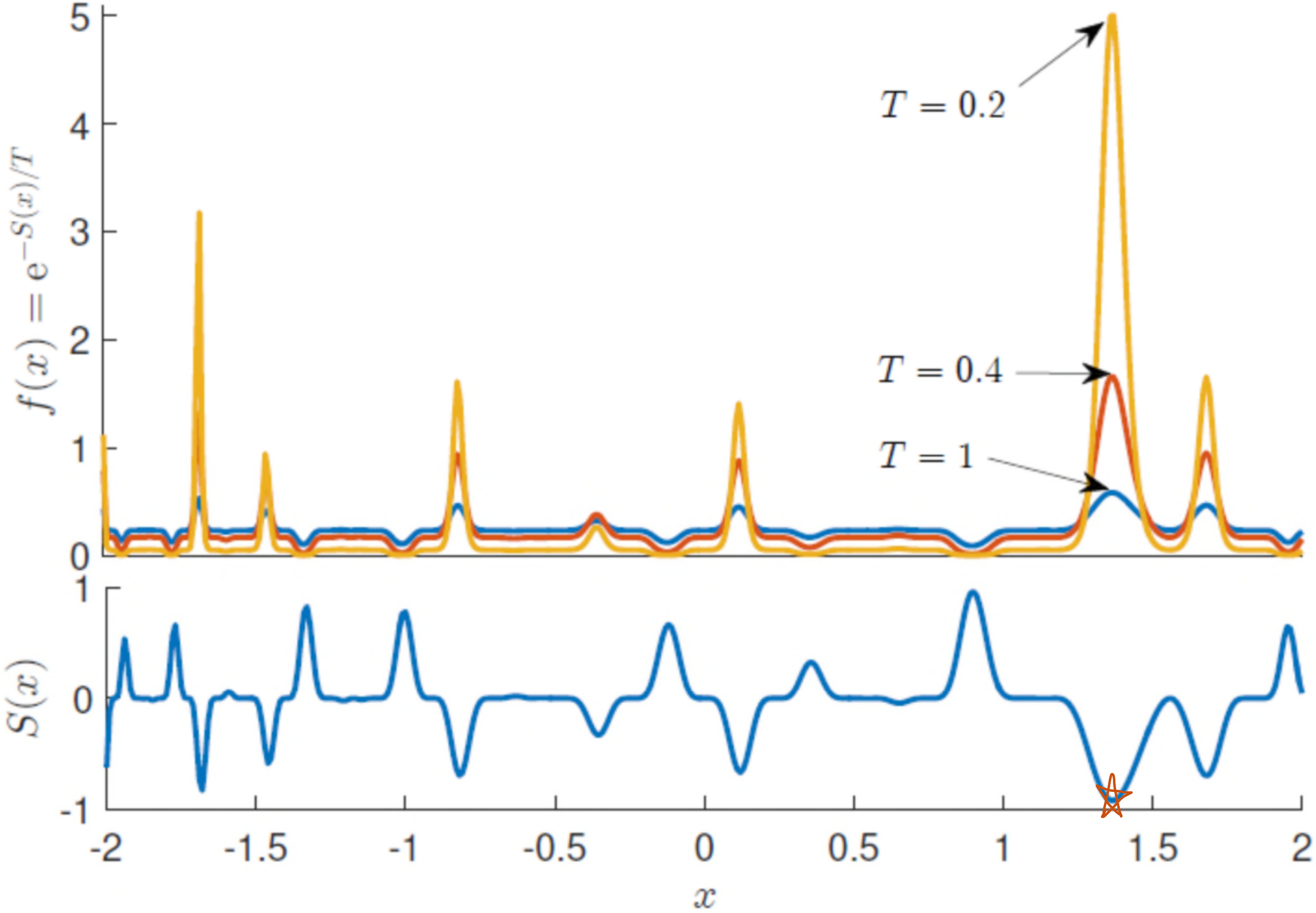
return $\mathbf{X}_N$ and $S(\mathbf{X}_N)$

$W_{new} \sim Normal(\mu = W_{old}, \sigma)$

反覆 Jump

skebook

Simulated Annealing



Multivariable Optimization

$ budget

R1 45:00

- **Consider the following assets in 2017/11/10 ~2020/11/10**
  APPLE, FACEBOOK, AMAZON, S&P500, NASDAQ, WWE (?)

  Build a portfolio: Six continuous decision variables *x* in [0, 1]

  $\Sigma x = 1$ $x \leq 1$ $x \geq 0$

  Daily return is {price(*t*)-price(*t*-1)}/price(*t*-1)
  Maximize the Sharpe ratio=E[Daily Return]/(Var[Daily Return])^0.5
  see https://en.wikipedia.org/wiki/Sharpe_ratio

- **Min objective = -1*Sharpe + lambda*(constraint)**

  penalty > 0

  minimum

  How can we accommodate key constraints?
  ((x1+x2+x3+x4+x5+x6)-1)^2 $\Rightarrow \Sigma x = 1$
  for $x_i \leq 1$ — **sum(max(0, x[*i*]-1)^2, for *i*=1,2…,6)**
  for $x_i \geq 0$ — **sum(max(0, -x[*i*])^2, for *i*=1,2…,6)**

  if $x_i = -2$

  Max $\frac{\text{return}}{\text{risk}}$

(套件)
pymo

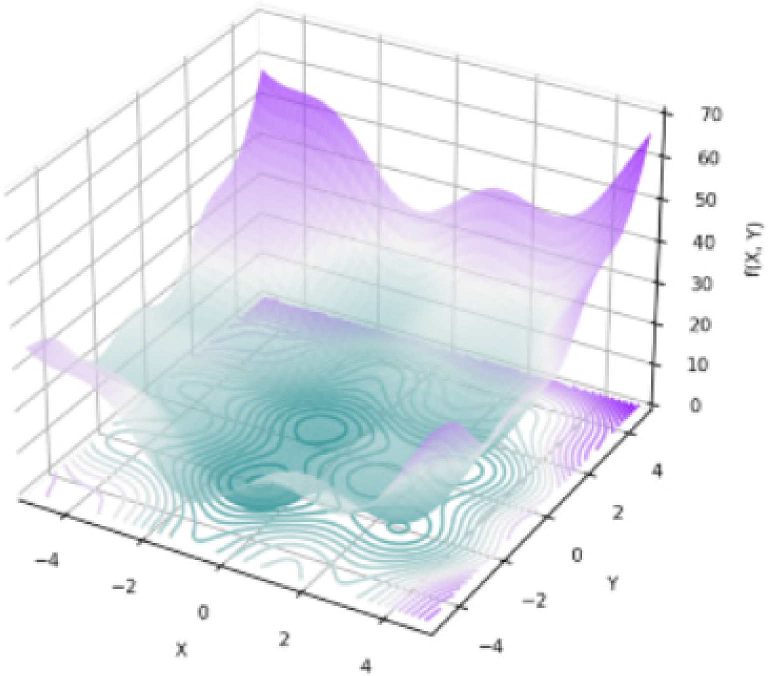
粒子 群

Particle Swarm Optimization, PSO

- **Inspired by social behavior**

  We make decisions by self-experience & group-experience

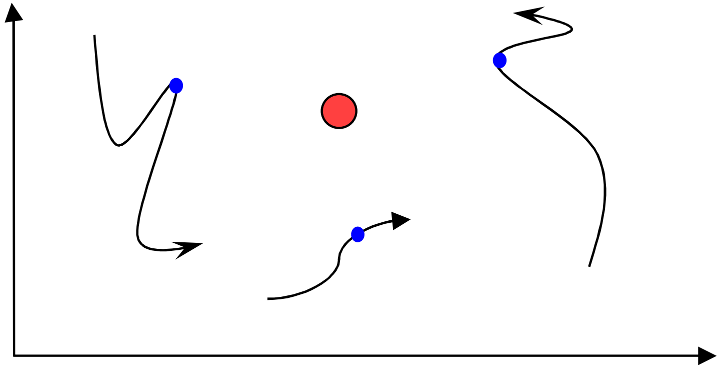
A group of birds is searching for food, how to move in a valley?



$$f(x,y)=x^2+(y+1)^2-5\cos(3x/2+3/2)$$
$$-3\cos(2y-3/2)$$

Particle Swarm Optimization, PSO

- **Let's simplify the case into 2 dimensions & only 3 particles**



- $\boldsymbol{X_{n,t}}$ **: travel route of *n* particles in time period *t***

  P(*aaron*) = [$X_{a,1}$,$X_{a,2}$,$X_{a,3}$ .... $X_{a,t}$], min at $X_{a,2}$ 過往最佳解 / 自己最棒的

  P(*betty*) = [$X_{b,1}$,$X_{b,2}$,$X_{b,3}$ .... $X_{b,t}$], min at $X_{b,3}$

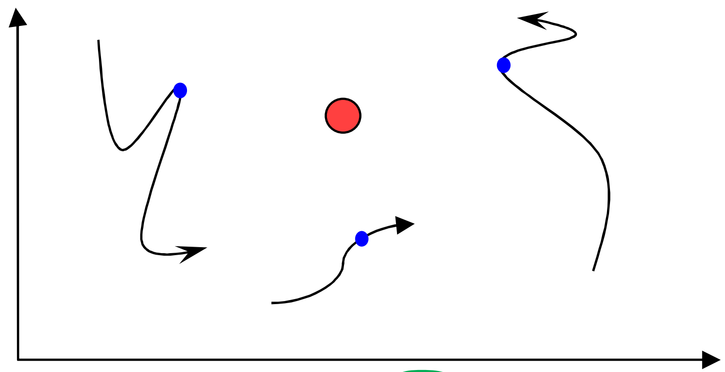
  P(*cathy*) = [$X_{c,1}$,$X_{c,2}$,$X_{c,3}$ .... $X_{c,t}$], min at $X_{c,t}$

- **Next position for each particle is calculated by:**

  $\boldsymbol{X_{n,t+1} = X_{n,t} + V_{n,t+1}}$ // adjustment

Particle Swarm Optimization, PSO

- **Let's simplify the case into 2 dimensions & only 3 particles**



- **Now, we have to decide $V_{n,t+1}$ by *pbest* and *gbest***

$$V_{n,t+1} = w*V_{n,t} + c1*r1(V_{pbest}) + c2*r2(V_{gbest})$$

r是隨機亂數

***r1* & *r2* ~ Uniform(0, 1)** past self_best group best

Take particle *aaron* for instance

***pbest* : best self-experience: $X_{a,2}$**

***gbest* : best group-experience: $X_{c,t}$**

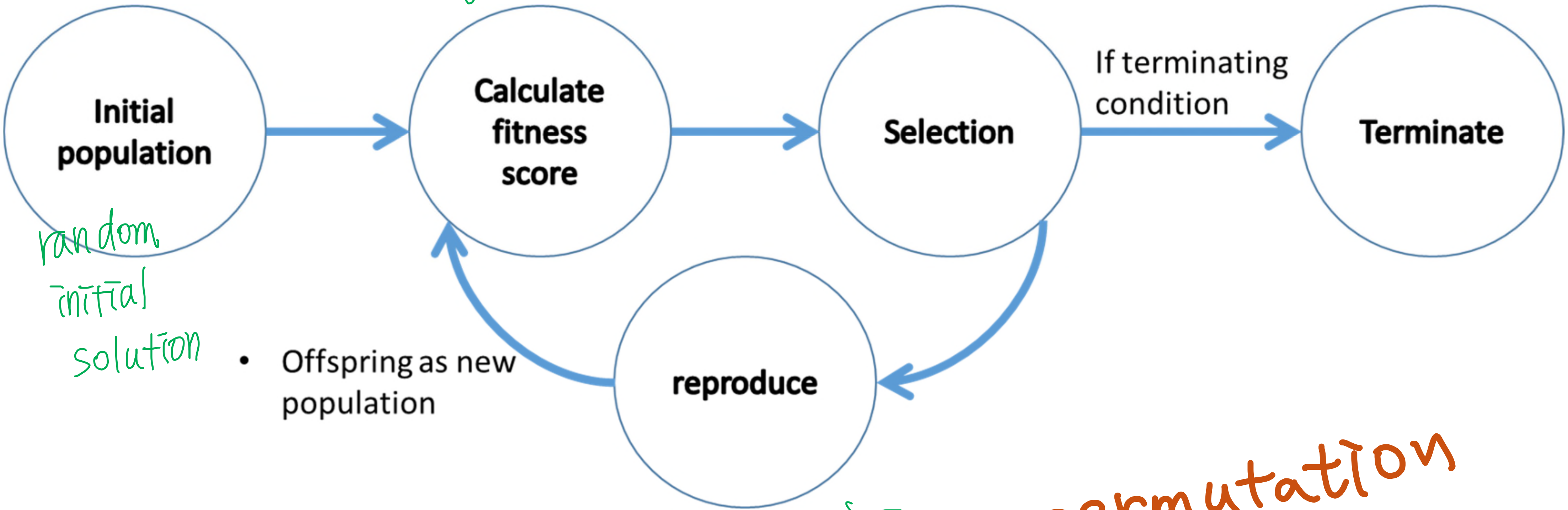
$$V_{a,t+1} = w*V_{a,t} + c1*r1*(X_{a,2} - X_{a,t}) + c2*r2*(X_{c,t} - X_{a,t})$$

多少機率/比例可以達到 → 自己曾經最佳位置 目前位置 能否複製 group best

Genetic Algorithms (GA)

evolutionary 基因algo

objective value



random initial solution

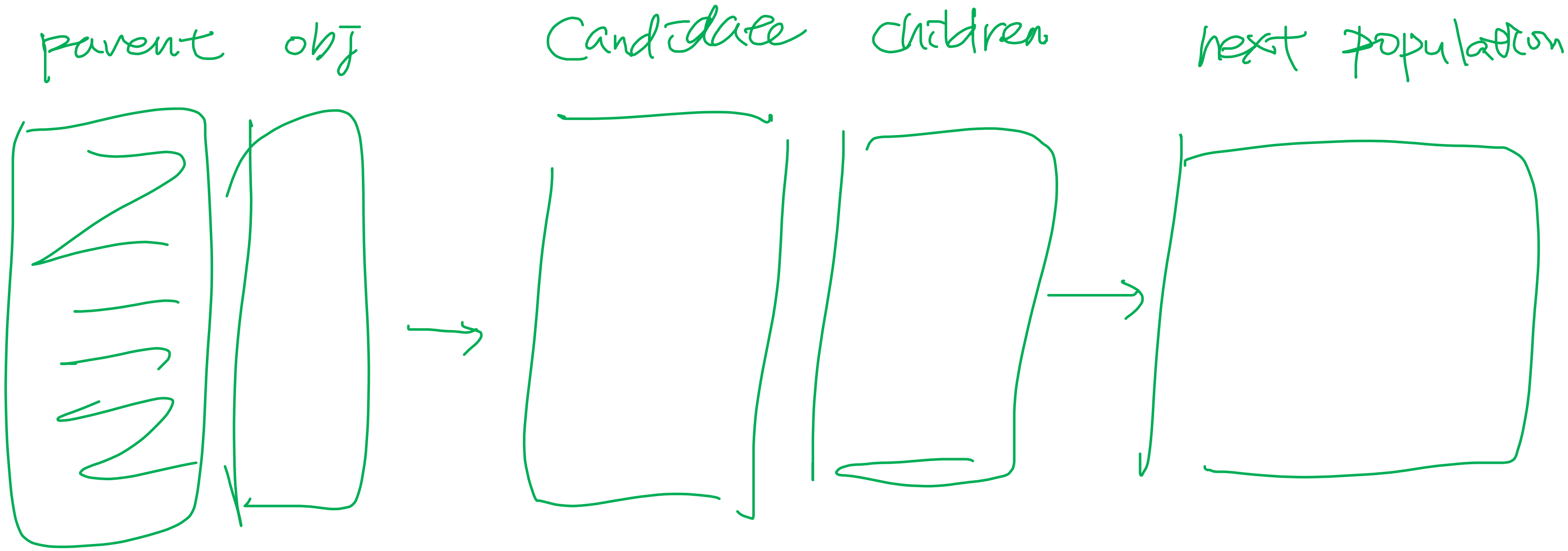
- Offspring as new population

- Crossover 交配
- Mutation 突變

permutation 排列!

Genetic Algorithms (GA)

- **An extremely popular optimization algorithm**
  1. **Initial population with pop_size**
  2. **Calculate fitness score of population**
  3. **Randomly select pop_size of chromosome with replacement**



Genetic Algorithms (GA)

- **Selection**

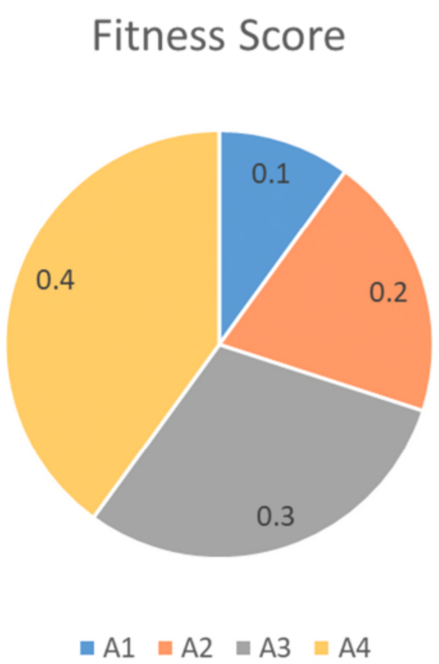
Chromosome	Fitness Score	Proportion
A1	10	0.1
A2	20	0.2
A3	30	0.3
A4	40	0.4

Min?

Roulette wheel selection

$$\frac{\frac{1}{10}}{\frac{1}{10}+\frac{1}{20}+\frac{1}{30}+\frac{1}{40}}$$

$$\frac{\frac{1}{40}}{\frac{1}{10}+\frac{1}{20}+\frac{1}{30}+\frac{1}{40}}$$



Genetic Algorithms (GA)

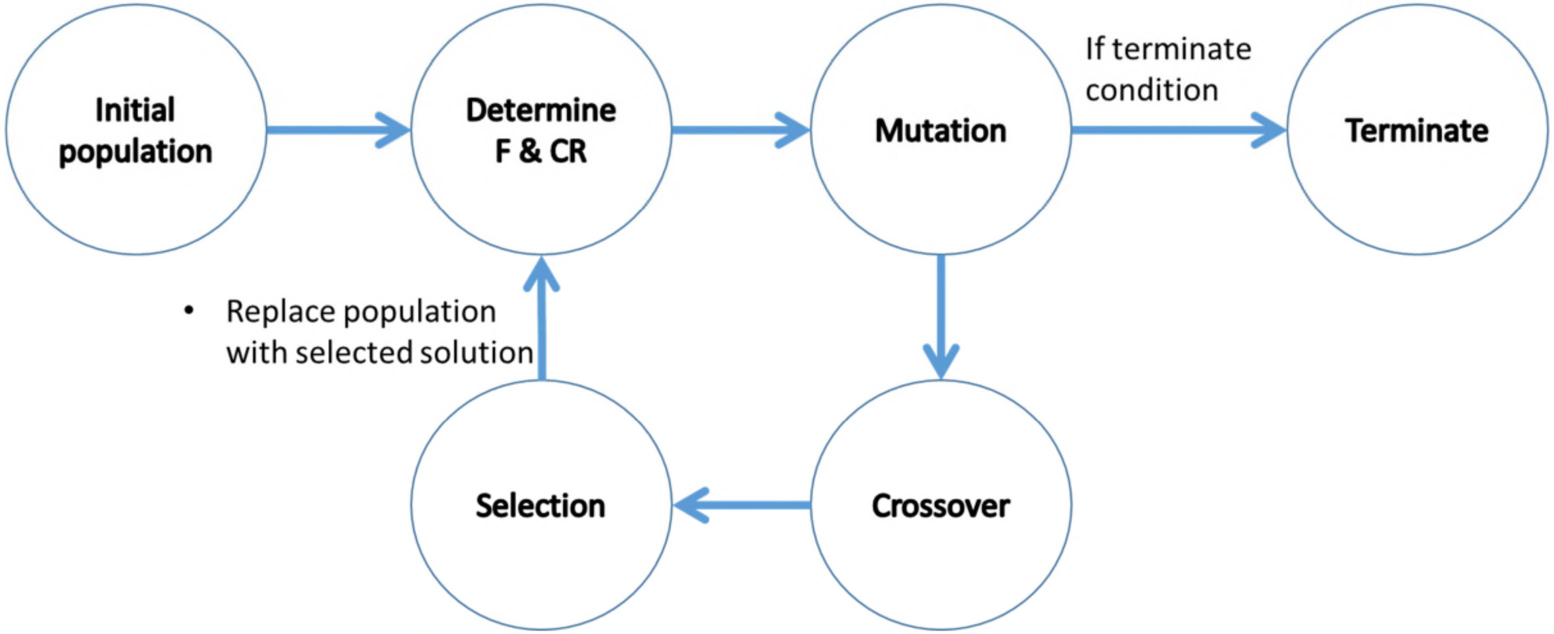
- **An extremely popular optimization algorithm**
  1. **Initial population with pop_size**
  2. **Calculate fitness score of population**
  3. **Randomly select pop_size of chromosome with replacement**
  4. **if $rand_1 \leq pcrossover$, make population into pairs and do crossover**
  5. **If $rand_2 \leq pmutation$, for each child, do mutate**
  6. **select elitism of highest fitness scores from previous population & randomly select (pop_size – elitism) of children as new population**

hyper-parameters

pop_size : population size
pcrossover : probability of crossover
pmutation : probability of mutation
elitism : number of best fitness to survive
maxiter : max iteration

Differential Evolution (DE)

- Choose the parameters $\mathrm{NP} \geq 4$, $\mathrm{CR} \in [0,1]$, and $F \in [0,2]$.
  - NP is the population size, i.e. the number of candidate agents or "parents"; a typical setting is $10n$.
  - The parameter $\mathrm{CR} \in [0,1]$ is called the *crossover probability* and the parameter $F \in [0,2]$ is called the *differential weight*. Typical settings are $F = 0.8$ and $CR = 0.9$.
  - Initialize all agents $\mathbf{x}$ with random positions in the search-space.



Differential Evolution (DE)

- For each agent $\mathbf{x}$ in the population do:
  - Pick three agents $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ from the population at random, they must be distinct from each other as well as from agent $\mathbf{x}$. ($\mathbf{a}$ is called the "base" vector.)
  - Pick a random index $R \in \{1, \ldots, n\}$ where n is the dimensionality of the problem being optimized.

$U(0,1)$

  - Compute the agent's potentially new position $\mathbf{y} = [y_1, \ldots, y_n]$ as follows:
    - For each $i \in \{1, \ldots, n\}$, pick a uniformly distributed random number $r_i \sim U(0,1)$
    - If $r_i < CR$ or $i = R$ then set $y_i = a_i + F \times (b_i - c_i)$ otherwise set $y_i = x_i$. (Index position R is replaced for certain.)
  - If $f(\mathbf{y}) \leq f(\mathbf{x})$ then replace the agent $\mathbf{x}$ in the population with the improved or equal candidate solution $\mathbf{y}$.

https://en.wikipedia.org/wiki/Differential_evolution