

Lecture

Metaheuristic Methods for Machine Learning [Part-1] (機器學習的元啟發式最佳化演算法, Part-1)

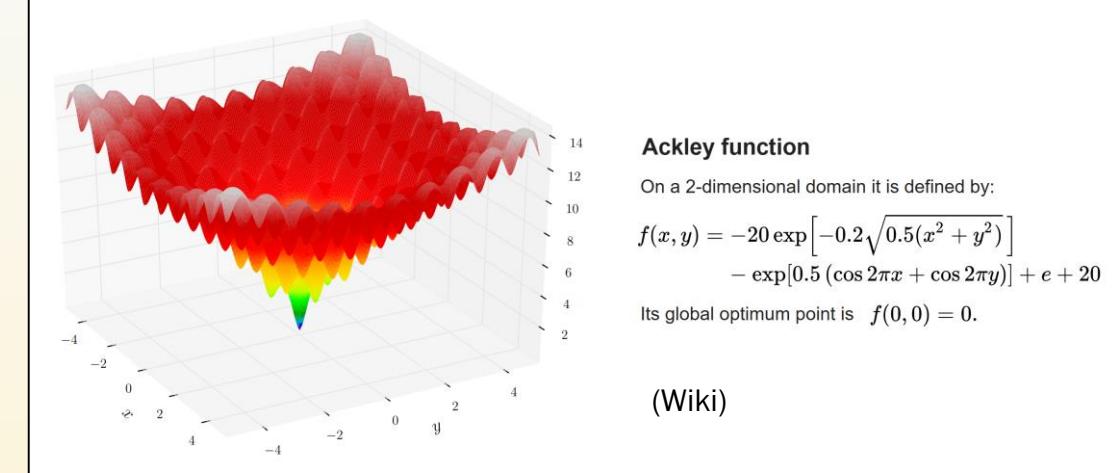
Outlines

1. Random Optimization (RO, 隨機最佳化演算法)
2. Particle Swarm Optimization (PSO, 粒子群最佳化演算法)
3. Whale Optimization Algorithm (WOA, 鯨群最佳化演算法)
4. Ant Colony Optimization (ACO, 蟻群最佳化演算法)

Lecture Objective

(學習目標)

無導數最佳化(Derivative-free optimization, 或稱黑盒最佳化 blackbox optimization)是數學最佳化中的一門學科,它不使用經典意義上的導數資訊(Derivative information)來尋找最佳解,這是因為有時有關損失函數[1]的導數資訊處於不可用、不可靠或不切實際的情況。例如,損失函數可能是不平滑的,或損失函數的評估是很耗時的,或在某種程度上有噪訊的,因此依賴導數或透過有限差分(Finite differences)來近似導數的方法幾乎沒有用處。在這種情況下尋找最佳解的問題稱為無導數最佳化,而不使用導數或有限差分的演算法稱為無導數演算法。在無導數最佳化中,一些方法可以被證明能夠發現極值(Optima),而有些則更像是元啟發法(Metaheuristics),因為與凸優化(Convex optimization)相比,這些問題通常更難解決。對於這些問題,我們企圖以足夠的資源有效地找到滿足解,這種解可能接近最佳解,但通常無法保證是最優性的。最佳化問題是多樣化的(diverse),因此通常無法對所有類型的問題使用一種算法。本講討論多種元啟發式演算法,它們都是無導數最佳化之方法,可用於機器學習。



[1] 測試函數(Test functions)可用於評估最佳化演算法的特徵,例如,收斂率(Convergence rate)、精確性(Precision)、穩健性(Robustness)、及其他一般效能。測試函數的取得途徑,參考下列鏈結。

- [Test functions for optimization - Wikipedia](#)
- [Funzione test \(ottimizzazione\) - Wikipedia](#)
- <https://www.mathworks.com/matlabcentral/fileexchange/23147-test-functions-for-global-optimization-algorithms>
- https://mpra.ub.uni-muenchen.de/2718/1/MPRA_paper_2718.pdf

Metaheuristic Methods

在優化(即最佳化, [optimization](#) or optimisation)研究中,物理領域使用某種函數 f 制定最小化問題, f 的值表示建模系統的能量,目標是將 f 的能量最小化。在機器學習中,始終需要使用成本函數來持續評估資料模型的效能,其中最小值意味著一組可能具有最佳(最低)誤差的最佳參數。函數 f 稱為目標函數、損失函數或成本函數(最小化)、效能函數或適應度函數(最大化),或在某些領域稱為能量函數最小化(或最大化)目標函數的可行解稱為最優解。[啟發式方法](#)(A heuristic [1])指任何採用某種實際方法(A practical method)來解決問題或自我發現,但不能保證是最優的、完美的或理性的,但仍足以在搜尋空間中達到即時的或短期的目標或近似值。當找出最佳解(An optimal solution, 最優解)成為不可能或不切實際時,可以使用啟發式方法(Heuristic methods)來加快找到滿意解(A satisfactory solution, 即品質可接受的解)。啟發法是從先前解決類似問題的經驗中得出的策略,這些策略依賴於使用易於獲取但鬆散適用的資訊來解決問題。啟發式方法一般能夠在一個合理的計算成本內找到一個接近理論最佳解的解,但不能夠保證其解的可行性與最佳性。啟發式方法通常是使用試誤法、經驗法則或有根據的猜測,以在龐大的解空間中搜尋最佳解。[元啟發式方法](#)(A metaheuristic [2],或稱萬用啟發式演算法)比一般啟發式方法在搜尋能力上更為高階。元啟發式方法旨在尋找、生成、調整或選擇可以為機器學習問題(或最佳化問題)提供滿意解,特別是在資訊不完整或不完善或計算能力有限的情況下。元啟發法對解空間進行採樣,否則該解空間太大而無法完全枚舉或探索。[元啟發法可能對正在解決的最佳化問題做出相對較少的假設,因此可用於各種問題](#)。

[超啟發式方法](#)(A hyper-heuristic)是一種啟發式搜尋方法,通常透過結合機器學習方法,尋求自動化選擇、組合、生成或調整幾種更簡單的啟發式(或此類啟發式的組件)的過程,以有效地解決搜尋問題。研究超啟發式方法的動機之一是建構能夠處理各類問題的系統,而不僅僅是解決一個問題。人們可以選擇多種啟發法來解決問題,而每種啟發法都有自己的優點和缺點。這個想法是透過結合已知啟發式的優點並補償其弱點來自動設計演算法。在典型的超啟發式框架中,有一個高階方法論和一組低階啟發式方法(建設性或擾動啟發式)。超啟發法可以被視為"現成的"方法("off-the-peg" methods),而不是"客製化"的元啟發法("made-to-measure" metaheuristics)。它們的目標是成為通用方法,基於一組易於實現的低階啟發法,以產生滿意解。在電腦科學、人工智慧和運籌學(Operational research)中,一些研究者已認識到:需要開發自動化系統來取代人類專家的角色。自動化啟發式設計的主要想法之一需要將機器學習機制納入演算法中以自適應地指導最佳解的搜尋。基於以上論述,元啟發式方法與超啟發式方法在本質上是基於啟發式方法(Heuristic methods)。著名的元啟發式方法有:粒子群演算法(Particle swarm optimization, PSO),基因演算法(Genetic Algorithm, GA),差分進化演算法(Differential evolution, DE),鯨群演算法(Whale optimization algorithm, WOA), ..., 等等。(Wiki)

[1] "Heuristic"一詞源自古希臘文"εύπίσκω"(讀音:heúrís̄kō),意為"to find, discover"(即找出或發現之意)。

[2] "Meta"源自拉丁文或希臘文,指"涉其自身或自身類型的事物"或"涉及或提供有關其所屬類別的成員的信息",亦有"超越"、"在上"、"轉化"、"當中, 與, 之後"之意。"Meta"的第一次已知使用是在1988年。

Random Optimization (RO)

(隨機最佳化演算法)

Random Optimization

- 1) 隨機最佳化法(Random optimization, RO)是一系列的數值最佳化方法(A family of numerical optimization methods)。RO不需要成本函數的梯度(The gradient),因此RO可以用於不連續或不可微的函數。此類優化方法也稱為直接搜尋法、無導數法或黑盒法(Direct-search, derivative-free, or black-box methods)。“Random optimization”一詞歸功於Matyas[1],他早期提出了RO以及基本數學分析。RO的工作原理是疊代地移動到搜尋空間中更好的解(位置),這些解是使用當前位置周圍的常態分佈(A normal distribution surrounding the current position)。
- 2) RO以看似隨機的方式順序探索成本函數的參數空間,以找到最小化(或最大化)成本函數的最佳點。除了無導數之外,RO方法最顯著的優點在於其簡單性,這使得該方法易於理解並方便地針對特定應用進行客製化。此外,已證明此方法在緊集[2]上(on a compact set)以機率100%收斂於全域最優[3][4]。
- 3) 此RO演算法對應於恆定步長的(1+1)-ES演化策略(A (1+1) evolution strategy with constant step-size)。[5][6]

- [1] Matyas, J. (1965). "Random optimization". Automation and Remote Control. 26 (2): 246–253.
- [2] 在數學中,特別是在一般拓樸中,緊緻性(Compactness)是一種旨在概括歐幾里德空間(Euclidean space)的封閉有界子集概念的屬性。這個想法是,緊緻空間(A compact space)沒有"穿孔(punctures)"或"遺失的端點(missing endpoints)",即它包括點的所有極限值。例如,開區間(0,1)不會是緊湊的,因為它排除了0和1的限制值,而閉區間[0,1]則將是緊緻的。緊緻性由雷內·莫里斯·弗雷歇(René Maurice Fréchet, 1878–1973, 法國數學家,他對一般拓樸學做出了重大貢獻,並且是第一個定義度量空間的人)於1906年正式提出,將Bolzano-Weierstrass定理從幾何點空間推廣到函數空間。在最初引入之後,在一般度量空間中發展了各種等效的緊緻性概念,包括順序緊緻性和極限點緊緻性。術語"compact set(緊集)"有時用作緊空間的同義詞,但也經常指拓撲空間(A topological space)的緊子空間。(Wiki)
- [3] Baba, N. (1981). "Convergence of a random optimization method for constrained optimization problems". Journal of Optimization Theory and Applications. 33 (4): 451–461.
- [4] Solis, F.J.; Wets, R.J-B. (1981). "Minimization by random search techniques". Mathematics of Operations Research. 6 (1): 19–30.
- [5] 演化策略(Evolution strategy, ES)中,令 μ =群體規模(The population size);令 λ =子群體的個體數目。ES有兩種變體方法(Two variants),可以最佳選擇出下一代的父代群體,亦即,(A)在(μ, λ)-ES策略中,僅使用最好的 μ 個子代個體;(B)在($\mu + \lambda$)-ES精英式策略中,最好的精英是從父代的 μ 個個體和子代的 λ 個個體中選出 μ 個個體。Bäck和Schwefel建議 λ 的值應為群體規模 μ 的七倍[*],其中 μ 不得設定太小,因為選擇壓力會增大(because of the strong selection pressure)。 μ 的合適值取決於應用,必須透過實驗確定。(Wiki)
- [*] Bäck, Thomas; Schwefel, Hans-Paul (1993). "An Overview of Evolutionary Algorithms for Parameter Optimization". Evolutionary Computation. 1 (1): 1–23.
- [6] 最簡單的演化策略(ES)適用於總體數目(即 $\mu + \lambda$ 的個體數目)為二個的情況,即當前位置(父代)及其突變體(its mutant)(子代),稱為(1+1)-ES。只有突變體(The mutant)的適應度(Fitness)至少與父代一樣好,它才能成為下一代的父代;否則突變體將被忽略。更廣義地,我們可以產生 λ 個突變體並與父代競爭,稱為(1 + λ)-ES。而在(1, λ)-ES中,最好的突變體成為下一代的父代,而目前的父代始終被忽略。(Wiki)

Random Optimization

基本的隨機最佳化演算法(Primitive RO) :: 步驟

設 $f: R^n \rightarrow R$ 為必須最小化的損失函數(The loss function, 或稱成本函數)。讓 $\mathbf{x} \in R^n$ 指定 n 維搜尋空間中的位置向量(或候選解)。原始版本的 RO 演算法[1]可以描述為：透過以下四個步驟的疊代的(1+1)-ES演化策略，來找到最優解。

Step 1: 在搜尋空間(參數空間)中，隨機選取初始位置 $\mathbf{x} \in R^n$ 。評估損失函數 $f(\mathbf{x})$ 。

Step 2: 取得隨機向量 \mathbf{dx} 。新增 \mathbf{dx} 至目前位置 \mathbf{x} ，以生成新位置 $\mathbf{x} + \mathbf{dx}$ 。評估損失函數 $f(\mathbf{x} + \mathbf{dx})$ 。

- 隨機向量 \mathbf{dx} 通常是藉由 n 綴常態分佈, $N(\mu, \sigma^2)$, 取樣而得($\mu = \mathbf{0}$), 即 $\mathbf{dx} = N(\mathbf{0}, \sigma^2)$, 其中 \mathbf{dx} 、 μ 、 $\mathbf{0}$ 及 σ 皆是 n 綴向量(n -dimensional vectors)。
- 一維常態分佈：參式1及圖1。在 x 軸某對應區間範圍，常態分佈曲線下的面積代表事件發生於該區間範圍內之機率。參圖2。在一維常態分佈中，一個標準差之範圍內所佔面積比率約為全部面積之68%；兩個標準差之內所佔面積比率約為95%；三個標準差之內所佔面積比率約為99.7%。

Step 3: 如果 $f(\mathbf{x} + \mathbf{dx}) < f(\mathbf{x})$, 則將更新目前位置 $\mathbf{x} = \mathbf{x} + \mathbf{dx}$ 。

Step 4: 檢查停止條件。如果停止條件滿足，則停止演算法。否則，返回**Step 2** 尋找更佳位置。

◆此基本的RO是一種真正隨機的方法，因為搜尋方向純粹由隨機數產生器引導。

[1] Matyas, J. (1965). "Random optimization". Automation and Remote Control. 26 (2): 246–253.

$$f(x) = N(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad (1)$$

where

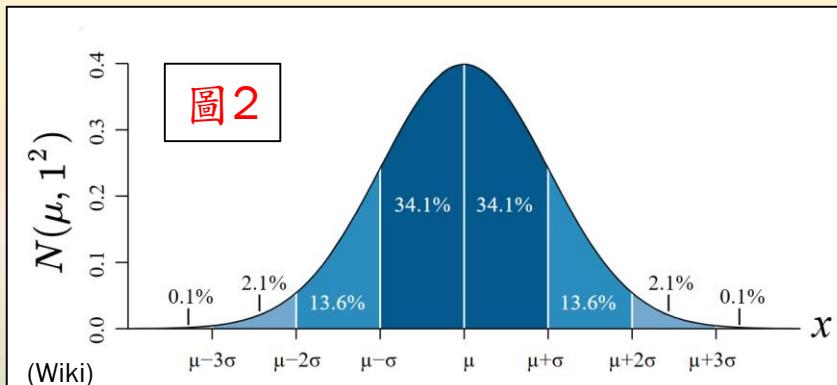
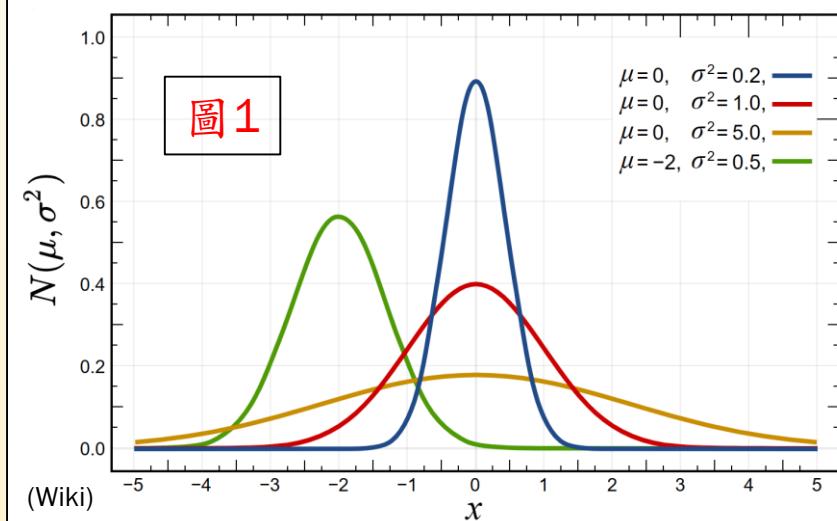
μ

σ

σ^2

$N(\mu, \sigma^2)$

the mean or expectation of the normal distribution,
the standard deviation of the distribution ,
the variance of the distribution ,
the notation referred to as the normal distribution.



Random Optimization

改良式隨機最佳化演算法(Improved RO) :: 步驟

有幾種方法可以改進這個RO原始版本(by Matyas 1965),這些是基於以下觀察,如式1所示。基於式1,第一個觀察導致在原始RO方法中進行一個反向步驟(A reverse step)。第二個觀察激發了將偏置項用作隨機向量的中心(A bias term as the center for the random vector)。在包含這兩個指導方針之後,改良後的RO方法[1]涉及六個步驟的疊代的(1+1)-ES演化策略。流程圖見圖1。

Step 1: 選擇初始點 x 作為目前位置。將初始偏置項 b 設定為等於零向量。在目前位置處評估損失函數 $f(x)$ 。

Step 2: 取得隨機向量 dx 。將偏置項 b 和隨機向量 dx 加到目前位置 x ,以生成新位置 $x + b + dx$ 。在新位置處評估損失函數,即 $f(x + b + dx)$ 。

Step 3: 若 $f(x + b + dx) < f(x)$,則更新 $x = x + b + dx$ 及 $b = 0.2b + 0.4dx$; 轉至**Step 6**。否則,轉至**Step 4**。

Step 4: 若 $f(x + b - dx) < f(x)$,則更新 $x = x + b - dx$ 及 $b = b - 0.4dx$; 轉至**Step 6**。否則,轉至**Step 5**。

Step 5: 將偏置項設為 $b = 0.5b$,並轉到**Step 6**。

Step 6: 檢查停止條件。如果停止條件滿足,則停止演算法。否則,返回**Step 2**尋找更佳位置。

◆此改良式RO可加入跳脫機制(Jumps)或內插位置(Interpolated points),以增加找到較優解之機率[2]。它也可使用 $(\mu + \lambda)$ -ES演化策略來形成新型態RO演算法。

觀察1:如果在某個方向的搜索結果導致更高的損失函數,相反的方向通常會導致更低的損失函數。

觀察2:連續成功的搜索在某個方向應該使後續的搜索偏向於這個方向。另一方面,連續在某個方向上失敗應該阻礙後續在這個方向的搜索。

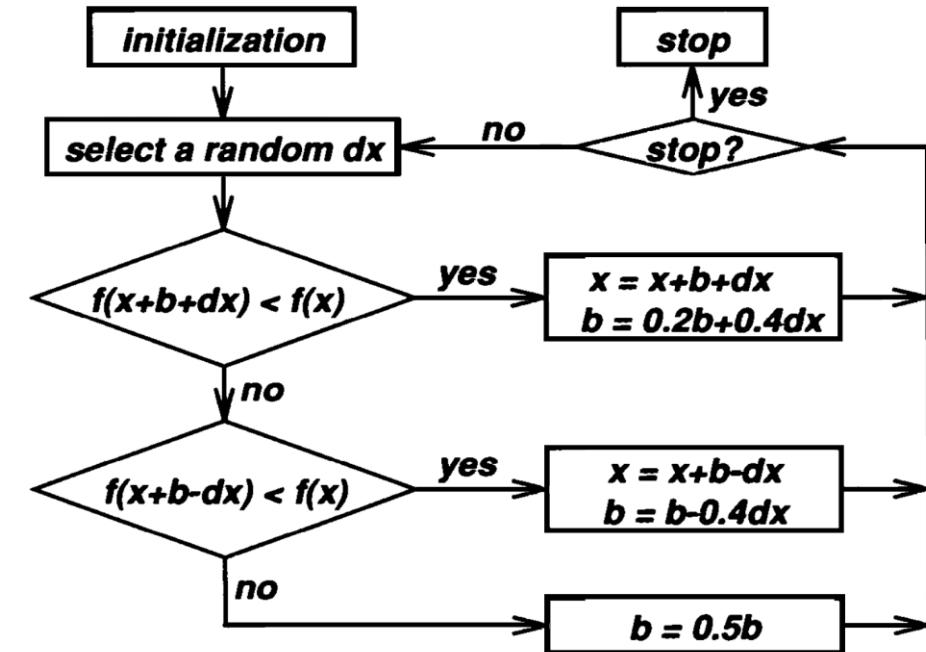


圖1 Flow chart for the improved RO.

[1] Solis, F.J.; Wets, R.J-B. (1981). "Minimization by random search techniques". Mathematics of Operations Research. 6 (1): 19–30.

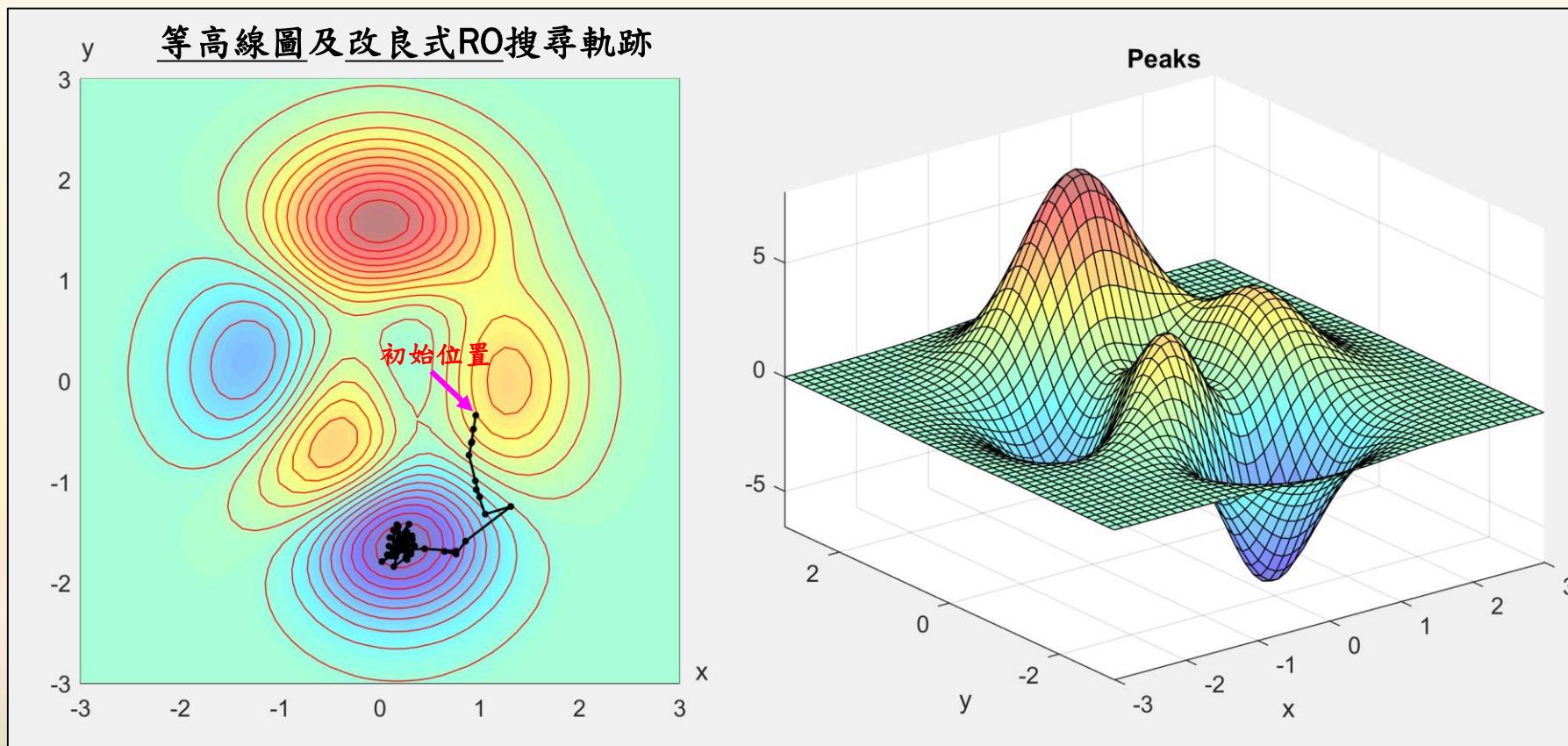
[2] Chunshien Li, Roland Priemer and Kuo-Hsiang Cheng: "Optimization by random search with jumps," International Journal for Numerical Methods in Engineering, vol.60, pp.1301–1315, June 2004.

Random Optimization

Example: 改良式RO演算法 ((1+1)-ES演化策略)

改良式RO演算法從某個起點,以疊代方式找到「peaks」函數(定義:參見式1)的最小點。

```
z = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...
- 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
- 1/3*exp(-(x+1).^2 - y.^2) (1)
```



Particle Swarm Optimization (PSO)

(粒子群最佳化演算法)

Particle Swarm Optimization

- 1) 粒子群最佳化演算法(Particle Swarm Optimization, PSO) [1]是由Kennedy和Eberhart (1995)提出的一種基於隨機群體的最佳化方法。它已成功應用於人工神經網路訓練、函數最佳化和模式分類(Pattern classification)等許多問題。PSO最初旨在模擬鳥群或魚群社會行為(Social behavior)的生物體行動風格的表示(A stylized representation)。學者 Poli (2007 & 2008)對PSO應用進行了廣泛的評述[2][3]。Bonyadi和Michalewicz (2017)發表了PSO理論和實驗工作的綜述[4]。
- 2) PSO是一種元啟發式演算法(A metaheuristic),因為它很少或根本不對正在最佳化的問題做出任何假設,並且可以搜尋非常大的候選解空間。此外,PSO不使用最佳化問題的成本函數之梯度,這意味著PSO不需要像梯度下降法或牛頓法等經典最佳化方法那樣要求最佳化問題可微(differentiable)。然而,如同其他元啟發法一樣,PSO並不能保證找到最佳解。
- 3) PSO是一種透過疊代過程來改進最佳化問題的候選解(A candidate solution)的方法。它藉由擁有一組候選解(此處稱為粒子)並根據粒子位置和速度的簡單數學公式,在搜尋空間中移動這些粒子來搜尋最佳解。此組粒子稱為粒子群(A swarm of particles)。每個粒子的運動都受到該粒子之歷史最佳位置的影響,但也會被引導至粒子群之歷史最佳位置,這些位置會隨著其他粒子找到更好的位置而更新。此PSO最佳化機制將使群體走向最佳解。

- [1] Kennedy, J.; Eberhart, R. (1995). "Particle Swarm Optimization". Proceedings of IEEE International Conference on Neural Networks. Vol. IV. pp. 1942–1948.
- [2] Poli, R. (2007). "An analysis of publications on particle swarm optimisation applications" (PDF). Technical Report CSM-469.
- [3] Poli, R. (2008). "Analysis of the publications on the applications of particle swarm optimisation" (PDF). Journal of Artificial Evolution and Applications. 2008: 1–10.
- [4] Bonyadi, M. R.; Michalewicz, Z. (2017). "Particle swarm optimization for single objective continuous space problems: a review". Evolutionary Computation. 25 (1): 1–54.

Particle Swarm Optimization

- 1) PSO演算法是一種基於群體的優化方法(A swarm-based optimization method),其機制是鳥群(Bird flocking)或魚群(Fish schooling)的食物搜索行為。鳥群中的每隻鳥都被視為一個粒子(A particle)。一個群(A swarm)由許多粒子組成假設食物的位置被視為PSO搜尋空間中某種效能指標函數的最佳解。每一個粒子位置(The location of each particle)被視為該函數的一個候選解(A candidate solution)。
- 2) 群中的所有粒子都有它們的位置(Locations)和速度(Velocities)。每個粒子位置都可以對應到具有某種給定效能指標函數(Performance index function,或稱成本函數 cost function,或稱適應函數 fitness function)的一個效能指標值(或稱成本值、適應值)。PSO群的所有粒子於疊代中相互競爭成為贏家。
- 3) 直到第 k 次疊代為止,歷次搜索中全群之最佳位置記為 $\mathbf{g}_{\text{best}}(k)$ 。直到第 k 次疊代為止,歷次搜索中粒子 i 之最佳位置記為 $\mathbf{p}_{\text{best},i}(k)$ 。
- 4) PSO使用 $\mathbf{g}_{\text{best}}(k)$ 及 $\mathbf{p}_{\text{best},i}(k)$ 的信息,更新群中粒子的速度和位置。假設PSO的搜索空間是 D 維的。基本的PSO演算法如式1及式2所示,其中,速度和位置向量的更新公式可以細化到個別維度的層次。

$$\mathbf{v}_i(k+1) = \boldsymbol{\omega} \cdot \mathbf{v}_i(k) + \mathbf{C}_p \cdot \xi_p \cdot (\mathbf{p}_{\text{best},i}(k) - \mathbf{x}_i(k)) + \mathbf{C}_g \cdot \xi_g \cdot (\mathbf{g}_{\text{best}}(k) - \mathbf{x}_i(k)) \quad (1)$$

$$\mathbf{x}_i(k+1) = \mathbf{x}_i(k) + \mathbf{v}_i(k+1) \quad (2)$$

for $i = 1, 2, \dots, n_{\text{swarm}}$, where

k the k th iteration;

$\mathbf{v}_i = [v_{i,1}, v_{i,2}, \dots, v_{i,D}]^T$ the velocity vector of particle i ;

$\mathbf{x}_i = [x_{i,1}, x_{i,2}, \dots, x_{i,D}]^T$ the location vector of particle i ;

D the dimension of the space searched by the PSO;

n_{swarm} the swarm size of particles;

$\boldsymbol{\omega} = [\omega_1, \omega_2, \dots, \omega_D]^T$ the vector of inertia factors, where usually $0.8 \leq \omega_d < 1$, $d = 1, 2, \dots, D$;

$\mathbf{C}_p = [c_{p,1}, c_{p,2}, \dots, c_{p,D}]^T$ the vector of **cognitive coefficients**,

$\mathbf{C}_g = [c_{g,1}, c_{g,2}, \dots, c_{g,D}]^T$ the vector of **social coefficients**,
where $1 \leq c_{p,d}, c_{g,d} \leq 3$, $d = 1, 2, \dots, D$;
usually, $c_{p,d}$ and $c_{g,d}$ are set to 2;

$\xi_p = [\xi_{p,1}, \xi_{p,2}, \dots, \xi_{p,D}]^T$ a random vector, where $\xi_{p,d}$ is uniformly random numbers in $[0,1]$, $d = 1, 2, \dots, D$;

$\xi_g = [\xi_{g,1}, \xi_{g,2}, \dots, \xi_{g,D}]^T$ a random vector, where $\xi_{g,d}$ is uniformly random numbers in $[0,1]$, $d = 1, 2, \dots, D$;

$\mathbf{p}_{\text{best},i}(k)$ the historical best location of particle i at iteration k ;

$\mathbf{g}_{\text{best}}(k)$ the historical best location of the swarm at iteration k ;

\cdot the symbol that performs an element-by-element multiplication.

Particle Swarm Optimization

在p.11式1中, PSO 結合了個體搜尋和群體搜尋的行為。

- 1) 所有粒子都互相競爭,成為群體的最好粒子(The swarm best)。在搜尋過程中存在著微妙的競爭與合作關係。所有粒子在搜尋過程中都會記住自己的最佳位置。此外,所有粒子都有自己的(慣性)速度 $v_i(k)$,以確定它們的搜尋運動。粒子利用損失函數的概念疊代地搜尋最佳解。在疊代 k 結束前,粒子透過兩個搜尋記憶來改變以其搜尋方向 $v_i(k + 1)$ 於疊代 $(k + 1)$,這兩個搜尋記憶是 $p_{best,i}(k)$ [即疊代 k 時,個體粒子的最佳位置] 和 $g_{best}(k)$ [即疊代 k 時,群體的最佳位置]。
- 2) PSO粒子的向量圖示於圖1。粒子*i*之速度(Velocity)受到3個因素的影響: (a)它自身的慣性速度 $v_i(k)$ [物理性質]; (b)粒子*i*自身的位置趨向於其自身歷史最佳位置 $p_{best,i}$ [自我認知性質]; (c)粒子*i*自身的位置趨向於群歷史最佳位置 g_{best} [群體社會認知性質]。
- 3) 圖2概念性地顯示了PSO群在搜尋過程中的位置演化,其中目標位置視食物位置(即最佳解位置)。為簡單起見,群體中只有兩個粒子,分別標示為A和B。兩個粒子在疊代 k 時更新其位置。當 $k = 2$ 時,顯然粒子B是距離目標位置最近的粒子。這意味著粒子B是 $k = 2$ 時群體中最好的粒子。

圖1

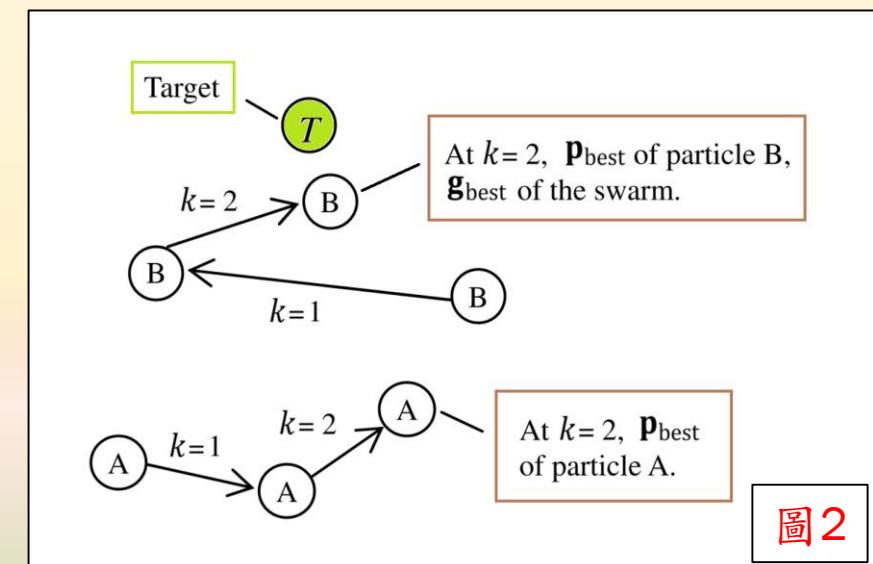
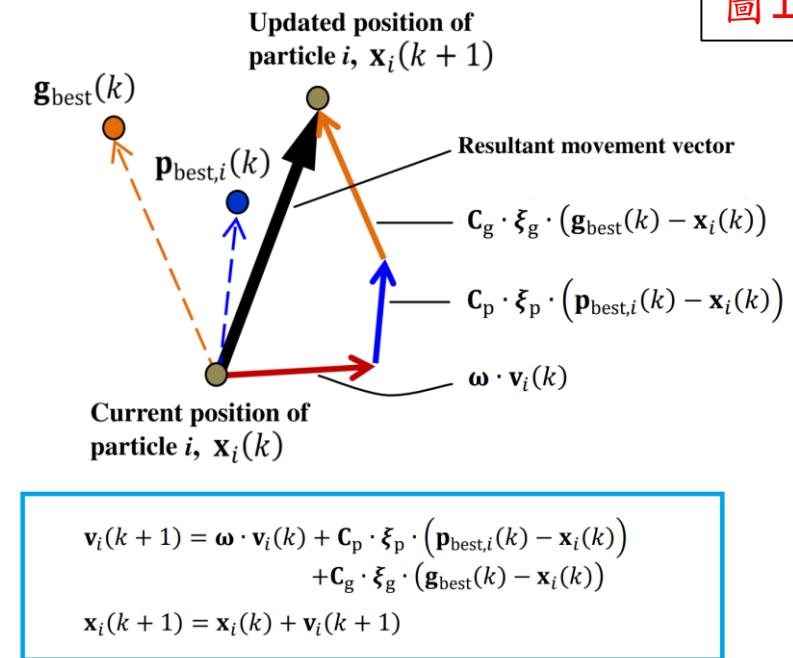


圖2

Particle Swarm Optimization

PSO最佳化演算法:: 步驟

假設PSO的搜索空間是 D 維的。設 $f: R^D \rightarrow R$ 是必須最佳化的損失函數(The loss function,或稱成本函數)。基本PSO優化演算法的步驟如下。

Step 1: 初始化位置向量 \mathbf{g}_{best} 。

執行迴圈 $i = 1, 2, \dots, n_{\text{swarm}}$ [初始化之粒子迴圈]

使用均勻分佈的隨機向量初始化粒子位置向量(即候選解) $\mathbf{x}_i \in R^D$, 其中, 位置分量 $x_{i,j}$ 於維度 j 設定搜索空間範圍之內。

將粒子 i 的 $\mathbf{p}_{\text{best},i}$ 位置向量初始化為其初始位置向量 \mathbf{x}_i 。

若 $f(\mathbf{p}_{\text{best},i}) < f(\mathbf{g}_{\text{best}})$, 則 $\mathbf{g}_{\text{best}} \leftarrow \mathbf{p}_{\text{best},i}$ 。

初始化粒子 i 的速度向量 $\mathbf{v}_i(k)$, 其中, 速度分量 $v_{i,d}$ 於維度 j 設定速度範圍之內, $d = 1, 2, \dots, D$ 。

直到初始化之粒子迴圈執行完畢。

Step 2: 執行迴圈 $i = 1, 2, \dots, n_{\text{swarm}}$ [粒子迴圈]

執行迴圈 $d = 1, 2, \dots, D$ [搜索空間維度迴圈]

選擇維度 d 之隨機亂數 $\xi_{p,d}$ 及 $\xi_{g,d}$ 於均勻分佈區間 $[0,1]$;

更新粒子 i 之維度 d 的速度。[參見: p.11式1之 $v_{i,d}$ 速度更新]

直到搜索空間維度迴圈執行完畢。[得到粒子 i 之速度向量 \mathbf{v}_i 更新]

更新粒子 i 之位置向量 \mathbf{x}_i 。[參見: p.11式2粒子位置向量 \mathbf{x}_i 更新]

若 $f(\mathbf{x}_i) < f(\mathbf{p}_{\text{best},i})$, 則 $\mathbf{p}_{\text{best},i} \leftarrow \mathbf{x}_i$ 。

若 $f(\mathbf{p}_{\text{best},i}) < f(\mathbf{g}_{\text{best}})$, 則 $\mathbf{g}_{\text{best}} \leftarrow \mathbf{p}_{\text{best},i}$ 。

直到粒子迴圈執行完畢。[得到所有粒子 \mathbf{x}_i 、 $\mathbf{p}_{\text{best},i}$ 、及 \mathbf{g}_{best} 之位置向量更新]

Step 3: 檢查停止條件。如果停止條件滿足, 則停止該PSO演算法。否則, 返回 Step 2 尋找更佳 \mathbf{g}_{best} 位置向量。

Particle Swarm Optimization

Comments on PSO

- 1) ω_d (慣性權重參數, inertia factor)、 $\xi_{p,d}$ (自我認知參數, cognitive coefficient)和 $\xi_{g,d}$ (群體認知參數, social coefficient), $d = 1, 2, \dots, D$, 由研究者選擇,並控制PSO演算法的搜尋行為和效能。PSO參數的選擇會對最佳化效能產生很大影響。因此,選擇產生良好性能的PSO參數已成為許多研究的主題。為了防止發散(即搜尋爆炸),慣性權重 ω_d 必須小於1;然後可以透過收縮方法或自由選擇來導出其他兩個參數[1],但分析表明收斂區間可用以約束 $\xi_{p,d}$ 及 $\xi_{g,d}$,其值位於[1,3]區間中。
- 2) PSO參數還可以透過使用另一個覆蓋優化器(An overlaying optimizer,一種元優化meta-optimization的概念)進行調整[2][3],或甚至在優化過程中進行微調,例如透過模糊邏輯[4][5]。PSO有時需針對各種最佳化場景調整了參數[6][7]。
- 3) PSO群的拓樸(The topology of a PSO swarm)定義了每個粒子可以與之交換資訊的粒子子集[8]。演算法的基本版本使用全域拓樸作為群體通訊結構。這種拓樸允許所有粒子與所有其他粒子資訊交流,因此整個群體共享單個粒子的相同最佳位置 g_{best} 。然而,這種方法可能會導致群體陷入局部最小值,因此,不同的拓樸被用來控制粒子之間的資訊流。例如,在局部拓樸中,粒子(particles)僅在粒子子集(A subset of particles)間共享資訊[9]。這個子集可以是幾何子集,例如,"距離最近的m個粒子",或更常見的是"社會子集"(即一組不依賴粒子間任何距離)。在這種情況下,PSO這種變體方式被稱為局部群最佳(相對於基本PSO的全群最佳)。常用的群體拓樸是環(A ring),其中每個粒子只有兩個鄰居。但是,還有許多其他拓樸[9]。拓樸不一定是靜態的(not necessarily static),事實上,由於拓樸與粒子通訊的多樣性(the diversity of communication of the particles)有關[10],有一些學者已經作出努力來創建自適應拓樸(Adaptive topologies),例如,SPSO, APSO, TRIBES, Cyber Swarm, 和C-PSO。

- [1] Clerc, M.; Kennedy, J. (2002). "The particle swarm - explosion, stability, and convergence in a multidimensional complex space". IEEE Transactions on Evolutionary Computation. 6 (1): 58–73.
- [2] Pedersen, M.E.H.; Chipperfield, A.J. (2010). "Simplifying particle swarm optimization". Applied Soft Computing. 10 (2): 618–628.
- [3] Mason, Karl; Duggan, Jim; Howley, Enda (2018). "A Meta Optimisation Analysis of Particle Swarm Optimisation Velocity Update Equations for Watershed Management Learning". Applied Soft Computing. 62: 148–161.
- [4] Nobile, M.S.; Cazzaniga, P.; Besozzi, D.; Colombo, R.; Mauri, G.; Pasi, G. (2018). "Fuzzy Self-Tuning PSO: a settings-free algorithm for global optimization". Swarm and Evolutionary Computation. 39: 70–85.
- [5] Nobile, M.S.; Pasi, G.; Cazzaniga, P.; Besozzi, D.; Colombo, R.; Mauri, G. (2015). "Proactive particles in swarm optimization: a self-tuning algorithm based on fuzzy logic". Proceedings of the 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2015), Istanbul (Turkey). pp. 1–8.
- [6] Pedersen, M.E.H. (2010). "Good parameters for particle swarm optimization". Technical Report HL1001.
- [7] Cazzaniga, P.; Nobile, M.S.; Besozzi, D. (2015). "The impact of particles initialization in PSO: parameter estimation as a case in point, (Canada)". Proceedings of IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology.
- [8] Kennedy, J.; Mendes, R. (2002). "Population structure and particle swarm performance". Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600). Vol. 2, pp. 1671–1676.
- [9] Bratton, Daniel; Kennedy, James (2007). "Defining a Standard for Particle Swarm Optimization". 2007 IEEE Swarm Intelligence Symposium (PDF). pp. 120–127.
- [10] Oliveira, M.; Pinheiro, D.; Andrade, B.; Bastos-Filho, C.; Menezes, R. (2016). "Communication Diversity in Particle Swarm Optimizers". Swarm Intelligence. Lecture Notes in Computer Science. Vol. 9882. pp. 77–88.

Whale Optimization Algorithm (WOA)

(鯨群最佳化演算法)

Whale Optimization Algorithm

學者S. Mirjalili及A. Lewis於2016年共同提出鯨群最佳化演算法(Whale optimization algorithm, WOA) [1]。WOA是一種受自然啟發的元啟發式最佳化演算法(A metaheuristic),它模仿座頭鯨的社會行為。WOA的靈感來自於座頭鯨的氣泡網狩獵機制(The bubble-net hunting mechanism by humpback whales) [2]。座頭鯨喜歡捕食靠近水面的磷蝦或小魚群。據觀察,這種覓食是透過沿著圓形或「9」形路徑產生獨特的氣泡來完成的。2011年之前,這種行為僅基於表面觀察進行研究。然而,Goldbogen等人利用標籤感測器(Tag sensors)研究了這種行為。他們獲得9頭座頭鯨的300次基於標籤的氣泡網進食事件。他們發現了兩種與氣泡相關的動作[3],並將它們命名為“向上螺旋”(Upward-spirals)和“雙環”(Double-loops)。在前者動作中,座頭鯨潛入水下約12米,然後開始在獵物周圍產生螺旋狀氣泡,然後向上游向水面。後者的動作包括三個不同的階段:coral loop, lobtail, 及 capture loop。

註:座頭鯨(Humpback whale; 學名: *Megaptera novaeangliae*)是鬚鯨(Baleen whale)的一種,也是大翅目(genus *Megaptera*)中唯一的物種。“Humpback whale”意為駝背鯨,又稱巨臂鯨、大翅鯨。成年鯨身長為14–17m,體重可達40公噸。座頭鯨以破水和其他獨特的表面行為而聞名。雄性會發出複雜的歌曲,通常持續4~33分鐘。座頭鯨遍布世界各地的海洋,通常每年遷徙長達16,000公里。它們在極地水域進食,並洄游到熱帶或亞熱帶水域繁殖和生育。它們的食物主要是磷蝦和小魚,它們利用氣泡來捕捉獵物。虎鯨是座頭鯨的主要天敵。(Wiki)



一群 15 頭鯨魚用氣泡網捕魚(near Juneau, Alaska)。



座頭鯨在氣泡網螺旋的中心衝撞。

[1] Seyedali Mirjalili and Andrew Lewis (2016). "The whale optimization algorithm". *Advances in Engineering Software*, 95: 51-67.

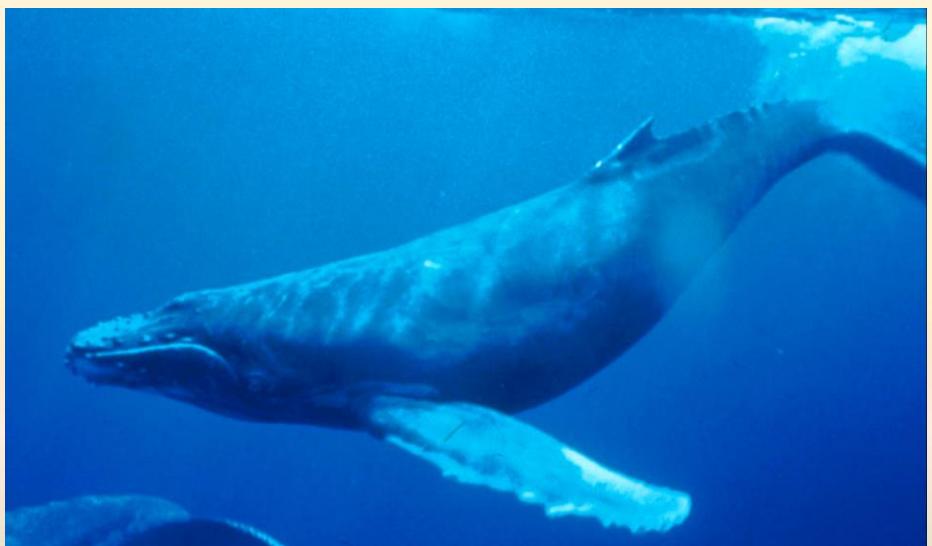
[2] 氣泡網進食(Bubble-net feeding)是座頭鯨([humpback whales](#))和布氏鯨([Bryde's whales](#))的一種進食行為。這是座頭鯨已知的少數幾種表面進食行為之一。這種類型的進食可以單獨進行,也可以與多達二十頭鯨魚一起同時進行。鯨魚也可以進行類似表面進食的方法,稱為弓箭步進食(Lunge feeding, 或稱突然前衝式進食)。(Wiki)

[3] J. A. Goldbogen, A. S. Friedlaender, J. Calambokidis, M. F. McKenna, M. Simon, and D. P. Nowacek, "Integrative Approaches to the Study of Baleen Whale Diving Behavior, Feeding Performance, and Foraging Ecology," *BioScience*, vol. 63, pp. 90-100, 2013.

Whale Optimization Algorithm

鯨魚大腦的某些區域有類似人類的常見細胞，稱為紡錘狀細胞(Spindle cells)，中間寬兩端窄的紡錘狀細胞結構，又稱梭形細胞。這些細胞負責人類的判斷、情緒和社會行為。換句話說，紡錘狀細胞使我們有別於其他生物。鯨魚的這些細胞數量是成年人的兩倍，這也是鯨魚它們聰明的主要原因。事實證明，鯨魚可以像人類一樣思考、學習、判斷、交流，甚至變得有情感，但顯然其聰明程度相較於人類要低得多。據觀察，鯨魚(主要是虎鯨)也能夠發展出自己的語言。[1]

[1] P. R. Hof and E. Van Der Gucht, "Structure of the cerebral cortex of the humpback whale, *Megaptera novaeangliae* (Cetacea, Mysticeti, Balaenopteridae)," *The Anatomical Record*, vol. 290, pp. 1-31, 2007.



座頭鯨
Humpback whale



<http://takenoie.jugem.jp/?eid=1320>

梭(shuttle): 織布時用來牽引橫線的器具，兩頭尖，中間粗，絲束放於中空部分。



尼泊爾的一名婦女正在紡羊毛(spinning wool)。(Wiki)



與拔杼(A distaff)一起使用的抓握紡錘軸(A grasped spindle)。(Wiki)

Whale Optimization Algorithm

我們討論WOA演算法的次序如下：(1) **包圍獵物**(Encircling prey), (2) **螺旋氣泡網進食(攻擊)策略**(Spiral bubble-net feeding/attacking maneuver)、和 (3) **搜索獵物**(Search for prey)等數學模型。氣泡網進食是一種獨特的行為，且可以在座頭鯨身上觀察到。在此WOA中，對螺旋氣泡網進食進行了數學建模，以進行最佳化。

包圍獵物 (Encircling Prey)

首先我們討論**包圍獵物**(Encircling prey)，如下。

- 1) 假設WOA的搜索空間是 D 維的。在以下討論中，我們將一隻座頭鯨視為一顆粒子，其位置視為一個候選解。**座頭鯨可以辨識獵物的位置並包圍它們**。由於搜尋空間中的最佳位置事先未知，WOA假設目前最佳候選解 \vec{X}^* 是**目標獵物位置**(The position of a target prey)或**接近最佳解**(close to the optimum)。定義**最佳位置** \vec{X}^* 之後，其他鯨魚(粒子)將試著將其位置趨向最佳位置(獵物位置)而更新；這種行為由**式1**表示，其中，向量 \vec{A} 及 \vec{C} 分別由**式2及式3**表示。基於**式1**的 $\vec{X}_i(t+1)$ 表達式，**位置向量更新可以細化到該鯨魚位置的個別維度層次**。

$$\vec{X}_i(t+1) = \vec{X}^*(t) - \vec{A} \cdot |\vec{C} \cdot \vec{X}^*(t) - \vec{X}_i(t)| \quad (1)$$

for $i = 1, 2, \dots, n_s$,

where

t the current iteration,

n_s the swarm size of whales,

D the dimension of a search space,

$\vec{A} = [A_1, A_2, \dots, A_D]^T$ a coefficient vector,

$\vec{C} = [C_1, C_2, \dots, C_D]^T$ a coefficient vector,

$\vec{X}_i = [x_{i,1}, x_{i,2}, \dots, x_{i,D}]^T$ the position vector of whale i ,

\vec{X}^* the best position vector obtained so far. (獵物位置)

$||$ the absolute operation on the components of a vector,

\cdot an element-by-element multiplication,

$\vec{D}_i(t) = |\vec{C} \cdot \vec{X}^*(t) - \vec{X}_i(t)| = [d_{i,1}, d_{i,2}, \dots, d_{i,D}]^T$ the distance vector of whale i to a prey.

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} = [A_1, A_2, \dots, A_D]^T \quad (2)$$

$$\vec{C} = 2 \cdot \vec{r}_2 = [C_1, C_2, \dots, C_D]^T \quad (3)$$

where

$\vec{a} = [a_1, a_2, \dots, a_D]^T$ a vector whose components are linearly decreased from 2 to 0 over the course of iterations,

$\vec{r}_1 = [r_{1,1}, r_{1,2}, \dots, r_{1,D}]^T$ a random vector in $[0,1]^D$,

$\vec{r}_2 = [r_{2,1}, r_{2,2}, \dots, r_{2,D}]^T$ a random vector in $[0,1]^D$,

D the dimension of a search space,

\cdot an element-by-element multiplication,

$r_{1,d}$ a random number in $[0,1]$ for dimension d ,

$r_{2,d}$ a random number in $[0,1]$ for dimension d ,

A_d the d th component of the vector \vec{A} , $-a_d \leq A_d \leq a_d$,

C_d the d th component of the vector \vec{C} , $0 \leq C_d \leq 2$, $d=1, 2, \dots, D$.

Whale Optimization Algorithm

包圍獵物 (Encircling Prey) (承上一頁)

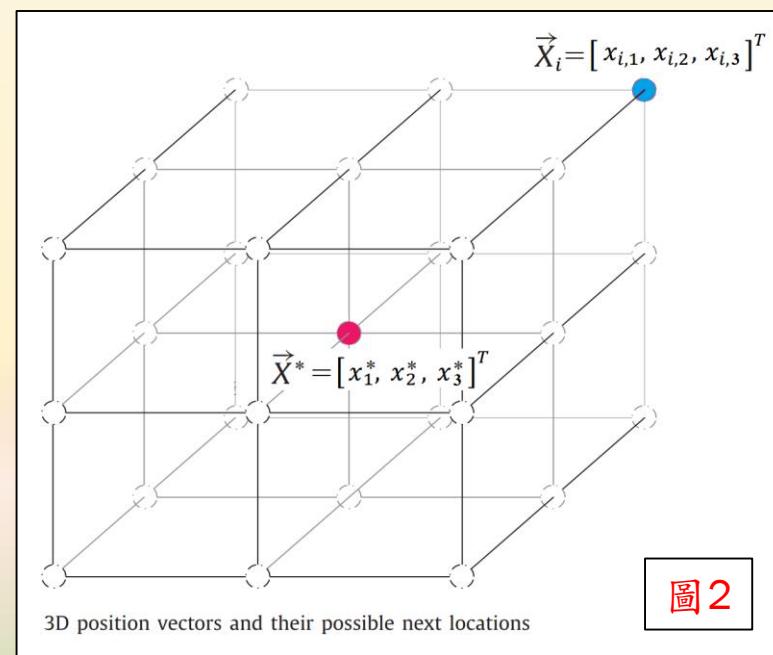
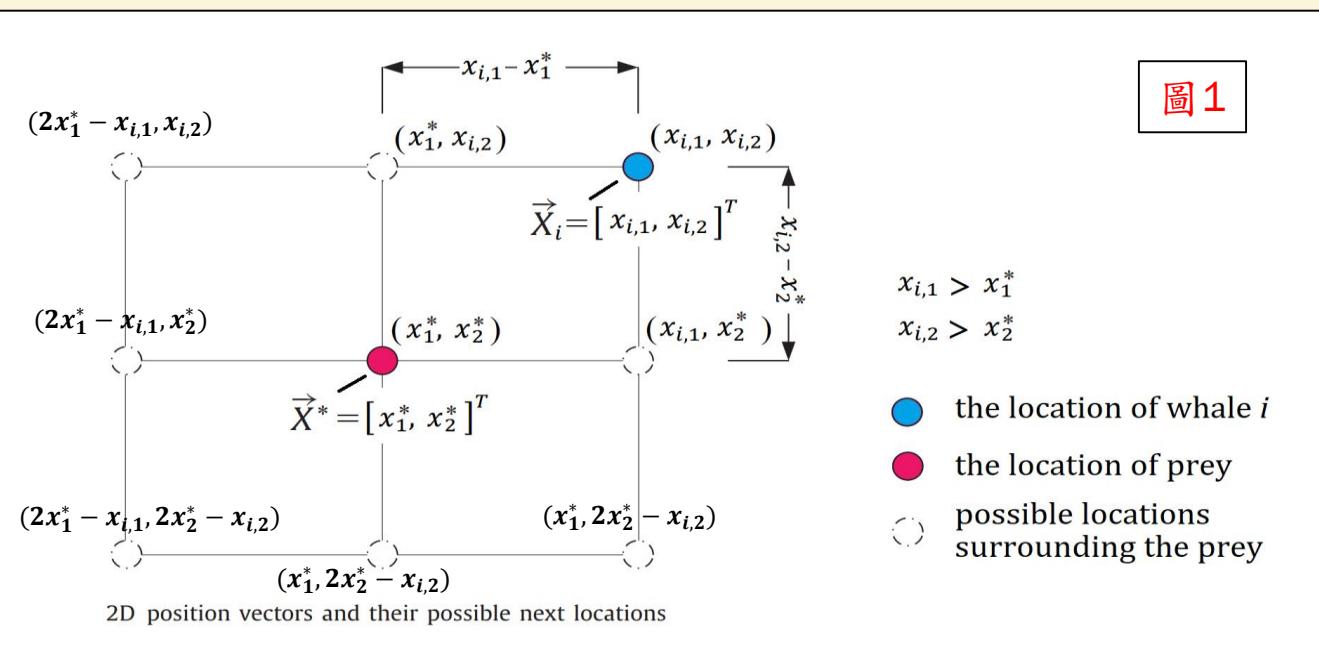
$$\vec{X}_i(t+1) = \vec{X}^*(t) - \vec{A} \cdot |\vec{C} \cdot \vec{X}^*(t) - \vec{X}_i(t)| \quad (\text{p.18式1})$$

2) 圖1說明了 p.18式1 的二維空間背後的基本原理。

鯨魚 i 之位置 $\vec{X}_i = [x_{i,1}, x_{i,2}]^T$ 可以根據目前最佳位置 $\vec{X}^* = [x_1^*, x_2^*]^T$ (即獵物位置) 進行更新。透過調整 p.18式2及式3的向量 $\vec{A} = [A_1, A_2]^T$ 和 $\vec{C} = [C_1, C_2]^T$ 的值, 相對於當前鯨魚 i 之位置 $\vec{X}_i(t)$, 得以實現在最佳位置 $\vec{X}^*(t)$ 週圍可能的不同位置。應該注意的是, 透過 p.18式2及式3中定義的隨機向量(即 \vec{r}_1 及 \vec{r}_2) 可以到達位於圖1所示關鍵位置(即 \vec{X}_i 及 \vec{X}^*) 之間

的任何位置。因此, p.18式1 允許任何鯨魚更新其位置於當前最佳解 $\vec{X}^*(t)$ 的鄰域之任何位置 (any position in the neighborhood of the current best solution), 如此, 就模擬了包圍獵物 (encircling prey)。

3) 圖2描述了三維空間中搜尋鯨魚可能的更新位置。相同的概念可以擴展到 D 維的搜尋空間, 鯨魚將在搜尋空間中圍繞最佳解 \vec{X}^* 的鄰域移動。



Whale Optimization Algorithm

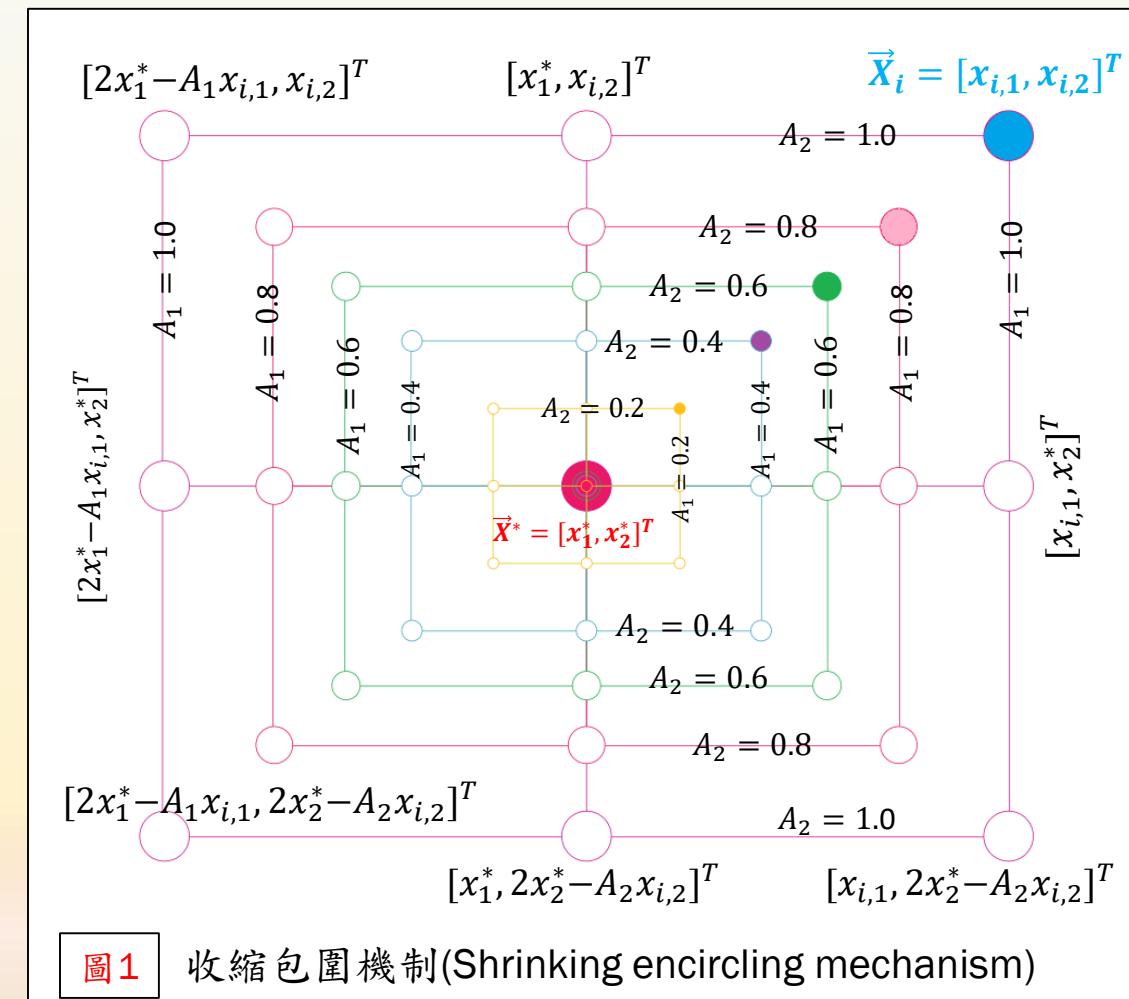
$$\vec{X}_i(t+1) = \vec{X}^*(t) - \vec{A} \cdot |\vec{C} \cdot \vec{X}^*(t) - \vec{X}_i(t)| \quad (\text{p.18式1})$$

氣泡網攻擊方法 (Bubble-Net Attacking Method)

[利用階段 (Exploitation Phase)]

為了對座頭鯨的氣泡網進食(攻擊)行為進行數學建模，WOA設計了收縮包圍機制(Shrinking encircling mechanism)和螺旋更新位置(Spiral updating position)等兩種方法。

1) 收縮包圍機制(Shrinking encircling mechanism)：這種行為是藉由p.18式3中向量 $\vec{a} = [a_1, a_2, \dots, a_D]^T$ 各維度值(即 $a_d, d = 1, 2, \dots, D$)逐漸減少來實現的，其中 a_d 隨疊代增加而從2逐漸減小到0。請注意，向量 $\vec{A} = [A_1, A_2, \dots, A_D]^T$ 的隨機波動範圍隨著向量 \vec{a} 的各維度成分值逐漸減少而縮小。換言之， \vec{A} 之維度 d 的隨機值 A_d 於 $[-a_d, a_d]$ 範圍內波動， $d = 1, 2, \dots, D$ 。故此，鯨魚 i 的新位置 $\vec{X}_i(t+1)$ (參p.18式1)可以定義在 $\vec{X}_i(t)$ 和目前最佳位置 $\vec{X}^*(t)$ 之間的任何位置，for $i = 1, 2, \dots, n_s$ ，其中 n_s 是鯨群大小(The swarm size of whales)。圖1顯示了在二維空間中透過 $\vec{A} = [A_1, A_2]^T$ 可以實現從 $\vec{X}_i = [x_{i,1}, x_{i,2}]^T$ 到 $\vec{X}^* = [x_1^*, x_2^*]^T$ 的可能位置 (圖1考慮情況: $0 \leq A_1 \leq 1$ 及 $0 \leq A_2 \leq 1$)。



Whale Optimization Algorithm

氣泡網攻擊方法 (Bubble-Net Attacking Method)

[利用階段 (Exploitation Phase)] (承上一頁)

2) 螺旋更新位置 (Spiral updating position)：考慮二維空間，螺旋更新位置如圖1所示。此方法首先計算位於 $\vec{X}_i = [x_{i,1}, x_{i,2}]^T$ 的鯨魚與位於 $\vec{X}^* = [x_1^*, x_2^*]^T$ 的獵物位置之間的距離。然後在鯨魚和獵物位置之間創建一個螺旋方程 (A spiral equation)，以模仿座頭鯨的螺旋形運動 (Helix-shaped movement, 參圖2)。如式1及式2所示，其中 b_d 是定義於指數螺旋形狀的參數， l_d 是 $[-1,1]$ 中的隨機數， $d = 1, 2, \dots, D$, D =搜尋空間維度。

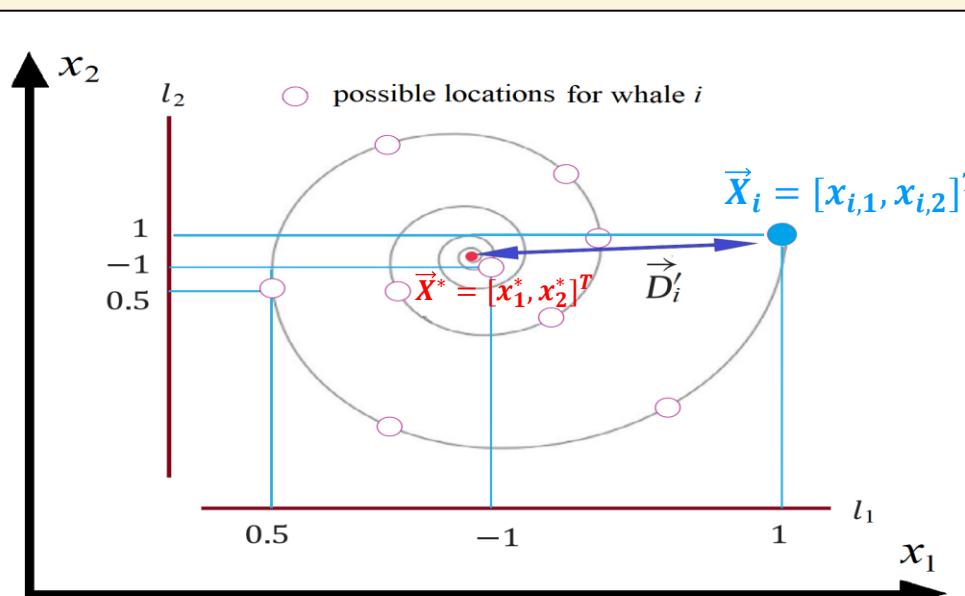


圖1 螺旋更新位置 (Spiral updating position)

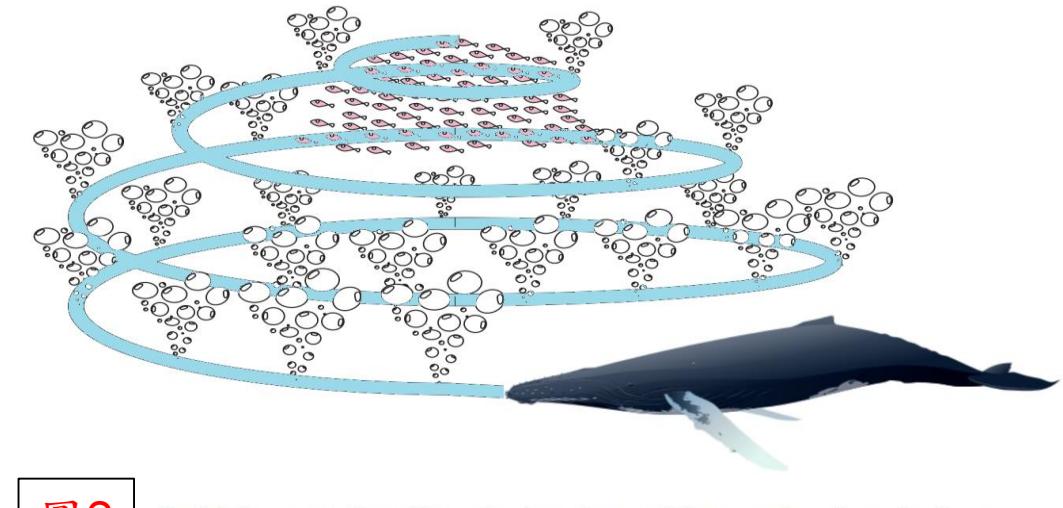


圖2

Bubble-net feeding behavior of humpback whales

$$\vec{X}_i(t+1) = \vec{D}'_i(t) \cdot \exp(B \cdot L) \cdot \cos(2\pi \cdot L) + \vec{X}^*(t) \quad (1)$$

$$\vec{D}'_i(t) = |\vec{X}^*(t) - \vec{X}_i(t)| \quad (2)$$

where

$B = [b_1, b_2, \dots, b_D]^T$ a coefficient vector,

$L = [l_1, l_2, \dots, l_D]^T$ a random vector in $[-1,1]^D$,

$\exp(B \cdot L) = [\exp(b_1 l_1), \exp(b_2 l_2), \dots, \exp(b_D l_D)]^T$,

$\cos(2\pi \cdot L) = [\cos(2\pi l_1), \cos(2\pi l_2), \dots, \cos(2\pi l_D)]^T$,

l_d a random number in $[-1,1]$ for dimension d ,

b_d a coefficient for dimension d , $d=1, 2, \dots, D$,

D the dimension of a search space,

\cdot an element-by-element multiplication,

$\vec{D}'_i = |\vec{X}^*(t) - \vec{X}_i(t)|$ the distance vector of whale i to the prey.

Whale Optimization Algorithm

氣泡網攻擊方法 (Bubble-Net Attacking Method)

[利用階段 (Exploitation Phase)] (承上一頁)

3) 小結(利用階段): 座頭鯨會在不斷縮小的圓圈內圍繞獵物游動, 同時沿著螺旋形路徑遊動。為了模擬這種同時行為WOA假設有50%的機率在最佳化過程中選擇收縮包圍機制(p.18式1)或螺旋機制(p.21式1及式2)來更新鯨魚 i 的位置, 數學模型表達如式1所示。

$$\vec{X}_i(t+1) = \begin{cases} \vec{X}^*(t) - \vec{A} \cdot |\vec{C} \cdot \vec{X}^*(t) - \vec{X}_i(t)| & \text{if } p < 0.5 \\ |\vec{X}^*(t) - \vec{X}_i(t)| \cdot \exp(B \cdot L) \cdot \cos(2\pi \cdot L) + \vec{X}^*(t) & \text{if } p \geq 0.5 \end{cases} \quad (1)$$

where p is a random number in $[0,1]$.

Whale Optimization Algorithm

尋找獵物 (Search for Prey)

[探索階段 (Exploration Phase)]

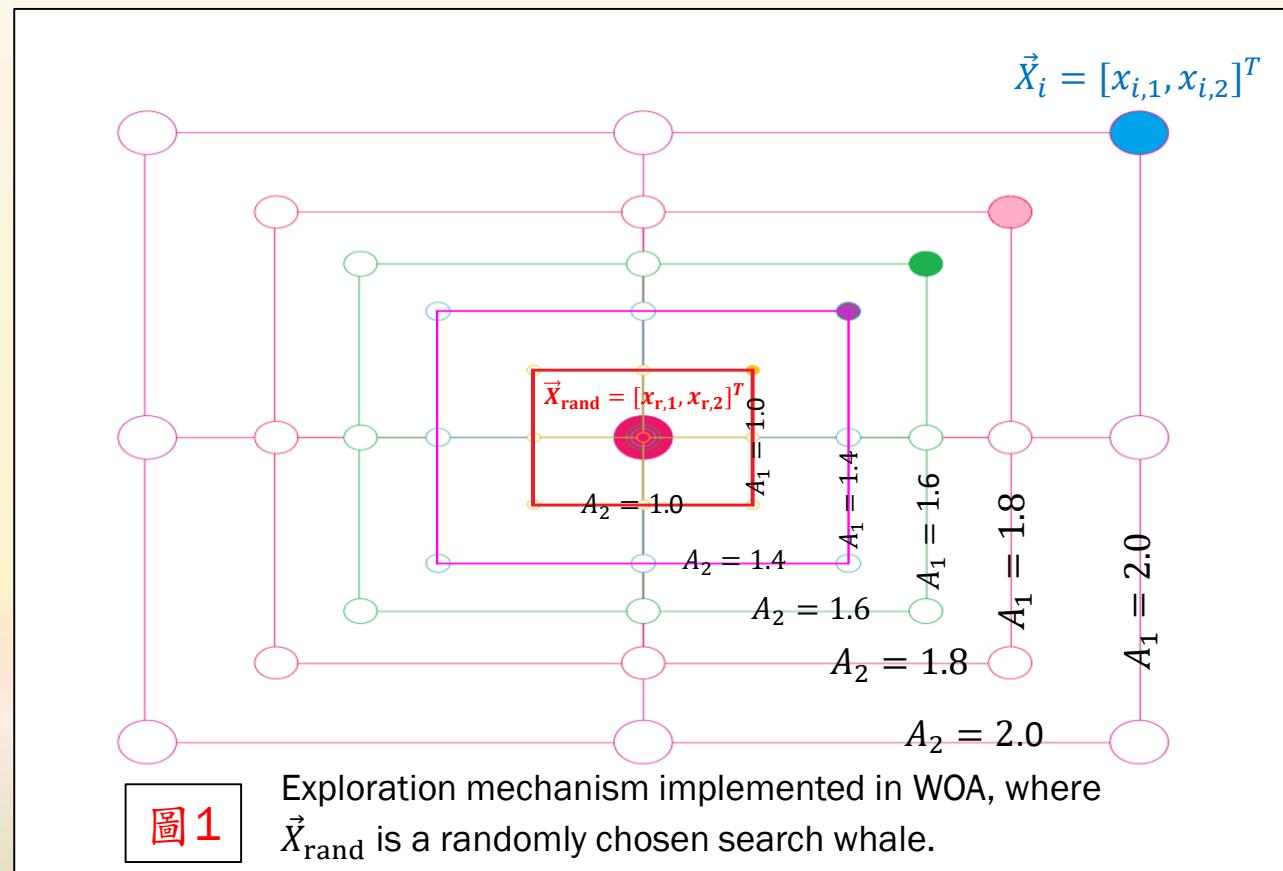
尋找獵物 (探索階段) : 除了利用階段(The exploitation phase)中的氣泡網攻擊方法外,座頭鯨還會隨機搜尋獵物(search for prey randomly),搜尋的數學模型說明如下。在探索階段(The exploration phase)中, 基於向量 \vec{A} 的維度成份值變化,相同的包圍獵物方法(The same method searching for prey)可用於搜尋獵物。事實上,座頭鯨是根據彼此的位置隨機搜尋獵物的。因此,我們使用向量 \vec{A} 的維度成份隨機值(A_d)大於1或小於-1(即 $|A_d| > 1$), $d = 1, 2, \dots, D$, 來強制搜尋鯨魚(A search whale)遠離參考鯨魚(A reference whale)。與利用階段相反,我們根據隨機選擇的參考鯨魚(而不是目前找到的最佳位置)來更新探索階段中搜尋鯨魚的位置。這種機制和 $|A_d| > 1$ 強調探索(Exploration)並允許WOA演算法執行全域搜尋(A global search)。在探索階段, 鯨魚 i 的位置更新, $\vec{X}_i(t + 1)$, 如式1所示。 $|A_d| > 1$ 的 $\vec{X}_{\text{rand}} = [x_{r,1}, x_{r,2}, \dots, x_{r,D}]^T$ 週圍一些 $\vec{X}_i(t + 1)$ 可能位置。 $D = 2$ 時,如圖1所示。

$$\vec{X}_i(t + 1) = \vec{X}_{\text{rand}}(t) - \vec{A} \cdot |\vec{C} \cdot \vec{X}_{\text{rand}}(t) - \vec{X}_i(t)| \quad (1)$$

where

$\vec{X}_{\text{rand}}(t)$ a random position vector (a random whale) chosen from the current population,

$\vec{D}_i(t) = |\vec{C} \cdot \vec{X}_{\text{rand}}(t) - \vec{X}_i(t)|$ the distance of whale i to a randomly chosen whale from the current swarm.



Whale Optimization Algorithm

WOA 最佳化演算法:: 步驟

假設 WOA 的搜尋空間是 D 維的。設 $f: R^D \rightarrow R$ 是必須最佳化的損失函數(The loss function)。基本 WOA 優化演算法的步驟如下。

Step 1: 初始化位置向量 \vec{X}^* (獵物位置)。

執行迴圈 $i = 1, 2, \dots, n_s$ [初始化之鯨魚粒子迴圈]

使用均勻分佈的隨機向量初始化鯨魚位置向量(即候選解) $\vec{X}_i \in R^D$, 其中, 位置分量 $x_{i,d}$ 於維度 d 設定搜尋空間範圍之內, $d = 1, 2, \dots, D$ 。

若 $f(\vec{X}_i) < f(\vec{X}^*)$, 則 $\vec{X}^* \leftarrow \vec{X}_i$ 。 [更新 \vec{X}^*]

直到初始化之粒子迴圈執行完畢。

Step 2: 執行迴圈 $i = 1, 2, \dots, n_s$ [鯨魚粒子迴圈]

執行迴圈 $d = 1, 2, \dots, D$ [搜尋空間維度迴圈]

取得維度 d 的 $\{a_d, A_d, C_d, l_d\}$ 和機率 p 之值;

if $p < 0.5$,

if $|A_d| < 1$,

 使用 p.18 式 1 更新 \vec{X}_i 之位置分量 $x_{i,d}$; [收縮包圍(利用階段)]

else if $|A_d| \geq 1$,

 隨機選擇一隻鯨魚 \vec{X}_{rand} 之位置分量 $x_{rand,d}$;

 使用 p.23 式 1 更新 \vec{X}_i 之位置分量 $x_{i,d}$; [尋找獵物(探索階段)]

end

else if $p \geq 0.5$,

 使用 p.21 式 1 及式 2 更新 \vec{X}_i 之位置分量 $x_{i,d}$; [螺旋機制(利用階段)]

end

直到搜尋空間維度迴圈執行完畢。[得到鯨魚 i 之完整位置向量 \vec{X}_i 更新]

若 $f(\vec{X}_i) < f(\vec{X}^*)$, 則 $\vec{X}^* \leftarrow \vec{X}_i$ 。 [更新 \vec{X}^*]

直到鯨魚粒子迴圈執行完畢。[得到所有鯨魚位置向量更新]

Step 3: 檢查停止條件。如果停止條件滿足, 則停止該 WOA 演算法。否則, 返回 Step 2 尋找更佳 \vec{X}^* 位置向量。

Ant Colony Optimization (ACO)

(蟻群最佳化演算法)

Ant Colony Optimization (ACO)

蟻群(或較普遍的社會性昆蟲社會)是一種分散式系統,儘管它們的個體很簡單,但卻呈現出高度結構化的社會組織(A highly structured social organization)。由於蟻群的這種組織,它們可以完成複雜的任務,在某些情況下遠遠超出單一隻螞蟻的能力。”螞蟻演算法”領域研究(The field of “ant algorithms”)涉及對真實螞蟻行為的觀察並從中得出模型,此類螞蟻行為模型可以成為新穎演算法的靈感來源,用以解決最佳化和分散式控制問題。靈感啟發的主要想法是:可以利用真實螞蟻高度協調的自組織(The self-organizing principles),來設計人工蟻群(Populations of artificial agents),解決最佳化問題。蟻群行為的幾個不同方面激發了不同類型的螞蟻演算法,例如,覓食(Foraging)、勞動分工(Division of labor)、幼蟻巢穴分類(Brood sorting)和合作運輸(Cooperative transport)。基於這些例子,學者指出,螞蟻透過共識主動性(Stigmergy) [1] 協調它們的活動。共識主動性指藉由環境改變作為媒介的一種間接交流形式。例如,一隻覓食的螞蟻會在地面上沉積一種化學物質,促使其他螞蟻遵循相同路徑的可能性增加。生物學家已經觀察到:社會性昆蟲的許多群體層面之行為可以透過相當簡單的模型來解釋,即群體僅有共識主動之交流(Stigmergic communication),就可發展出這些群體層面的行為。換言之,基於共識主動性的間接交流通常足以解釋社會性昆蟲群體如何實現自組織。螞蟻演算法的概念是利用一種人工式共識主動性形式(A form of artificial stigmergy)來協調人工蟻群的社會(Societies of artificial agents)。螞蟻演算法最成功的例子之一被稱為蟻群最佳化演算法(Ant colony optimization, ACO)。許多螞蟻物種的視覺感知能力仍處於初級階段,有些螞蟻物種完全失明。事實上,早期對螞蟻行為研究的一個重要見解是:個體之間或個體與環境之間的大部分交流都是基於使用螞蟻產生的化學物質。這些化學物質稱為費洛蒙(Pheromones,或稱信息素)。這與人類(或其他高等物種)發生的情況不同,它們最重要的感官是視覺或聽覺。對於某些螞蟻物種,它們的社會生活特別重要的是蹤跡費洛蒙(The trail pheromone)。蹤跡費洛蒙是一種特定類型的費洛蒙,一些螞蟻物種(例如*Lasius niger*或阿根廷螞蟻*Iridomyrmex humilis* [4])使用它來標記地面上的路徑,例如從食物源到蟻巢的路徑。透過感知蹤跡費洛蒙,一隻覓食蟻可以追蹤由其他多隻螞蟻已發現的食物路徑。某些螞蟻受到其他螞蟻留下的化學蹤跡(A chemical trail)的集體鋪設和追尋這些路徑蹤跡的行為之影響成為了發展ACO的靈感來源。

[1]共識主動性(Stigmergy) [2][3] 是一個由 Pierre-Paul Grassé 發明的新詞(1959年),用來解釋白蟻的築巢行為。這個詞是在幾個希臘語詞根的基礎上造出來的,源自希臘語單字“στίγμα”(stigma, 標記、標誌之意)和“ἔργον”(ergon, 工作、行動、功(能量)之意),並捕捉了這樣一個概念:一個代理者的行為(An agent's action)在環境中留下了跡象(Signs,標記),這些跡象可以被它和其他代理人感知到,並決定和影響他們的後續行動。因此,”stigmergy”的含義是「標記+行動」。Grasse 的靈感來自於螞蟻群和白蟻群,這些蟻群幾乎是沒有任何智慧的生物,居然創立了如此精緻的信息系統,並建造了複雜的建築結構,Grasse 想其中必有什麼奧妙。通過觀察發現,在螞蟻的大腦或者基因裡,並沒有一個關於巢穴的計劃、組織和控制機制,螞蟻之間甚至沒有直接的交流。因此,蟻巢的精緻框架和複雜結構,完全是每個螞蟻單獨識別其他螞蟻留下的生物激素和存留物後,達成共識並直接行動的結果。(Wiki)

[2]Bonabeau, Eric (1999). "Editor's Introduction: Stigmergy." Artificial Life. 5 (2): 95–96.

[3]Theraulaz, Guy (1999). "A Brief History of Stigmergy." Artificial Life. 5 (2): 97–116.

[4] Goss, S., Aron, S., Deneubourg, J.-L., Pasteels, J. (1989). "Selforganized shortcuts in the Argentine ant." Naturwissenschaften. 76: 579–581.

Ant Colony Optimization (ACO)

蟻群最佳化演算法(Ant colony optimization, ACO)最初是為解決組合最佳化問題(Combinatorial optimization problems, COPs)而開發的演算法。ACO是一種受螞蟻覓食行為所啟發的最佳化演算法,1992年由學者Dorigo提出[1]。蟻群最佳化演算法(ACO)自1992年以來已經有許多不同變體(Variants) [2]。

- 1) **組合最佳化問題:** 組合最佳化涉及尋找可用問題組件的最佳組合或排列 (combinations or permutations of available problem components)。因此,需要將問題劃分為一組有限的組件,並且組合最佳化演算法嘗試找到它們的最佳組合或排列。許多現實世界的最佳化問題可以用簡單方式表示為組合最佳化問題(COP)。
- 2) **連續最佳化問題:** 然而最佳化問題並非總是如COP問題一樣。最佳化還存在另一類重要的問題: 需要為連續變數選擇值的最佳化, 即所謂的連續最佳化問題。雖然我們可以允許變數之值的連續範圍轉換為有限集合,然後再使用組合最佳化演算法來解決此類COP問題。但這並不總是很方便的,特別是初始可能範圍很寬且所需的解析度非常高時。在這些情況下,若演算法本身可以處理連續變數,其表現通常會更好。

註: ACO可以進一步發展為連續型蟻群最佳化演算法(Ant Colony Algorithm for continuous domains, 記為 ACO_R) [3]。 ACO_R 可以應用於連續最佳化問題(Continuous optimization problems)。



- [1] Dorigo, M. (1992), Optimization, Learning and Natural Algorithms (in Italian). Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy.
- [2] Dorigo, M., Stutzle, T., 2004. Ant Colony Optimization. MIT Press, Cambridge, MA, USA.
- [3] Krzysztof Socha and Marco Dorigo (2008), Ant colony optimization for continuous domains. European Journal of Operational Research 185, 1155–1173.

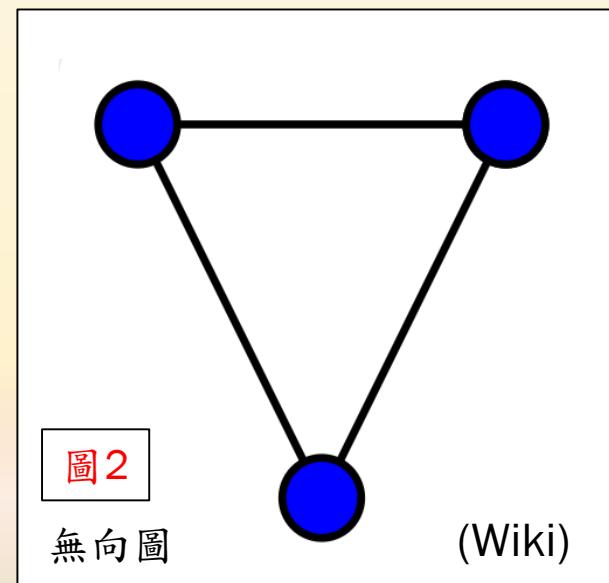
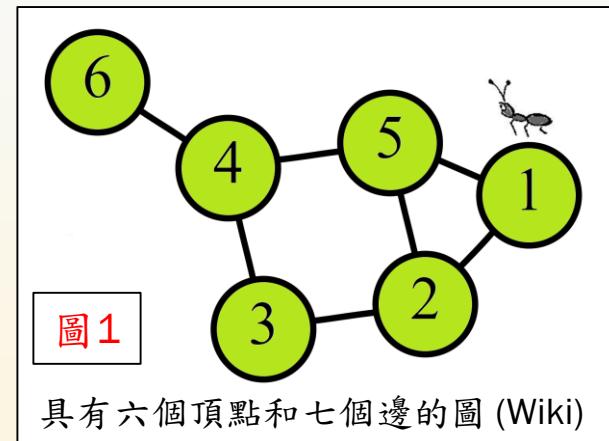
Ant Colony Optimization (ACO)

1) 組合最佳化問題中，銷售員旅行問題(The traveling salesman problem, TSP)是最具代表性之一，其中銷售員需要以最短路徑走完他要去的城市，每個城市只拜訪一次，此涉及圖之路徑最小化問題。圖(Graph)是圖論(Graph theory)中的基本概念。1878年，詹姆斯·西爾維斯特(J. J. Sylvester, 1814–1897, 英國數學家)首次使用「圖」這一名詞：他用圖來表示數學和化學分子結構之間的關係(他稱為「化學圖」，chemico-graphical image)。在離散數學中，更具體地在圖論中，圖是用於表示物件(Object)與其他物件之間存在某種關係的結構(A structure)。數學抽象後的「物件」稱作節點或頂點(Vertex、node、point)。每個相關的頂點對(A pair of vertices)稱為邊(An edge, 也稱為連結 link、線 line)。在描繪一張圖的時，通常用一組點或小圓圈表示節點，其間的邊則使用直線或曲線。參見圖1。有時，圖的定義中允許自環(簡稱環，loop)，即一條連接一個節點和其自身的邊。允許自環的圖也稱為自環圖(Graphs with loops)。圖分為兩大類，即無向圖(Undirected graphs)及有向圖(Directed graphs)。(Ref. to [Wiki](#))

2) 無向圖：無向圖也稱簡單圖(A simple graph)。一張無向圖是一個二元組 $G = (V, E)$ ，其中集合V中的元素稱為節點，集合E中的元素是兩個節點組成的無序對，稱為邊。參見圖2。集合V稱為點集(A set of vertices)，E稱為邊集(A set of edges)。一條邊 $\{x, y\}$ 上的兩個節點x和y稱作這條邊的頂點或端點；也說這條邊連接了節點x和y，或這條邊與x和y關聯(Incident)。一個節點可以不屬於任何邊(即它不與任何節點相連)。由於 $\{x, y\}$ 是無序對， $\{x, y\}$ 及 $\{y, x\}$ 表示的是同一個元素。更廣義地，多重圖(A multigraph)的定義中允許同一對節點之間存在多條不同的邊。在特定語境中，多重圖也可以直接稱作圖(Multigraphs are simply called graphs.)。

■ 圖的階(Order)是指其中節點的數量(記為 $|V|$)。圖的邊數是指其邊的數量(記為 $|E|$)。圖的大小(Size)一般也指圖的邊數。但在一些語境中，例如描述算法複雜度的時候，圖的大小是指 $|V| + |E|$ (否則非空圖的大小也有可能是0)。節點的度(Degree、valency)是指連接到該節點的邊的數量；對於允許自環的圖，環要算做兩條邊(因為兩端都連接到這個節點)。

■ 如果一個圖的階是 n ，則其節點的度最大可能取 $n - 1$ ，而邊最多可能有 $n(n - 1)/2$ 條。對於允許自環的圖，則節點的度最大可能是 n ，而邊最多可能有 $n(n + 1)/2$ 條。



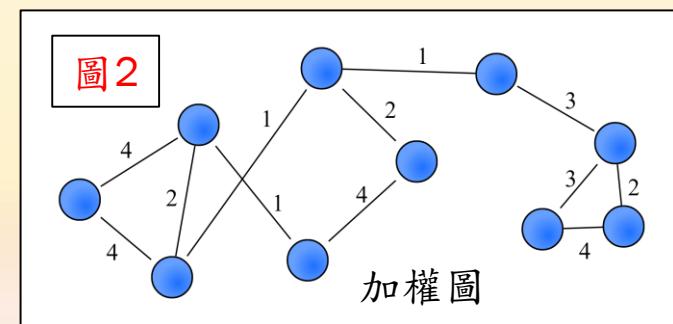
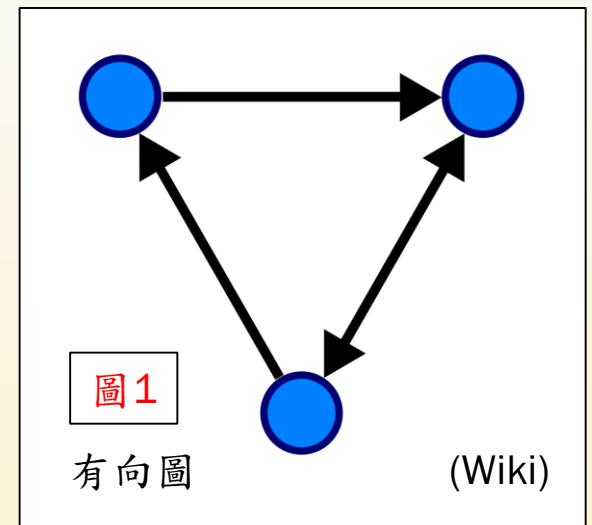
Ant Colony Optimization (ACO)

(承上一頁)

3) **有向圖**: 有向圖也稱為有向簡單圖(A directed simple graph; 為了與有向多重圖區分)。參見**圖1**。一張有向圖的一種比較嚴格的定義：一個二元組 $G = (\mathbf{V}, \mathbf{E})$, 其中 \mathbf{V} 是節點的集合； $\mathbf{E} \subseteq \{(x, y) | (x, y) \in \mathbf{V}^2 \wedge x \neq y\}$ 是有向邊(Directed edges、directed links、arrows、arcs(弧))的集合；其中的**有向邊是節點的有序對**(Ordered pairs of distinct vertices)。在從 x 到 y 的邊 (x, y) 上，節點 x 和 y 稱作這條邊的頂點，其中 x 是起點或尾(The tail of the edge), y 是終點或頭(The head of the edge)。也說這條邊連接了節點 x 和 y 、這條邊與節點 x 和 y 關聯。一張有向圖中的節點可以不屬於任何一條邊。邊 (y, x) 稱為邊 (x, y) 的反向邊。更廣義地，如果一張有向圖允許多條頭尾都分別相同的邊，則稱這樣的圖為**有向多重圖**(A directed multigraph)，這樣的邊稱為**多重邊**(Multiple edges)。一種允許多重邊的有向圖定義方法如下：一個有向圖是這樣的三元組 $G = (\mathbf{V}, \mathbf{E}, \phi)$, 其中 \mathbf{V} 是節點的集合； \mathbf{E} 是邊的集合； $\phi: \mathbf{E} \rightarrow \{(x, y) | (x, y) \in \mathbf{V}^2 \wedge x \neq y\}$ 是一個關聯函數，將每條邊映射到一個由頂點組成的有序對上 (即一條邊被按順序關聯到兩個頂點)。節點的**出度**(Out-degree)是指**起點**為該節點的邊的數量；節點的**入度**(In-degree)是指**終點**為該節點的邊的數量。

■ **自環**是指一條起點和終點相同的邊。前面的兩個定義都不允許自環，因為若節點 x 上有一個自環，則邊 (x, x) 存在；但這樣的邊不在 $\{(x, y) | (x, y) \in \mathbf{V}^2 \wedge x \neq y\}$ 中。因此，如果要允許自環，則上面的定義要做修改：對於**有向簡單圖**， E 的定義應該修改為 $\mathbf{E} \subseteq \{(x, y) | (x, y) \in \mathbf{V}^2\}$ 。對於**有向多重圖**， ϕ 的定義應該修改為 $\phi: \mathbf{E} \rightarrow \{(x, y) | (x, y) \in \mathbf{V}^2\}$ 。這樣的圖分別稱作**允許自環的有向簡單圖**及**允許自環的有向多重圖**(Quiver)。

4) **加權圖**: 加權圖(A weighted graph)也稱賦權圖或網絡(A network)是指**每條邊都對應有一個數字**(即**權重**, weight)的圖。根據具體問題，權重可以表示諸如費用、長度或容量等意義。這樣的圖會出現在**最短路徑問題**和**銷售員旅行問題**(TSP)等問題中。參見**圖2**。



Ant Colony Optimization (ACO)

蟻群最佳化演算法(Ant colony optimization, ACO)作為一種受自然啟發的方法,並被使用於解決組合最佳化問題(Hard combinatorial optimization problems)。ACO的靈感來源是真實螞蟻的覓食行為(The foraging behavior of real ants)。在尋找食物時,螞蟻最初會隨機探索蟻穴週圍的區域。一旦螞蟻找到食物來源,它就會對其進行評估並將一些食物帶回蟻穴。在回程過程中,螞蟻會在地面上留下費洛蒙蹤跡(A pheromone trail)。沉積的費洛蒙[1](沉積的費洛蒙數量可能取決於食物數量和品質)引導其他螞蟻到達食物源。螞蟻之間透過費洛蒙蹤跡進行的間接交流使它們能夠找到蟻穴和食物源之間的最短路徑[2]。真實蟻群(Real ant colonies)的這種能力啟發了人工蟻群(Artificial ant colonies)的定義。人工蟻群可用於尋找COP的近似解。

- ACO 演算法的核心組成部分是費洛蒙模型(A pheromone model),該模型以機率方式對搜尋空間進行採樣。
- 費洛蒙模型可以從COP模型中推導出來[3]。

[1] **費洛蒙**(Pheromone)源自古希臘文「φέρω」(phérō, 意指“我攜帶”)與「օρμή」(hormé, 意指“刺激”),合起來意思是“我攜帶刺激物”的意思。「費洛蒙」一詞由 Peter Karlson 和 Martin Lüscher 於 1959 年所創造。費洛蒙(也稱作外激素)是一種化學傳訊素(訊息化合物, semiochemical),是指由一個個體分泌到體外且具有揮發性的化學物質,被同物種的其他個體通過嗅覺器官(例如,副嗅球、犁鼻器)察覺,使後者表現出行為、情緒、心理或生理機制改變的物質。費洛蒙具有促使同物種個體間之交流功能。幾乎所有動物都有費洛蒙的存在。慕尼黑大學的 Adolf Butenandt 教授,歷經 30 年的努力,在 1959 年以德文發表雌蠶蛾分泌性費洛蒙的研究成果;這是科學界首次證明了性費洛蒙是存在的[\[Link\]](#)。**費洛蒙**是一種被分泌或排泄的化學因子,可引發同物種成員的社會反應。費洛蒙就像是分泌於體外的荷爾蒙,其作用會影響接收個體的行為。從基本的單細胞原核生物到複雜的多細胞真核生物,許多生物體都使用費洛蒙。此外,一些脊椎動物、植物和纖毛蟲透過使用費洛蒙進行溝通。費洛蒙的生態功能和演化是化學生態學領域的一個主要研究主題。費洛蒙可依其作用目的分成警報、追蹤、性、聚集、空間、其他等種類(通常,這些是用來描述昆蟲的費洛蒙)。費洛蒙依作用機制分為釋放體費洛蒙(Releaser pheromones)與引體費洛蒙(Primer pheromones)兩種。由生物釋放並具有傳遞資訊功能的物質通稱為訊息傳遞素(Semiochemicals),除了費洛蒙外,還有開洛蒙(Kairomone)、阿洛蒙(Allomone)與新洛蒙(Synomone)等,都是不同物種間個體傳遞資訊的物質(相對地,費洛蒙是同物種個體間傳遞資訊的物質)。開洛蒙專指對接收者有利,但不利於釋放者的物質,例如,由獵物釋放並可被掠食者偵測的。阿洛蒙則對釋放者有利但對接收者不利,例如,由植物所釋放以抵抗昆蟲的化學物質。新洛蒙則對釋放者與接收者均有利。(Wiki)

- [2] Goss, S., Aron, S., Deneubourg, J.-L., Pasteels, J. (1989). Selforganized shortcuts in the Argentine ant. *Naturwissenschaften* 76, 579–581.
- [3] Blum, C. (2004). Theoretical and Practical Aspects of Ant Colony Optimization. Ph.D. thesis, vol. 282 of *Dissertationen zur Künstlichen Intelligenz*. Akademische Verlagsgesellschaft Aka GmbH, Berlin, Germany.

Ant Colony Optimization (ACO)

組合最佳化問題模型 (A Model of COP)

因為從COP模型可以推導出費洛蒙模型，我們討論COP模型，記為 $P = (\mathbf{S}, \Omega, f)$ ，如式1所示。如果集合 Ω 為空集合，則 P 稱為無約束問題模型，否則稱為約束問題模型。

- 搜尋空間 \mathbf{S} 定義如下。在空間 \mathbf{S} 中，給了一組離散決策變數(A finite set of discrete decision variables), $\{X_i, i = 1, 2, \dots, n\}$ ，其中 X_i 的值 $v_i^j \in \mathbf{D}_i$ ； $\mathbf{D}_i = \{v_i^1, v_i^2, \dots, v_i^{|\mathbf{D}_i|}\}$ 是一個離散集合，是 X_i 的定義域(Domain)。變數實例化(A variable instantiation)指將值 v_i^j 賦予變數 X_i ，記為 $X_i \leftarrow v_i^j$ 。
- 當每個決策變數 X_i (for $i = 1, 2, \dots, n$) 都分配了一個值而且滿足集合 Ω 中的所有約束條件(All the constraints)，我們就得到一個解 s , $s \in \mathbf{S}$ 。換言之，給定COP, s 是一個可行解(A feasible solution)。亦即，一個可行解由 n 個決策變數實例化組織而成。
- 解 $s^* \in \mathbf{S}$ 稱為全域最佳解(A global optimum) 若且唯若(if and only if) $f(s^*) \leq f(s) \quad \forall s \in \mathbf{S}$ 。所有全域最佳解的集合記為 \mathbf{S}^* , $\mathbf{S}^* \subseteq \mathbf{S}$ 。我們需要為COP找出至少一個全域最佳解 s^* , $s^* \in \mathbf{S}^*$ 。

$$P = (\mathbf{S}, \Omega, f) \quad (1)$$

where

- P a model of a combinatorial optimization problem (COP),
 \mathbf{S} a search space defined over a finite set of discrete decision variables,
 Ω a set of constraints among the variables,
 f an objective function $f : \mathbf{S} \rightarrow \mathbb{R}_0^+$ to be minimized.

ACO的費洛蒙模型 [基於COP模型]

- 1) 首先，實例化的決策變數 $X_i = v_i^j$ (即變數 X_i 的值 v_i^j 從其定義域 $\mathbf{D}_i = \{v_i^1, v_i^2, \dots, v_i^{|\mathbf{D}_i|}\}$ 取得) 稱為解組件(A solution component, 或稱解分量)，該解組件記為 c_{ij} 。
- 2) 所有可能的解組件之集合以 \mathbf{C} 表示，即 $\mathbf{C} = \{c_{ij}\}$ 。
- 3) 然後，費洛蒙踪跡參數(記為 T_{ij}) 與每個解組件 c_{ij} 相關聯。
- 4) 所有費洛蒙踪跡參數的集合以 \mathbf{T} 表示，即 $\mathbf{T} = \{T_{ij}\}$ 。
- 5) T_{ij} 的值以 τ_{ij} 表示，其中 τ_{ij} 稱為費洛蒙值(A pheromone value)。(τ_{ij} 通常是演算法疊代 t 的函數，即 $\tau_{ij} = \tau_{ij}(t)$ 。)
- 6) 然後，ACO演算法在搜尋過程中使用並更新 τ_{ij} 。費洛蒙值 τ_{ij} 允許對該解之不同解組件的機率分佈進行建模。

Ant Colony Optimization (ACO)

ACO元啟發式演算法(The ACO metaheuristic)

演算法概述: 在自然界中,某些物種的螞蟻最初隨機漫步,找到食物後返回蟻穴,同時留下費洛蒙蹤跡。如果其他螞蟻找到了這樣的路徑,它們很可能不會繼續隨機旅行,而是沿著這條路徑前進,如果它們最終找到食物,就會返回並分泌它們的費洛蒙以加強該路徑。然而,隨著時間的推移,費洛蒙(信息素)痕跡開始蒸發,降低了其吸引力。螞蟻沿著路徑行走並返回所需的時間越長,信息素蒸發的時間就越長。相較之下,短路徑會更頻繁地經過,因此較短路徑上的費洛蒙密度比長路徑上的費洛蒙密度更高。信息素蒸發也具有避免收斂到局部最佳解的優點。如果根本沒有蒸發,第一隻螞蟻選擇的路徑往往會對後續螞蟻產生過度的吸引力。在這種情況下,對解空間的探索將受到限制。信息素蒸發在真實螞蟻系統中的影響尚不清楚,但在人工蟻系統中非常重要。總體結果是,當一隻螞蟻找到從蟻群到食物源的良好路徑(即短路徑)時,其他螞蟻更有可能遵循該路徑,並且藉由費洛蒙而產生的正回饋(即自催化)過程(A positive feedback process or called an autocatalytic process) [1] 最終導致許多螞蟻遵循單一路徑。蟻群演算法的想法是透過人工螞蟻(Artificial ants, 即模擬螞蟻 simulated ants)在代表需要解決的問題的圖(A graph)上行走來模仿這種覓食行為。參見圖1。

- ACO元啟發式演算法由三個演算法元件區塊組成,如圖2所示,分別是:
AntBasedSolutionConstruction()、**PheromoneUpdate()**、及**DaemonActions()**。這些元件聚集在**ScheduleActivities**構造中。**ScheduleActivities**構造沒有指定如何排程和同步這三個元件之活動,該決定留給演算法設計者。
- ACO (for “ant-cycle algorithm”)的虛擬碼,請參見[3]。

以下將解釋演算法的這三個元件區塊。

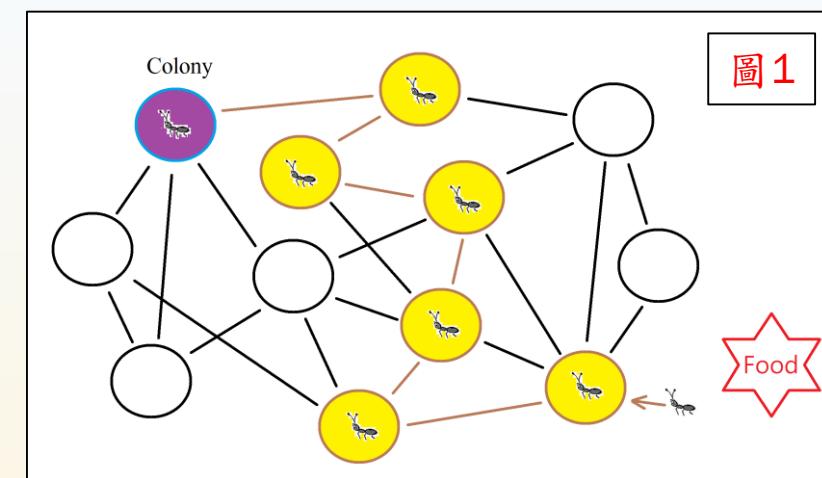


圖1

Ant Colony Optimization metaheuristic

```
while termination conditions not met do
    ScheduleActivities
        AntBasedSolutionConstruction()
        PheromoneUpdate()
        DaemonActions()      {optional}
    end ScheduleActivities
endwhile
```

圖2

- [1] 自催化過程(Autocatalytic process, 即正回饋)[2]是一個自我強化的過程,其方式會導致非常快速的收斂,如果不存在限制機制,則會導致爆炸。
- [2] M. Dorigo, “Optimization, learning and natural algorithms,” Ph.D. Thesis, Dip. Elettronica e Informazione, Politecnico di Milano, Italy, 1992.
- [3] Dorigo, M., Maniezzo, V., Colorni, A. (1996). Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics – Part B 26 (1), 29–41.

Ant Colony Optimization (ACO)

ACO元啟發式演算法(The ACO metaheuristic) (承上一頁)

AntBasedSolutionConstruction(): 每隻螞蟻透過一個解組件的序列來構建出一個可行解。該序列共有 n 個可用解組件 (n available solution components), 序列的每個解組件皆取自集合 $\mathbf{C} = \{c_{ij}\}$ 。構建一個可行解的過程可以看成是構建圖 $G_C = (\mathbf{V}, \mathbf{E})$ 上的一條完整路徑。一個可行解之構建從空解組件 $s^p = \emptyset$ 開始。假設螞蟻 k 位於節點 i (即狀態 i)。然後, 在每一個可行解的構建步驟中, 透過加入由解構建機制(The solution construction mechanism)定義的集合 $N_i(s^p)$ 中取出一個可用解組件, 來擴展目前解組件 s^p (The current partial solution)。 $N_i(s^p) \subseteq \mathbf{C} \setminus s^p$, 其中記號 " $\mathbf{C} \setminus s^p$ " 表示"從集合 \mathbf{C} 中排除目前解組件 s^p "之意。集合 \mathbf{C} (即所有可能的解組件 c_{ij} 之集合)可以與圖 G_C 的頂點集合 \mathbf{V} 相關聯, 或與其邊集合 \mathbf{E} 相關聯。圖 G_C 中允許的路徑由解構造機制隱式定義, 該機制定義了節點 i 的可用解組件集合 $N_i(s^p)$ 。[參見 p.32 圖 1]

- 從 $N_i(s^p)$ 中選擇一個可用解組件(即從節點 i 過渡到下一節點 j 的一條路徑 c_{ij})是在每個構建步驟中以機率方式完成的。節點 i 的可用解組件的機率選擇之確切規則因ACO不同變體而異。[節點 i 過渡到下一節點 j 意味著在兩個決策變數間的轉移改變。]
- 依據 Ant System (AS) [1], 螞蟻 k 從節點 i 要過渡到另一節點 j 時, 其選用某一可用解組件(即 c_{ij})之選擇機率 $p_{ij}^{(k)}$, 如式1。

$$p_{ij}^{(k)} = p^{(k)}(c_{ij}|s^p) = \frac{[\tau_{ij}]^\alpha \cdot [\eta(c_{ij})]^\beta}{\sum_{c_{il} \in N_i(s^p)} [\tau_{il}]^\alpha \cdot [\eta(c_{il})]^\beta}, \quad \forall c_{ij} \in N_i(s^p) \quad (1)$$

where

-
- $p_{ij}^{(k)}$ a probability that the k th ant selects edge (i,j) , from node i to node j ,
 c_{ij} an instantiated solution component assigned from the domain \mathbf{D}_i of a discrete variable X_i in the search space \mathbf{S} , $i = 1, \dots, n$, $c_{ij} \in \mathbf{C}$,
 \mathbf{D}_i the domain of a discrete variable X_i , $\mathbf{D}_i = \{v_i^1, \dots, v_i^{|\mathbf{D}_i|}\}$,
 v_i^j the j th element of the set \mathbf{D}_i , $j=1,2, \dots, |\mathbf{D}_i|$, (即解組件 c_{ij})
 \mathbf{C} the set of all possible solution components, $\mathbf{C} = \{c_{ij}\}$,
 \mathbf{S} the search space defined by a set of discrete variables X_i , $i = 1, \dots, n$,
 s^p the current solution component at node i , [a partial solution of s]
 s a feasible solution $s \in \mathbf{S}$, that satisfies the constraint set Ω ,
 $N_i(s^p)$ the set of available solution components at node i , $N_i(s^p) \subseteq \mathbf{C} \setminus s^p$,
 τ_{ij} 費洛蒙資訊 the pheromone value associated with component c_{ij} ,
 $\eta(\cdot)$ a weighting function that assigns at each construction step a heuristic value to each feasible solution $c_{ij} \in N_i(s^p)$,
 (The values that are given by the weighting function $\eta(\cdot)$)
 (are commonly called the heuristic information. 啟發資訊)
 α, β positive parameters, whose values determine the relation between pheromone information and heuristic information.

[1] Dorigo, M., Maniezzo, V., Colorni, A. (1996). Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics – Part B 26 (1), 29–41.

Ant Colony Optimization (ACO)

ACO元啟發式演算法(The ACO metaheuristic) (承上一頁)

Comments on 路徑(i, j)選擇(Edge selection) 及 完整路徑構建(Tour construction)

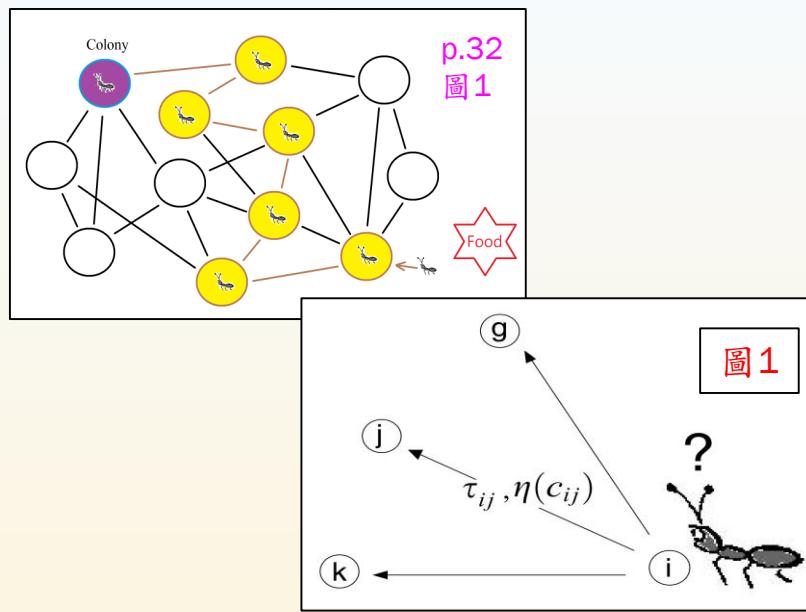
每隻螞蟻都需要構造一個可行解(即一條完整路徑,參見p.32圖1),以在圖(A graph)中進行旅行。對於螞蟻 k ,完整路徑(A tour by ant k)記為 $T^{(k)}$,其路徑長度記為 $L^{(k)}$ 。為了選擇其旅程中當前位置的下一條邊,螞蟻會考慮從其當前位置的每條可用邊的長度以及其相應的費洛蒙水平。參見圖1。在演算法的每一步,每隻螞蟻從節點(狀態) i 移動到下一節點(狀態) j ,此移動對應於更完整路徑的一個中間解組件。因此,螞蟻 k 在每次疊代中計算當前節點(狀態 i)的一組可行擴展(即, p.33式1的 $N_i(s^p)$),並以機率方式移動到其中的下一個節點(狀態 j)。對於螞蟻 k ,從狀態 i 移動到狀態 j 的機率 $p_{ij}^{(k)}$ 取決於兩個值的組合: τ_{ij} 及 $\eta(c_{ij})$ 。(參見p.33式1)。

■ $\eta(c_{ij})$ 是一種路徑移動吸引力(An attractiveness of the move),它透過某種啟發式計算得出,指示該移動的先驗合意性(A priori desirability of that move)。 $\eta(c_{ij})$ 通常是 $1/d_{ij}$,其中 d 是距離函數。

■ τ_{ij} 是該移動的費洛蒙踪跡水平(A pheromone trail level),指示過去執行該移動的熟練程度特別的舉動。費洛蒙踪跡水平代表該移動之合意性的後驗指示(A posteriori indication of the desirability of that move)。

■ P.33式1中的 α 及 β : $\alpha > 0$ 是 τ_{ij} 的控制影響參數,通常設定 $\alpha = 1$ (for TSP)。 $\beta \geq 1$ 是 $\eta(c_{ij})$ 的控制影響參數,通常設定 $2 \leq \beta \leq 5$ (for TSP)。參見表1 [1]。

[1] Dorigo, M., Stutzle, T. (2004). Ant Colony Optimization. P.71. MIT Press, Cambridge, MA, USA.



$$p_{ij}^{(k)} = p^{(k)}(c_{ij} | s^p) = \frac{[\tau_{ij}]^\alpha \cdot [\eta(c_{ij})]^\beta}{\sum_{c_{il} \in N_i(s^p)} [\tau_{il}]^\alpha \cdot [\eta(c_{il})]^\beta} \quad (\text{p.33式1})$$

表1

Parameter Settings for ACO Algorithms (for the TSP) without Local Search

ACO algorithm	α	β	ρ	m	τ_0
AS	1	2 to 5	0.5	n	m/C^{mn}
EAS	1	2 to 5	0.5	n	$(e+m)/\rho C^{mn}$
AS _{rank}	1	2 to 5	0.1	n	$0.5(r-1)/\rho C^{mn}$
MMAS	1	2 to 5	0.02	n	$1/\rho C^{mn}$
ACS	—	2 to 5	0.1	10	$1/nC^{mn}$

n the number of cities in a TSP instance,

m the number of ants,

ρ a pheromone evaporation rate,

τ_0 initial pheromone for all trails, $\forall(i, j) \tau_{ij} = \tau_0 = m/C^{mn}$,

C^{mn} the length of a tour generated by the nearest-neighbor heuristic.

Ant Colony Optimization (ACO)

ACO元啟發式演算法(The ACO metaheuristic) (承上一頁)

PheromoneUpdate(): 費洛蒙更新的目的如下。若有好的解(good or promising solutions)時,增加其相關的費洛蒙值。相反地,若遇壞的解(bad solutions)時,減少其相關的費洛蒙值。通常,此費洛蒙更新策略的實現如下。藉由選擇一良好解 s_{ch} (假設螞蟻 k 的完整路徑被選中),然後增加其相關費洛蒙水平(increasing the pheromone levels associated with chosen good solution)予某一定值 $\Delta\tau_{ij}^{(k)}$ 。並且,藉由費洛蒙蒸發,降低所有路徑的費洛蒙值(decreasing all the pheromone values through pheromone evaporation)。

- 螞蟻 k 於路徑 (i, j) 之費洛蒙值 τ_{ij} 的更新公式,如式1。
- 對於TSP,全群螞蟻於路徑 (i, j) 之總體費洛蒙更新的一個例子,參見式2及式3。
- 需要費洛蒙蒸發(Pheromone evaporation)以避免演算法過快收斂到次佳區域。費洛蒙蒸發實現了一種有用的遺忘形式(A useful form of forgetting),有利於搜尋空間中的未知領域之探索。
- 一般來說,蟻群疊代早期找到的好的解被用來更新費洛蒙,以增加後續螞蟻在搜索空間的有希望的區域中搜索的機率。不同的ACO演算法,例如, Ant Colony System (ACS)[1]或 MAX-MIN Ant System (MMAS)[2]其費洛蒙更新方式有所不同。原則上,演算法使用當前疊代最佳解方式(The iteration-best solution, 即在當前疊代中找到的最佳解),或迄今為止最佳解方式(The best-so-far solution, 即從演算法運行開始至今為止找到的最佳解),來更新費洛蒙(有時會使用螞蟻發現的幾種解的組合)。迄今為止最佳解方式更新導致更快的收斂,而當前疊代最佳解方式更新允許搜尋更加多樣化[3]。

$$\tau_{ij} \leftarrow \begin{cases} (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij}^{(k)} & \text{if } \tau_{ij} \in s_{ch} \\ (1 - \rho)\tau_{ij} & \text{otherwise} \end{cases} \quad (1)$$

where

ρ 費洛蒙蒸發率
 s_{ch} a pheromone evaporation rate, $\rho \in (0, 1]$,
 a chosen good complete solution,
 $\Delta\tau_{ij}^{(k)}$ the amount of pheromone deposited on edge (i, j) by the k th ant,
 τ_{ij} the pheromone value associated with component c_{ij} , $c_{ij} \in N(s^p)$.

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \sum_{k=1}^m \Delta\tau_{ij}^{(k)} \quad (2)$$

$$\Delta\tau_{ij}^{(k)} = \begin{cases} Q/L^{(k)} & \text{if ant } k \text{ uses edge}(i, j) \text{ in its tour } T^{(k)} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where

$T^{(k)}$ the tour by ant k ,
 Q a positive constant,
 $L^{(k)}$ the length of the tour $T^{(k)}$ by ant k ,
 m the size of an ant colony.

- [1] Dorigo, M., Gambardella, L.M. (1997). Ant Colony System: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66.
- [2] Stutzle, T., Hoos, H.H. (2000). MAX-MIN Ant System. Future Generation Computer Systems 16 (8), 889–914.
- [3] Stutzle, T., Dorigo, M. (1999). ACO algorithms for the traveling salesman problem. In: Miettinen, K., Ma“kela”, M.M., Neittaanma“ki, P., Pe’riaux, J. (Eds.), Evolutionary Algorithms in Engineering and Computer Science. John Wiley and Sons, Chichester, UK, pp. 163–183.

Ant Colony Optimization (ACO)

ACO元啟發式演算法(The ACO metaheuristic) (承上一頁)

DaemonActions(): DaemonActions用於實作單一螞蟻無法執行的集中操作。DaemonActions 操作的範例是啟動局部最佳化程序(the activation of a local optimization procedure)或收集全局信息(the collection of global information),這些全域資訊可用於決定是否有必要沉積額外的信息素(費洛蒙)以從非局部角度偏置搜尋過程(to bias the search process from a nonlocal perspective)。

- 例如, Daemon(常駐程式、守護程式)[1]可以觀察蟻群中每隻螞蟻找到的路徑,並選擇一隻或幾隻螞蟻(例如,在演算法疊代中建立最佳解的螞蟻),然後允許它們在解組件(即,可用連接路徑(或稱邊))上沉積額外的信息素(費洛蒙)。

[1]Daemon 指在一個多工的電腦作業系統中的一種常駐程式(或稱守護程式),在後台執行,而不由使用者直接互動控制的電腦程式。此類程式會被以行程(Process,指電腦中正在執行的程式)的形式初始化。常駐程式的名稱通常以字母d結尾,以指明這個行程實際是常駐程式,並與普通的電腦程式區分開來。例如, syslogd就是指管理系統紀錄檔的常駐程式、shd是接收傳入SSH連接的常駐程式。1963年,麻省理工學院MAC專案的程式設計師費南多·柯巴托(Fernando José "Corby" Corbató, 1926–2019)發明「常駐程式」這個概念。根據他的說法,他的團隊最早採用daemon這個概念,其靈感來源於麥克斯韋妖(Maxwell's demon)——一種物理學和熱力學中虛構的媒介,能幫助排列分子。他對此表示:「我們別出心裁地開始使用daemon這個詞來描述後台行程,它們不知疲倦地處理系統中的雜務。」Unix系統繼承了這個術語。作為一種在後台起作用的超自然存在,麥克斯韋妖與古希臘神話中的代蒙一致。daemon是單詞「demon」較早的拼寫形式,源於希臘語δαίμων。伊維·尼梅斯(Evi Nemeth, 1940 – missing-at-sea June or July, 2013)在Unix系統管理手冊中對常駐程式有如下闡釋:許多人將「daemon」與「demon」這兩個詞等同,藉此暗示UNIX與陰間的某種邪惡聯絡。這是一種極壞的誤解。「Daemon」事實上是「demon」另一種早得多的寫法;daemon並無善或惡的傾向,相反,它定義一個人的品質或性格。古希臘的「個人的代蒙」(A “personal daemon”)概念類似於現代的「守護天使」(A “guardian angel”)概念——快樂即是得到友好靈魂幫助或保護的狀態。通常,UNIX系統看起來充斥著許多守護天使(Daemons)和惡魔(Demons)。(Wiki)

“To Do More” in the future.

- 1) Differential Evolution (DE, 差分進化最佳化演算法)
- 2) Grey Wolf Optimizer (GWO, 狼群最佳化演算法)