國立臺灣大學工學院工業工程學研究所

柔性計算與應用-期末報告(2020)

Algorithm of Beetle Swarm Antennae Search (BSAS), Beetle Swarm Optimization (BSO) for Continuous Optimization Problems

工業工程學研究所碩士班一年級 R08546002

劉晏誠

指導教授：楊烽正 老師

中華民國 110 年 1 月 18 日

目錄

圖目錄

表目錄

一、 緒論

許多演算方法的發展是來自於對自然界現象的觀察，藉由觀察獲得的靈感作為演算法的理論基礎。在自然界中有許多群體活動的生物，為了覓食、遷移和防衛之便等，成立其「社會系統」，其是由簡單的個體及其組成的群體產生互動行為所構成，許多科學家開始探討這些生物界社會系統的組成結構、資訊溝通和行為模式等。舉例來說，Holland 模擬地球上生物進化規律提出了遺傳算法 (Genetic Algorithm)，其與眾不同的搜索機制引起了人們對啟發式算法的興趣，因而掀起研究啟發式算法的熱潮，像是模擬退火算法 (Simulated Annealing Algorithm)、擬螞蟻覓食的蟻群算法(Ant Algorithms)、鳥群及魚群的移動陣形等演算法，後面幾種以模仿其中自然現象之機制而得到的方法，並透過群體的力量所展現出來的「智慧」，可稱之為群體智慧 (swarm intelligence, SI)，群體智慧便是藉由個體的貢獻使群體得到最大利益的一種方式。以上幾種啟發式算法都有一個共同的特點：從隨機的可行初始解出發，再用迭代改進的策略去逼近問題的最優解。

二、 演算法介紹及流程架構

本報告使用是以天牛鬚搜尋 (Beetle Antennae Search, BAS)演算法為基底的兩個方法，分別為天牛群最佳化 (Beetle Swarm Optimization, BSO)及天牛群鬚搜尋 (Beetle Swarm Antennae Search, BSAS)演算法，雖然此兩種方法皆為延伸 BAS 所得出的新方法，但在操作上仍是有不少的相異之處，而上述三種方法都是用來解決連續優化問題。以下將先介紹 BAS 演算法，接著再介紹 BSO 及 BSAS。

1. BAS

Jiang and Li (2017)提出來的 BAS 概念是模仿天牛的覓食機制，當天牛覓食時，天牛並不知道食物在哪裡，而是根據食物氣味的強弱來覓食，其判斷食物氣味的強弱是依靠它的兩隻長觸角 (天牛鬚)，如果左邊觸角收到的氣味強度比右邊大，那下一步天牛就往左飛，否則就往右飛，藉由這一原理找到食物。BAS 的解代理人為天牛的位置，其流程架構如下:

(1) 設定初始天牛兩觸角的直徑長度d^0、往下一步移動的距離δ^0及天牛的初始位置x^0，藉由初始位置得到當前最佳解，直徑長度與移動距離成正比，越大隻的天牛(直徑大)移動的距離會越大。

(2) 產生隨機方向向量，並將其標準化。(k 表示維度)

$$\vec{b} = \frac{rnd(k,1)}{\|rnd(k,1)\|} \tag{1}$$

(3) 使用以下公式計算左、右兩觸角的位置。

$$x_r = x^t + d^t\vec{b}$$
$$x_l = x^t - d^t\vec{b} \tag{2}$$

(4) 將左右兩觸角位置帶入目標函式，判斷下一步天牛要往右飛或左飛，並使用以下公式更新天牛位置，並將其帶入目標式與當前最佳解比較。

$$x^{t+1} = x^t + \delta^t\vec{b}\ \text{sign}(f(x_r) - f(x_l)) \tag{3}$$

(5) 更新天牛兩觸角的直徑長度及往下一步的移動距離。

(d_0為一很小的值，避免觸角直徑為 0，等同於剩一隻觸角)

$$d^{t+1} = \eta_d d^t + d_0 \tag{4}$$
$$\delta^{t+1} = \eta_t \delta^t \tag{5}$$

(6) 重複 2~5 步驟，直到到達迭代次數的停止條件。

Algorithm 1: BAS algorithm for global minimum searching

Input: Establish an objective function $f(x^t)$, where variable $x^t = [x_1, \cdots, x_i]^{\mathrm{T}}$, initialize the parameters x^0, d^0, δ^0.
Output: x_{bst}, f_{bst}.
while *($t < T_{max}$) or (stop criterion)* **do**
  Generate the direction vector unit $\vec{b}$ according to (1);
  Search in variable space with two kinds of antennae according to (2);
  Update the state variable x^t according to (3);
  if $f(x^t) < f_{bst}$ **then**
    $f_{\text{bst}} = f(x^t)$, $x_{\text{bst}} = x^t$.
  Update sensing diameter d and step size δ with decreasing functions (4) and (5) respectively, which could be further studied by the designers.
return x_{bst}, f_{bst}.

圖 1 BAS pseudo code

2. BSAS

BAS 的做法主要有兩大缺點，第一點為在每一次的迭代中僅會產生一個隨機方向的向量，這樣使其無法保證天牛移動到下一個位置後會得到更好的目標值，第二點為無論有無優於當前最佳解皆會更新兩觸角直徑的長度及往下一步的距離長度，有可能會產生不必要的更新，以上兩點會導致目標值落入區域最佳解而無法跳出。因此，Jiang and Li (2018)提出了 BSAS 演算法改善以上的問題，針對第一點它改採用 k 個天牛來取代一隻天牛，生成 k 個隨機方向，而第二點則是針對更新兩觸角直徑及移動距離訂定一套機制，會在演算法架構做說明。BSAS 的解代理人為天牛的位置，其流程架構大致與 BAS 相似，僅做一些修正，如下:

(1) 設定初始天牛鬚兩觸角的直徑長度d^0、往下一步移動的距離δ^0、天牛的初始位置x^0、k 隻天牛及 k 隻天牛會錯過更好 Solutions 的機率p_δ (值越大表示使用者認為錯過更好 Solutions 的機率越高，反之亦然)，並藉由初始位置x^0得到當前最佳解。

(2) 產生 k 個隨機方向向量，並將其標準化。$[\vec{b}_1, \vec{b}_2, \dots, \vec{b}_k]$

(3) 使用以下公式計算左、右兩觸角的位置。BSAS 的 k 隻天牛是每隻天牛皆以同一個點為基點，加減一隨機方向向量後得出一對觸角，有 k 隻天牛就有 k 對觸角。

$$\begin{aligned} x_r &= x^t + d^t\vec{b} \\ x_l &= x^t - d^t\vec{b} \end{aligned} \tag{1}$$

(4) 將各個左右兩觸角位置帶入目標函式，判斷該隻下一步天牛要往右飛或左飛，並使用以下公式更新天牛位置。有 k 隻天牛就會有 k 個新的天牛位置。

$$x^{t+1} = x^t + \delta^t\vec{b}\ \text{sign}(f(x_r) - f(x_l)) \tag{2}$$

(5) 將 k 個新的天牛位置帶入目標式與當前最佳解比較，如果優於當前最佳解，則選擇跳回至步驟(2)進行下一次迭代，並將x^t更新為最佳的x^*，否則(亦即 k 個新的天牛位置沒有任何一個優於當前最佳解)繼續步驟(6)。

(6) 隨機產生[0,1]之間的亂數，如果大於p_δ則保持當前的參數不變，跳回至步驟(2)進行下一次迭代，否則使用以下公式更新天牛兩觸角的直徑長度及往下一步的移動距離。(d_0及δ_0皆為很小的值，避免直徑及步伐為 0)

$$d^{t+1} = \eta_d d^t + d_0 \tag{3}$$

$$\delta^{t+1} = \eta_t \delta^t + \delta_0 \tag{4}$$

(7) 重複 2~6 步驟，直到到達迭代次數的停止條件。

Algorithm 1: BSAS algorithm for global minimum searching

Input: Establish an objective function $f(\boldsymbol{x}^t)$, where variable $\boldsymbol{x}^t = [x_1, \cdots, x_n]^{\mathrm{T}}$, initialize the parameters $\boldsymbol{x}^0, d^0, \delta^0, k, p_\delta$.

Output: x_{bst}, f_{bst}.

while $(t < T_{max})$ *or* $(\delta < \delta_{criterion})$ **do**

  Generate k direction vector units $[\overrightarrow{b_1}, \overrightarrow{b_2}, \cdots, \overrightarrow{b_k}]$;

  Search in variable space with two kinds of antennae according to (1) for k beetles ;

  if $min(f(\boldsymbol{x}_i^t)) < f_{bst}, where\ i \in [1, 2, \cdots, k]$ **then**

    $f_{\mathrm{bst}} = f(\boldsymbol{x}^t)$;

    $\boldsymbol{x}_{\mathrm{bst}} = argmin(f(\boldsymbol{x}_i^t))$;

    $\boldsymbol{x}^{t+1} = \boldsymbol{x}_{\mathrm{bst}}$.

  else

    if $rnd(1) > p_\delta$ **then**

      Update sensing diameter d and step size δ with decreasing functions (3) and (4) respectively;

    else

      Parameters δ and d remain the same.

return x_{bst}, f_{bst}.

圖 2 BSAS pseudo code

3. BSO

在 BSAS 提出來一個月後，Tiantian and Long (2018)提出了 BSO，此方法為結合 BAS 及 PSO 的演算法，有別於 BSAS 地方在於其是分配 k 隻天牛不同的初始位置，而不是像 BSAS 以一個點做為基點，再使用 k 個不同的隨機方向向量得出天牛位置，這樣會導致 k 個不同位置的解過於集中，可能落入局部最佳解，而 BSO 的做法能讓 k 隻天牛在尋找最佳解時能有廣泛的選擇，此外，BSO 使用到更多的參數，更新參數的方式也與 BSAS 不同，且因 BSO 是 PSO 的延伸，故有許多地方與 PSO 雷同。BSO 的解代理人為天牛的位置，其流程架構綜合了 PSO 及 BAS，如下:

(1) 設定初始天牛兩觸角的直徑長度d^0、往下一步移動的距離δ^0及天牛的初始位置x_i^0, where $i = 1,2,\dots,k$、初始速度v_i^0, where $i = 1,2,\dots,k$，藉由初始位置得到 k 隻天牛的當前最佳解，並記錄每隻天牛初始位置的解。

(2) 產生隨機方向向量，並將其標準化。(k 表示維度)

$$\vec{b} = \frac{rnd(k,1)}{\|rnd(k,1)\|} \tag{1}$$

(3) 使用以下公式計算左、右兩觸角的位置。

$$\begin{aligned} x_r &= \mathrm{x}_i^t + \frac{d^0\vec{b}}{2}, i = 1,2,\dots,k \\ x_l &= \mathrm{x}_i^t - \frac{d^0\vec{b}}{2}, i = 1,2,\dots,k \end{aligned} \tag{2}$$

(4) 使用以下公式計算慣性權重，此權重是用來決定更新速度時要考慮多少前一次迭代的速度。(K 為迭代次數，k 為第 k 次迭代)

$$w = w_{max} - \frac{w_{max} - w_{min}}{K} \times k \tag{8}$$

(5) 使用以下公式更新觸角直徑。(c_2為一常數)

$$d^t = \delta^t / c_2 \tag{3}$$

(6) 步驟(6)到步驟(9)都是針對每一個解代理人所執行的步驟。使用以下公式更新兩觸角位置，並帶入目標式看下一步要往哪飛。

$$\begin{aligned} X_r^{t+1} &= X_r^t + V_i^t \times \frac{d}{2} \\ X_l^{t+1} &= X_l^t - V_i^t \times \frac{d}{2} \end{aligned} \tag{10}$$

(7) 使用以下公式更新 incremental function ξ，針對 BAS 演算法的參數。

$$\xi_i^{t+1} = \delta^t \times V_i^t \times \ \mathrm{sign}(f(x_r) - f(x_l)) \tag{9}$$

(8) 使用以下公式更新速度 V，針對 PSO 演算法的參數。(P_i^t是各天牛最佳解，P_g^t是全域最佳解)

$$V_i^{t+1} = wV_i^t + c_1 r_1 (P_i^t - X_i^t) + c_2 r_2 \left(P_g^t - X_g^t\right) \tag{7}$$

(9) 使用以下公式更新解代理人的位置。

$$X_i^{t+1} = X_i^t + \lambda V_i^t + (1-\lambda)\xi_i^t \tag{6}$$

(10) 將 k 個新的天牛位置帶入目標式與當前最佳解比較，並記錄每個天牛的當次解，比較完後使用以下公式更新移動距離δ。

$$\delta^{t+1} = \eta_t \delta^t + \delta_0 \quad (4)$$

(11) 重複 4~10 步驟，直到到達迭代次數的停止條件。

Procedure:

Initialize the swarm $X_i (i = 1,2,...,n)$
Initialize population speed v
Set step size δ ,speed boundary $v_{\max}$ and $v_{\min}$,population size *sizepop* and maximum number of iterations K
Calculate the fitness of each search agent
While($k < K$)
  Set inertia weight ω using Eq.(8)
  Update d using Eq.(5)
  for each search agent
    Calculate $f(X_{rs})$ and $f(X_{ls})$ using Eq.(10)
    Update the incremental function ξ by the Eq.(9)
    Update the speed formula V by the Eq.(7)
    Update the position of the current search agent by the Eq.(6)
  end for
  Calculate the fitness of each search agent $f(x)$
  Record and store the location of each search agent

  for each search agent
    if $f(x) < f_{pbest}$
      $f_{pbest} = f(x)$
    end if
    if $f(x) < f_{gbest}$
      $f_{gbest} = f(x)$
    end if
  end for
  Update x^* if there is a better solution
  Update step factor δ by the Eq.(4)
end while
Return x_{best} , f_{best}

圖 2 BSO pseudo code

4. 演算法參數設定

表 1 為 BAS、BSAS、BSO 三種方法的參數初始設定，定義上述皆有提及。

表 1 參數設定

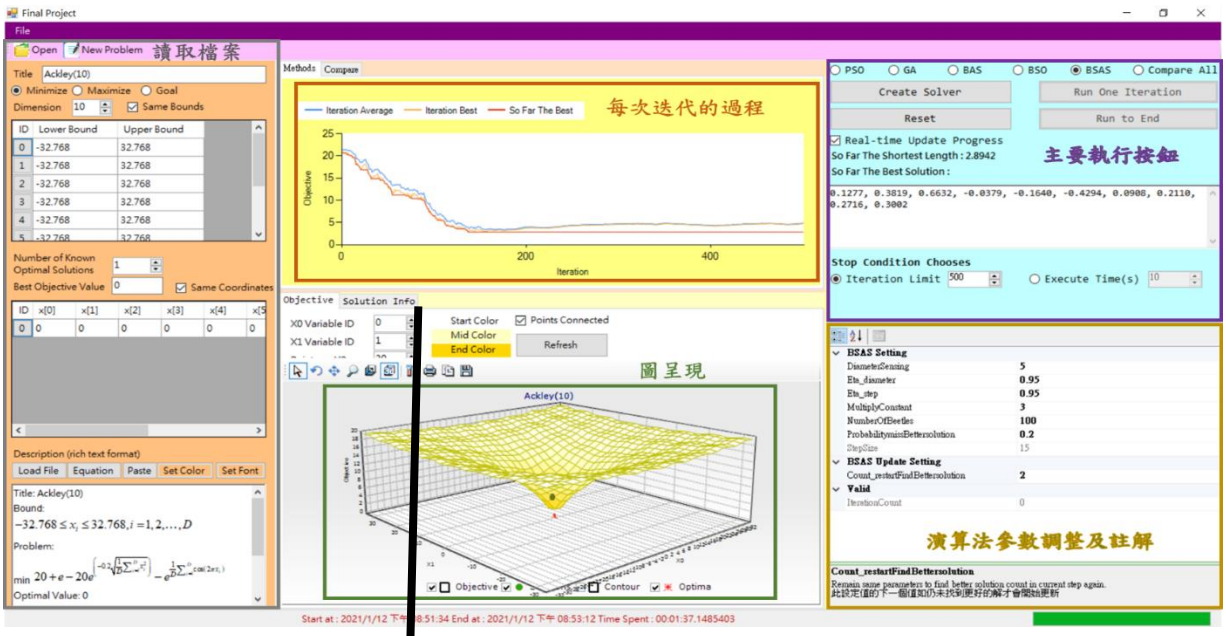
	# of Beetles	d^0	c_2	δ^0	η_d	η_t	p_δ	λ
BAS	1	5	3	5*3	0.95	0.95		
BSAS	100	5	3	5*3	0.95	0.95	0.2	
BSO	100	5	3	5*3	0.95	0.95		0.4

三、 程式介面介紹

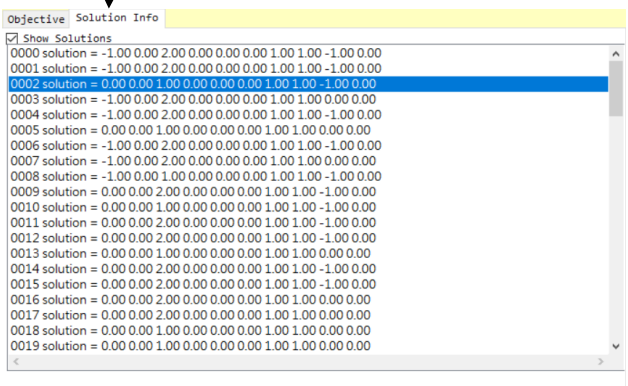
本章節將會介紹使用者介面 (User Interface, UI)，主要分為五種方法 (PSO、GA、BAS、BSO、BSAS)及方法比較的介面，並對其如何呈現做說明。

1. PSO、GA、BAS、BSO、BSAS 程式介面介紹

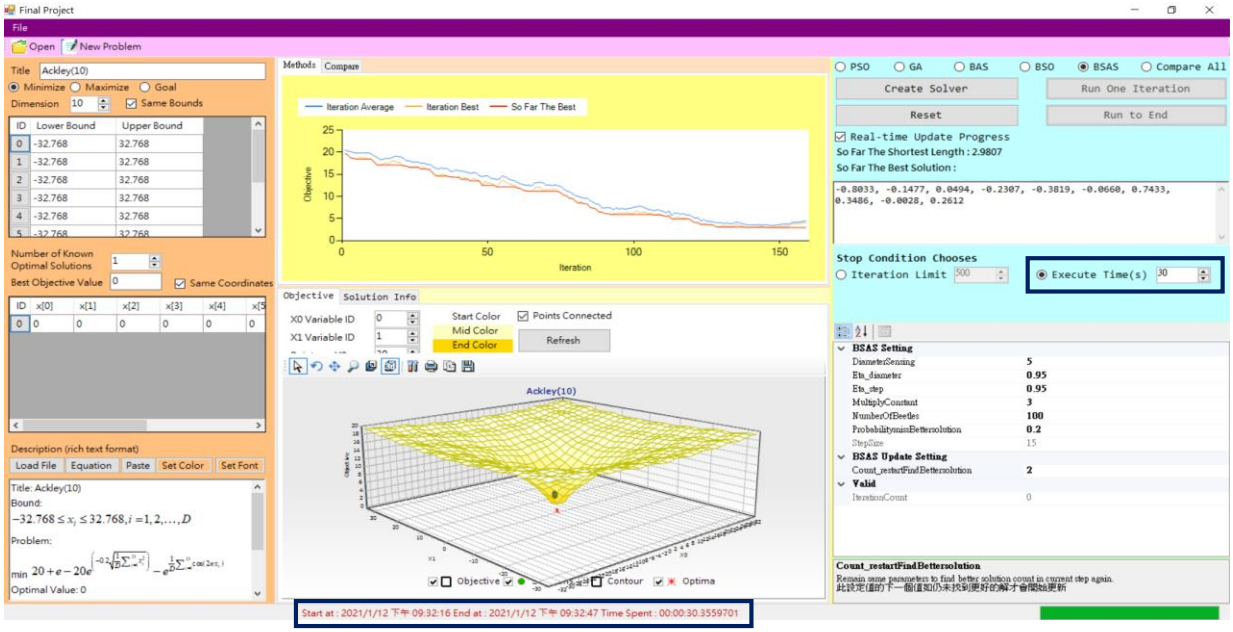
見圖 3-a 可知道此五種演算法有基本的讀檔功能、每次迭代的過程、圖呈現、各解代理人的當次迭代的解(圖 3-b)、主要執行按鈕、參數調整及註解以及最下面的時間花費和 progressBar，其中學生將迭代次數限制從 Grid 中獨立出來，以便後續方法比較使用，此外，學生新增執行時間的限制，能讓使用者設定自己想要的執行時間，時間以秒計算(圖 3-c)。



(a) 基本介面介紹



(b) 各解代理人的解



(c) 使用時間停止條件執行

圖 3 方法程式介面介紹

2. 方法比較之程式介面介紹

圖 4 為 Cross Validation 設定為 1 時的結果，其主要呈現的內容有各方法比較之每次迭代過程、各方法最佳解、選擇解代理人的個數、CV 次數及方法比較後得出最好的方法、解及目標值。

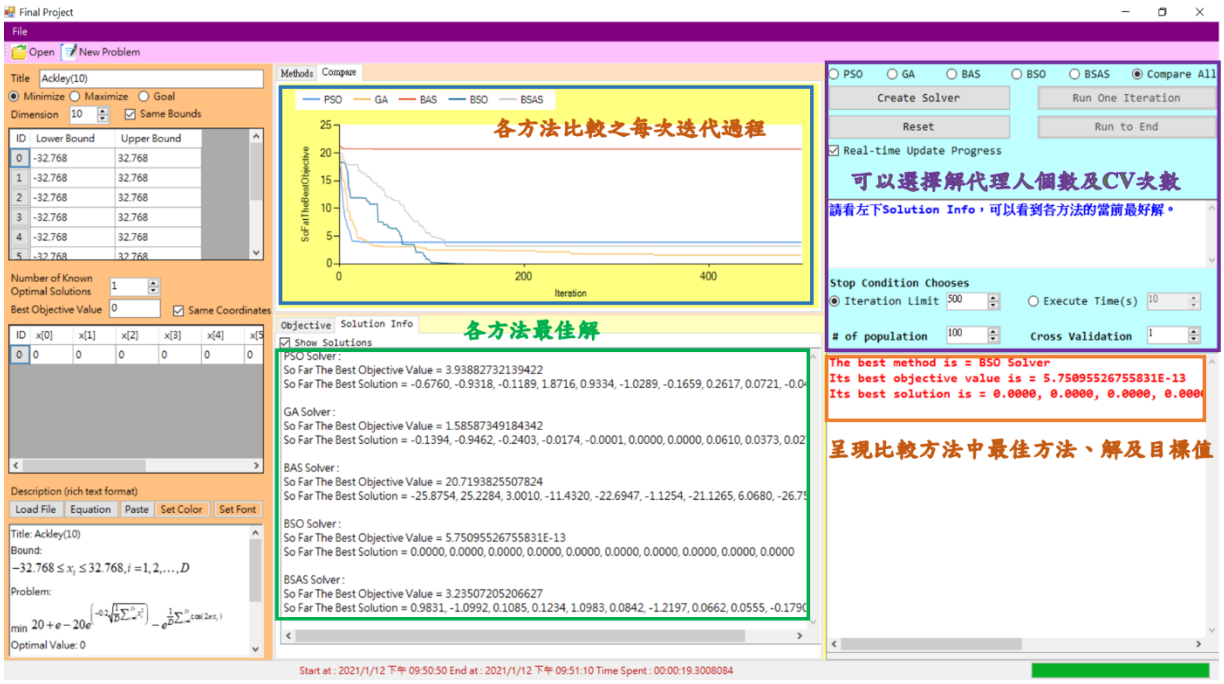


圖 4 方法比較之程式介面介紹_CV = 1

圖 5 為 Cross Validation 設定為 30 時的結果，其主要呈現的內容有各方法每次 Cross Validation 的最佳值、各方法在所有 Cross Validation 中之最佳解、目標值、平均數及標準差、以及方法比較後得出最好的方法、解及目標值和方法比較中平均數最好的方法及其值。此外，程式介面亦設有 New Problem 供使用者新增想要觀察的問題。

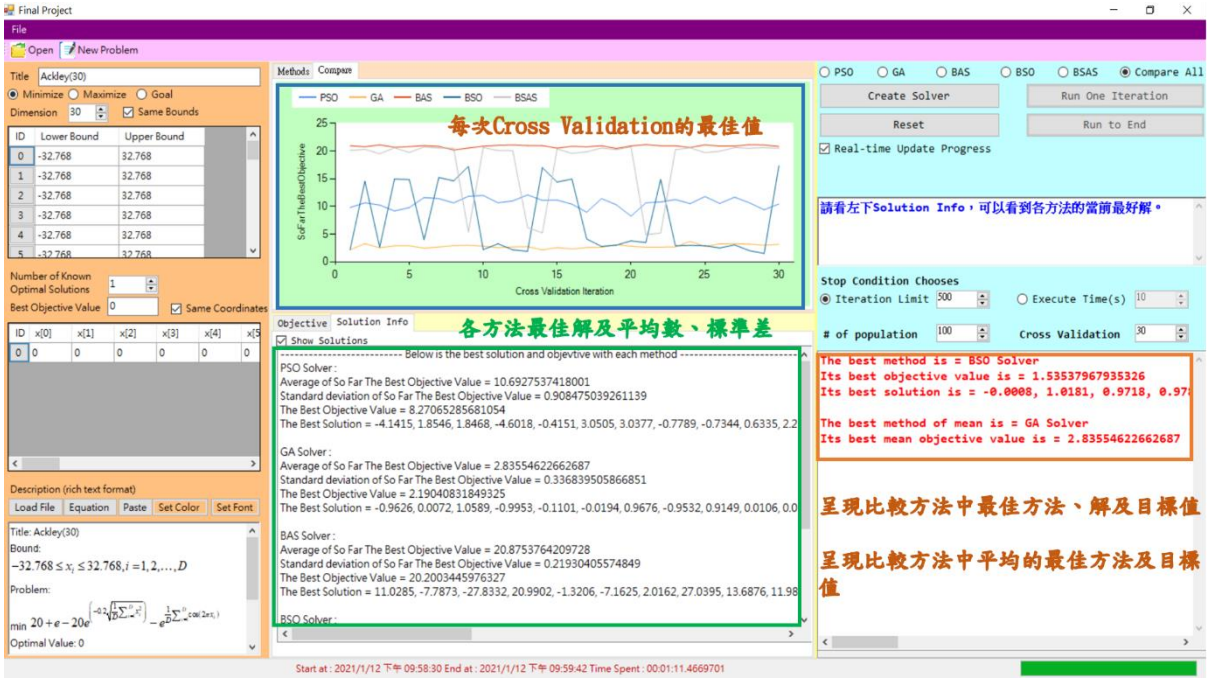


圖 5 方法比較之程式介面介紹_CV = 30

四、 案例實作及方法比較

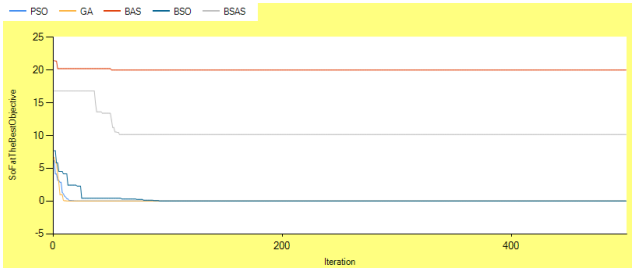
本報告將以 Ackley、Girewank、Schwefel、及學生取自文獻的目標式 Function9 (低維為 3D)等四種標竿問題作探討 (各標竿問題的公式定義如表 2，皆是用來解決最小化問題)，將其分為 2D (低維)、10D (中維)、30D (高維)去比較在不同程度的維度中，各方法的表現效果。各方法比較又細分為將 Cross Validation 設為 1 去看在給定的迭代次數下各方法的變化，以及將 Cross Validation 設為 30 去看各方法的穩定程度，查看方式為藉由平均數、標準差。以下使用的各方法皆設 Iteration Limit 為 500、解代理人除了 BAS 外，其他皆設為 100，而 BAS、BSAS、BSO 此三種方法的參數設定如表 1。

表 2 標竿問題之公式

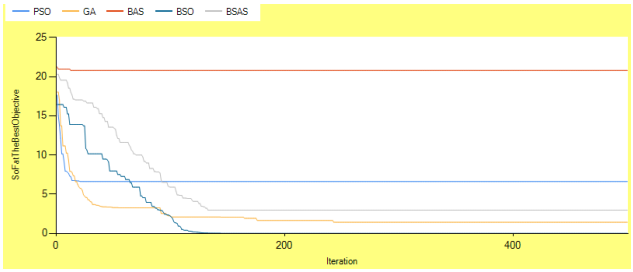
Benchmark Problem	Formula (all of them are minimum problem)	Bound
Ackley	$20+e-20e^{\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)}-e^{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)}$	$-32.768 \le x_i \le 32.768, i=1,2,\ldots,D$
Function9	$\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10)$	$-5.12 \le x_i \le 5.12, i=1,2,\ldots,D$
Rastrigin	$\sum_{i=1}^{D}\left(x_i^2-10\cos\left(2\pi x_i\right)+10\right)$	$-15 \le x_i \le 15, i=1,2,\ldots,D$
Schwefel	$418.9829D-\sum_{i=1}^{D}x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	$-500 \le x_i \le 500, i=1,2,\ldots,D$

1. 各方法比較_CV=1

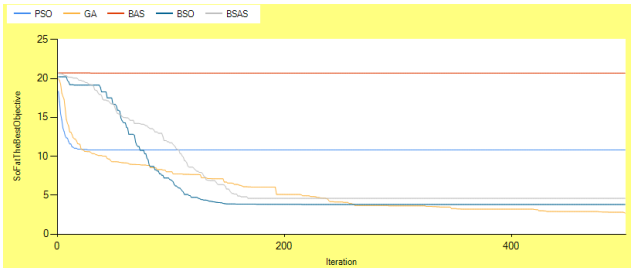
藉由 CV=1 去觀察在不同標竿問題下各方法的呈現結果，圖 6 到圖 9 為各標竿問題之各方法比較的圖形呈現，而表 3 為各方法在不同標竿問題下的 SoFarTheBestObj。從圖可以看出 BAS 無論在何種標竿問題及維度上，都是表現最差，且幾乎不更新最佳解，而 BSAS 及 BSO 都是會隨著迭代次數的增加而更新最佳解，且 BSO 的收斂速度雖然較 PSO 慢但結果較好，而其收斂速度較 GA 快但解較差。就表 3 的結果來說，排名為 GA > BSO > PSO > BSAS > BAS。



(a) 2D

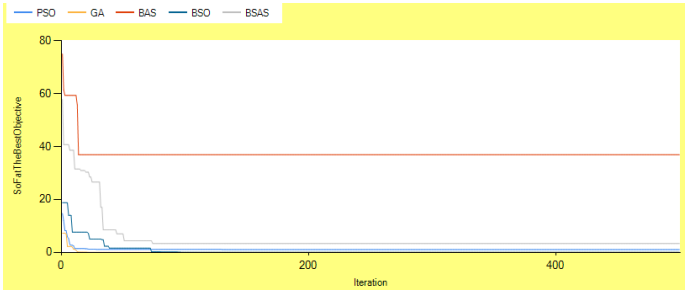


(b) 10D

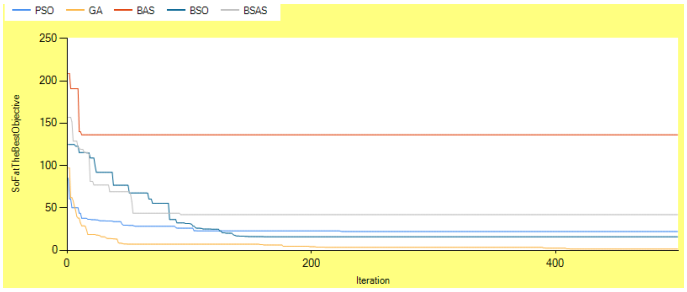


(c) 30D

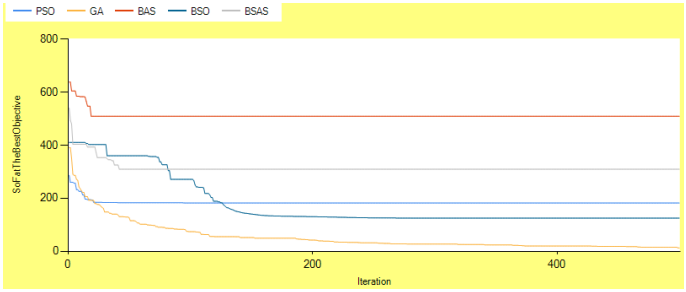
圖 6 Ackely 執行結果 CV=1 – (a) 2D, (b) 10D, (c) 30D



(a) 3D

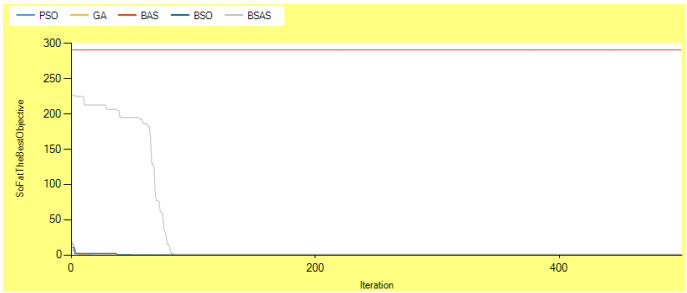


(b) 10D

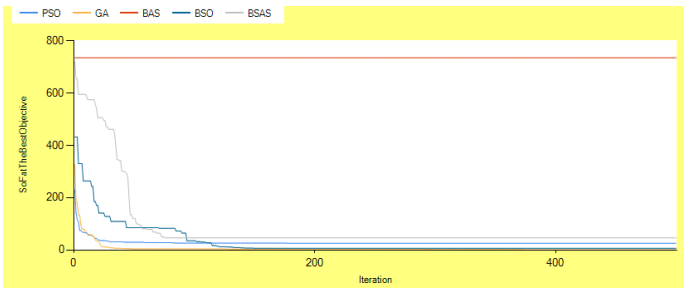


(c) 30D

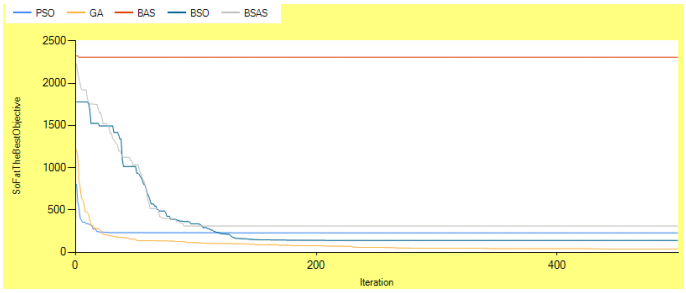
圖 7 Function9 執行結果 CV=1 – (a) 3D, (b) 10D, (c) 30D



(a) 2D

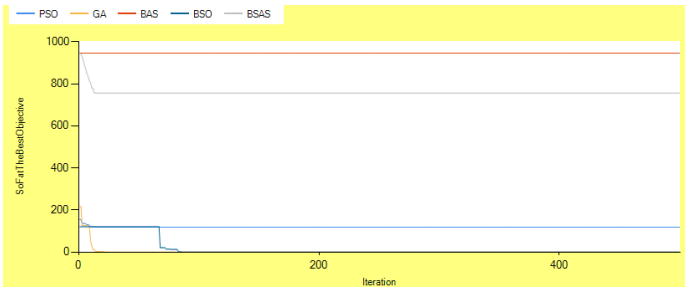


(b) 10D

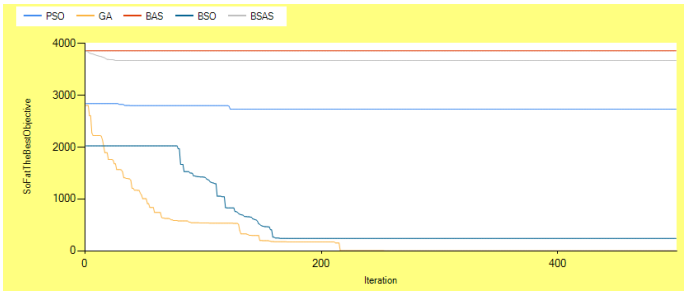


(c) 30D

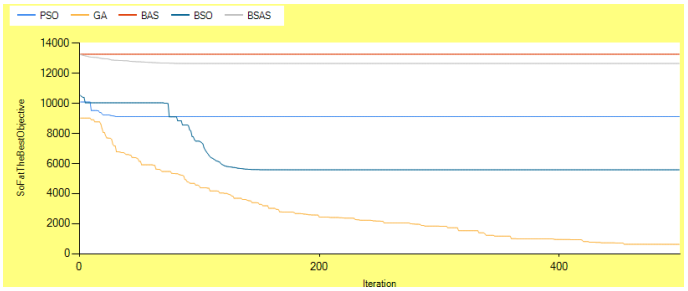
圖 8 Rastrigin 執行結果 CV=1 – (a) 2D, (b) 10D, (c) 30D



(a) 2D



(b) 10D



(c) 30D

圖 9 Schwefel 執行結果 CV=1 – (a) 2D, (b) 10D, (c) 30D

表 3 各方法比較_CV=1 之結果

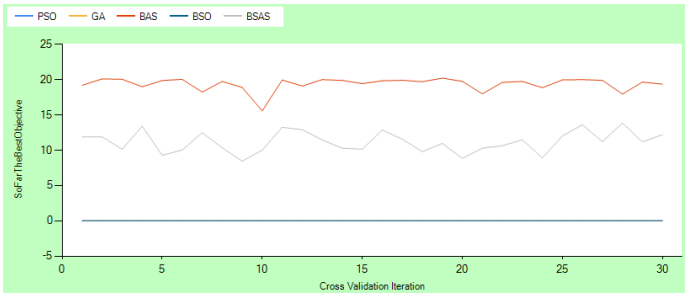
		PSO	GA	BAS	BSO	BSAS
Ackely	2D	1.021E-14	-4.44E-16	20.0074	-4.44E-16	10.1589
	10D	6.5995	1.4032	20.7823	3.15E-14	2.9498
	30D	10.8212	2.6992	20.6866	3.8096	4.5887
Function9	3D	1.0298	0	36.9105	0	3.2929
	10D	22.1197	1.3913	136.3231	15.9193	42.1022
	30D	181.8830	12.2830	509.5264	124.8737	309.4464
Rastrigin	2D	0	0	290.9886	0	1.029
	10D	36.0.67	2.4331	734.1159	7.9597	55.7589
	30D	229.0416	37.5862	2308.3309	138.9116	309.659
Schwefel	2D	118.4569	2.54E-05	945.5281	2.54E-05	756.2428
	10D	2731.9412	0.6132	3857.9793	236.8768	3671.2761
	30D	9119.2345	616.9067	13274.8639	5581.2299	12653.7981

2. 各方法比較_CV=30

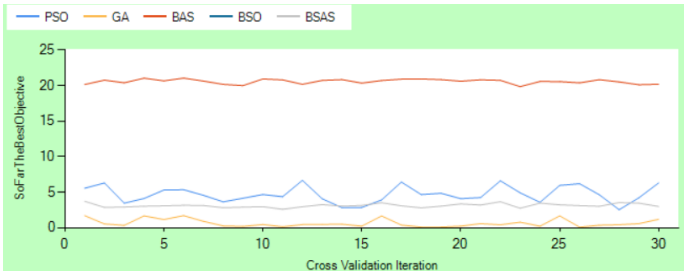
同樣將各方法 Iteration Limit 設為 500、解代理人除了 BAS 外，皆設為 100，並藉由 CV=30 去觀察在不同標竿問題下各方法的呈現結果，透過 CV 的設置去證實這幾種方法的穩定性，使結果得以令人更信服。圖 14 到圖 17 為各標竿問題之各方法比較的圖形呈現，而表 3 為各方法在不同標竿問題下的每次 Cross Validation 下最好的 SoFarTheBestObj 及 CV=30 下 SoFarTheBestObj 的平均數、標準差。

從圖 14 來看 Ackely 的標竿問題，在 2D 下 PSO、GA、BSO 這三種方法在每次的 CV 下達到最佳解，而 BAS 及 BSAS 皆未能達到全域最佳解，其中以 BAS 最差，接著從 10D 來看可以發現在不同的 CV 下 PSO、GA、BSO 都開始產生波動，而原本表現第二差的 BSAS 則是優於 PSO，表現最差的仍是 BAS，最後再來看 30D 高維下的結果，這裡可以清楚觀察到 BSO、BSAS 兩者的波動又較 10D 時來的更大，而 GA 則是在 CV=30 下變異最小的方法，從表 3 來看這一標竿問題，可以清楚知道 BSO 在 2D、10D 表現最好，且雖然在 30D 是 GA 的平均數表現最好，但在找最佳解上市 BSO 能找到更好的解；而圖 15 到圖 17 則是呈現一面倒的情形，在 2D 都是 BSO 表現最好，到了 10D、30D 則都是 GA 表現最好，其中還可以從圖中觀察到表現最差的 BAS 有時甚至優於 BSAS，原因在於一開始隨機設置的初始位置，因 BAS 在每次的迭代僅有一隻天牛，故在搜尋時能移動的方向僅有一個，因此如果一開始的初始位置就與全域最佳解接近，則 BAS 最後的解也就會非常靠近最佳解。

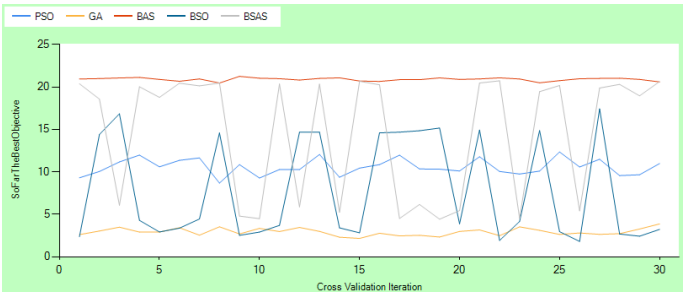
綜觀來看，可以得出 BAS 無論在何種標竿問題及維度上仍然是表現最差，而 BSAS 及 BSO 在 CV=30 下可以看出其波動很大，時好時壞，但 BSO 能找出甚至優於 GA 的解，而 GA 及 PSO 在每次 CV 下的表現相對穩定，其中又以 GA 得出的解最好。就表 33 的結果來說，排名同樣為 GA > BSO > PSO > BSAS > BAS。



(a) 2D

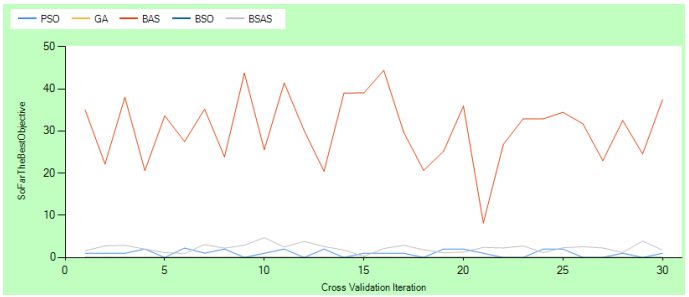


(b) 10D

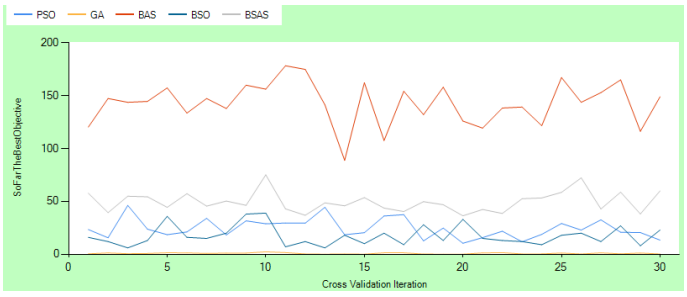


(c) 30D

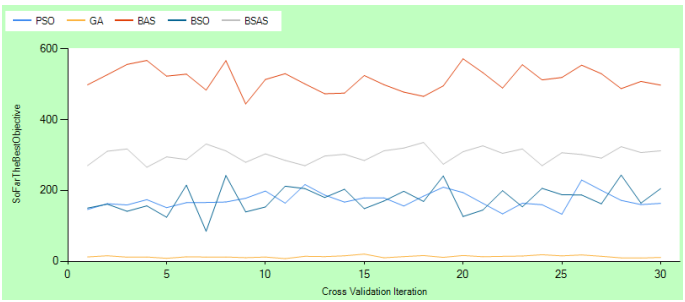
圖 10 Ackely 執行結果 CV=30 – (a) 2D, (b) 10D, (c) 30D



(a) 3D

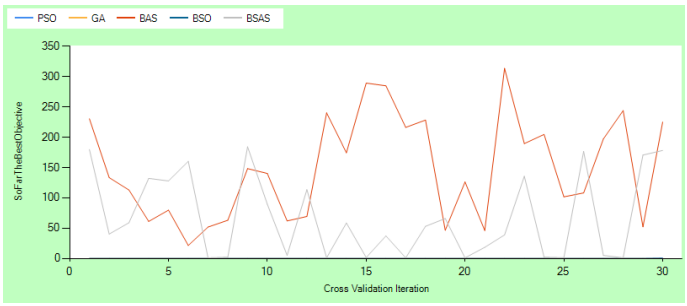


(b) 10D

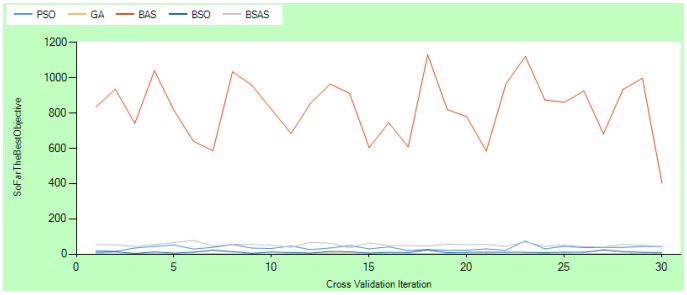


(c) 30D

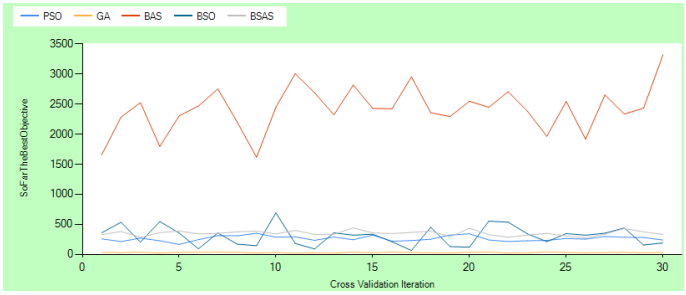
圖 11 Function9 執行結果 CV=30 – (a) 3D, (b) 10D, (c) 30D



(a) 2D

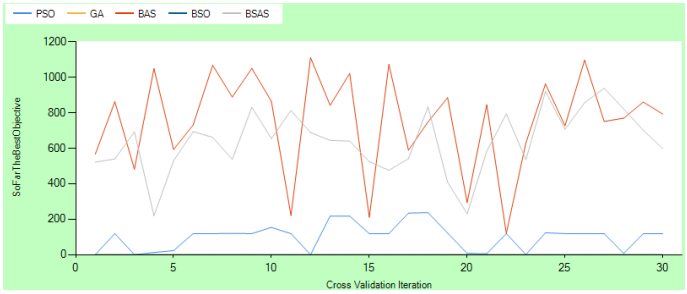


(b) 10D

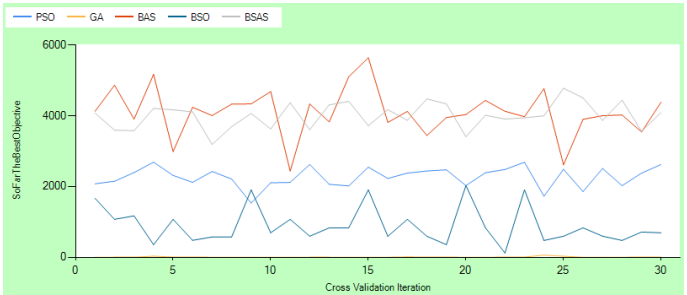


(c) 30D

圖 12 Rastrigin 執行結果 CV=30 – (a) 2D, (b) 10D, (c) 30D



(a) 2D



(b) 10D

(c) 30D

圖 13 Schwefel 執行結果 CV=30 – (a) 2D, (b) 10D, (c) 30D

表 4 各方法比較_CV=30 之結果 - (a) Ackely & Function9 (b) Rastrigin & Schwefel

(a) Ackely & Function9

			PSO	GA	BAS	BSO	BSAS
Ackely	2D	Mean	1.05E-10	2.56E-06	19.3913	-4.44E-16	11.1851
		Std	5.45E-10	1.38E-05	0.9309	0	1.490
		Best	-4.44E-16	-4.44E-16	15.579	-4.44E-16	8.4489
	10D	Mean	4.8109	0.5973	20.6963	0.0385	3.6941
		Std	7.984	0.4434	0.3202	0.2074	4.4568
		Best	3.2722	0.0602	19.7705	3.11E-15	2.5989
	30D	Mean	10.5669	2.9153	20.8957	7.9094	14.5928
		Std	0.9348	0.4209	0.1836	5.9565	7.1977
		Best	8.6746	2.1557	20.4636	1.8002	4.4207
Function9	3D	Mean	0.9722	0.0019	30.5167	0	2.2189
		Std	0.8002	0.0039	8.0191	0	0.9401
		Best	1.77E-15	0	8.1671	0	0.2514
	10D	Mean	24.3423	0.7391	142.7730	17.3857	49.5181
		Std	9.0601	0.5678	19.9150	9.2406	9.4662
		Best	10.3492	0.0345	88.7945	5.9698	36.3088
	30D	Mean	172.9453	13.1603	513.7311	175.9694	300.8913
		Std	21.9199	2.9822	31.9697	36.9762	19.0829
		Best	133.0789	7.5881	444.5825	84.8890	265.7669

(b) Rastrigin & Schwefel

			PSO	GA	BAS	BSO	BSAS
Rastrigin	2D	Mean	0.0332	8.02E-10	148.7067	0	68.0336
		Std	0.1786	4.3213	83.6188	0	67.5938
		Best	0	0	21.3585	0	1.0092
	10D	Mean	35.6589	2.5044	828.4719	11.1436	52.2245
		Std	12.2054	1.3105	171.4318	4.8019	9.5046
		Best	15.6126	0.4105	402.8278	3.9798	35.9136
	30D	Mean	263.3152	29.7678	2419.047	304.7676	351.5660
		Std	42.2239	6.1768	372.2679	160.1292	42.3915
		Best	165.0287	19.0971	1614.734	63.7944	272.9127
Schwefel	2D	Mean	100.6269	0.0001	757.0797	2.55E-05	637.8798
		Std	71.3839	0.0005	270.5582	0	173.5709
		Best	3.52E-05	2.55E-05	120.2398	2.55E-05	219.2132
	10D	Mean	2269.847	6.2571	4105.837	888.3357	4005.609
		Std	279.5655	12.1219	672.1639	504.8425	364.9250
		Best	1532.856	0.0160	2436.484	118.4385	3188.595
	30D	Mean	9324.374	420.1990	12327.53	5022.552	12196.33
		Std	549.7999	174.9897	936.6405	1120.350	983.8486
		Best	8231.627	60.7323	10396.05	2693.144	10033.41

五、 結論與討論

透過上述各方法比較後能得出以下結論:

1. BAS 相較於其他演算法，其在各標竿問題下表現最差，原因在於 PSO、GA、BSAS、BSO 的解代理人數量是可以調整，而 BAS 至始至終僅只有一個。
2. 由 BAS 延伸出來的兩種方法 BSAS 及 BSO 確實較 BAS 還要好，而 BSAS 及 BSO 這兩種方法又以 BSO 表現更好，原因在於 BSO 是結合 BAS 及 PSO 的方法，保存了此兩種方法的優點，因此其表現是優於 PSO，而 BSAS 僅是對原本的 BAS 的初始點增加 k 個隨機向量，改善效果仍然有限。
3. BSAS 及 BSO 此兩種方法維度越高，每次所得到的最佳解越不穩定。
4. 在低維度的問題 PSO、GA、BSO 皆表現很好，但 BSO 相較於其他兩者又更穩定，而在中、高維度問題，則是 GA 的表現及穩定性較好，但如果要冒險找最佳解 BSO 仍可以一試。因此，BSO 適合使用於低維度的問題，而 GA 適合使用於中、高為度的問題。
5. 綜觀來看，GA > BSO > PSO > BSAS > BAS。

啟發式演算法所得到的結果不一定為最佳解，且結果會深受每種演算法參數的設定、限制或是其他因素影響，如本報告的方法僅有 Iteration Limit 及解代理人的條件是相同，其餘參數皆未做任何調整，因此想要得到最佳解，那 User 就必須根據當前問題執行大量且多方面的反覆測試，才能找出對於連續優化問題最適合的求解演算法 。

六、 參考文獻

[1] Jiang, X., & Li, S. Bas: Beetle antennae search algorithm for optimization problems. . *arXiv preprint arXiv:1807.10724*.

[2] Wang, J., & Chen, H. (2018). BSAS: Beetle swarm antennae search algorithm for optimization problems. *arXiv preprint arXiv:1807.10470*.

[3] Wang, T., Yang, L., & Liu, Q. (2018). Beetle swarm optimization algorithm: Theory and application. *arXiv preprint arXiv:1808.00206*.