柔性計算法與應用

Final Project

蜘蛛猴算法
Spider Monkey optimization algorithm

學生：鄭登元 R08546012

授課老師：楊烽正 教授

日期：2020/1/15

目錄

Introduction（介紹）

Swarm Intelligence是一種師法大自然的meta-heuristic的方法，而為了要去好好的了解一種行為是否是Swarm Intellegence的行為：Devision of labor 和 Self-organization 為兩個充份且必要的條件。

蜘蛛猴算法是透過模擬蜘蛛猴基於fission-fussion-social-structure進行覓食的社會行為而衍生的一種演算法，fission-fussion-social-structure透過將蜘蛛猴群體分為小組來減少蜘蛛猴個體彼此間的競爭覓食壓力，從而能夠更好的尋找食物。

蜘蛛猴通常是由母猴所領導整個族群，而當萬一母猴沒辦法找到足夠的食物時，他會將族群分成很多小組，而這些小組組數和食物的多寡呈現反向關係。

Methods and Problems（目標方法與問題）

Test Problem	Objective function	Search Range	Optimum Value	D	C	AE
Schwefel function 1.2	$f_1(x) = \sum_{i=1}^{D}(\sum_{j=1}^{i} x_j)^2$	$[-100, 100]$	0	30	UN	$1.0E-03$
Step function	$f_2(x) = \sum_{i=1}^{D}(\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]$	0	30	US	$1.0E-03$
Schwefel function	$f_3(x) = -\sum_{i=1}^{D}(x_i sin\sqrt{\lvert x_i \rvert})$	$[-500, 500]$	$-418.9829 \times D$	30	MS	$1.0E-03$
Rastrigin	$f_4(x) = 10D + \sum_{i=1}^{D}[x_i^2 - 10\cos(2\pi x_i)]$	$[-5.12, 5.12]$	0	30	MS	$1.0E-03$
Levy function 1	$f_5(x) = \frac{\pi}{D}[10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2(1 + 10\sin^2(\pi y_{i+1}))$ $+ (y_D - 1)^2] + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$, where $y_i = 1 + \frac{1}{4}(x_i + 1)$ and $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a; \\ 0, & -a \le x_i \le a; \\ k(-x_i - a)^m, & x_i < a. \end{cases}$	$[-50, 50]$	0	30	MN	$1.0E-03$

D Dimensions, *C* Characteristic, *U* Unimodal, *M* Multimodal, *S* Separable, *N* Non-Separable, *AE* Acceptable Error

在此篇文獻中，筆者解了很多的標竿問題，但最多只解到維度為30的標竿問題，在更高維度的時候未必可以得到如此優良的結果，不過在30維以下的時候就能夠有很好的結果。在參照其他學者專家的文章後可以發現，很多學者專家都是針對解的收斂速度來改善此演算法，我想原因就是因為SMO在進行找解的過程中，很容易就會要面臨拆組的動作，而這樣往往就會在找解的時候耗廢掉一些時間。

Requirements and Functionality Implementation of the System（系統運行需求與功能性實作）

1. 程式邏輯、結構（pseudo-code）

Algorithm 5 Spider Monkey Optimization (SMO) Algorithm:

1. Initialize Population, $LocalLeaderLimit$, $GlobalLeaderLimit$, pr.
2. Calculate fitness (i.e., the distance of individuals from food sources).
3. Select global leader and local leaders by applying greedy selection (see section 3.1.4, 3.1.5).

while (Termination criteria is not satisfied) **do**

(i) For finding the objective (Food Source), generate the new positions for all the group members by using self experience, local leader experience and group members experience.[Refer Algorithm 1].

(ii) Apply the greedy selection process between existing position and newly generated position, based on fitness and select the better one;

(iii) Calculate the probability $prob_i$ for all the group members using equation (4).

(iv) Produce new positions for the all the group members, selected by $prob_i$, by using self experience, global leader experience and group members experiences. [Refer Algorithm 2].

(v) Update the position of local and global leaders, by applying the greedy selection process on all the groups (see section 3.1.4, 3.1.5).

(vi) If any Local group leader is not updating her position after a specified number of times (*LocalLeaderLimit*) then re-direct all members of that particular group for foraging by algorithm (3)

(vii) If Global Leader is not updating her position for a specified number of times (*GlobalLeaderLimit*) then she divides the group into smaller groups by algorithm (4).

end while

Algorithm 3 Local Leader Decision Phase:

for $k = \{1...MG\}$ **do**
  if $LocalLimitCount_k > LocalLeaderLimit$ **then**
    $LocalLimitCount_k = 0$.
    $GS = k^{th}$ group size;
    for $i \in \{1...GS\}$ **do**
      for each $j \in \{1...D\}$ **do**
        if $U(0,1) \geq pr$ **then**
          $SMnew_{ij} = SM_{minj} + U(0,1) \times (SM_{maxj} - SM_{minj})$
        else
          $SMnew_{ij} = SM_{ij} + U(0,1) \times (GL_j - SM_{ij}) + U(0,1) \times (SM_{ij} - LL_{kj})$
        end if
      end for
    end for
  end if
end for

Algorithm 1 Position update process in Local Leader Phase:

for each $k \in \{1, ..., MG\}$ **do**
  for each member $SM_i \in k^{th}$ group **do**
    for each $j \in \{1, ..., D\}$ **do**
      if $U(0,1) \geq pr$ **then**
        $SMnew_{ij} = SM_{ij} + U(0,1) \times (LL_{kj} - SM_{ij}) + U(-1,1) \times (SM_{rj} - SM_{ij})$
      else
        $SMnew_{ij} = SM_{ij}$
      end if
    end for
  end for
end for

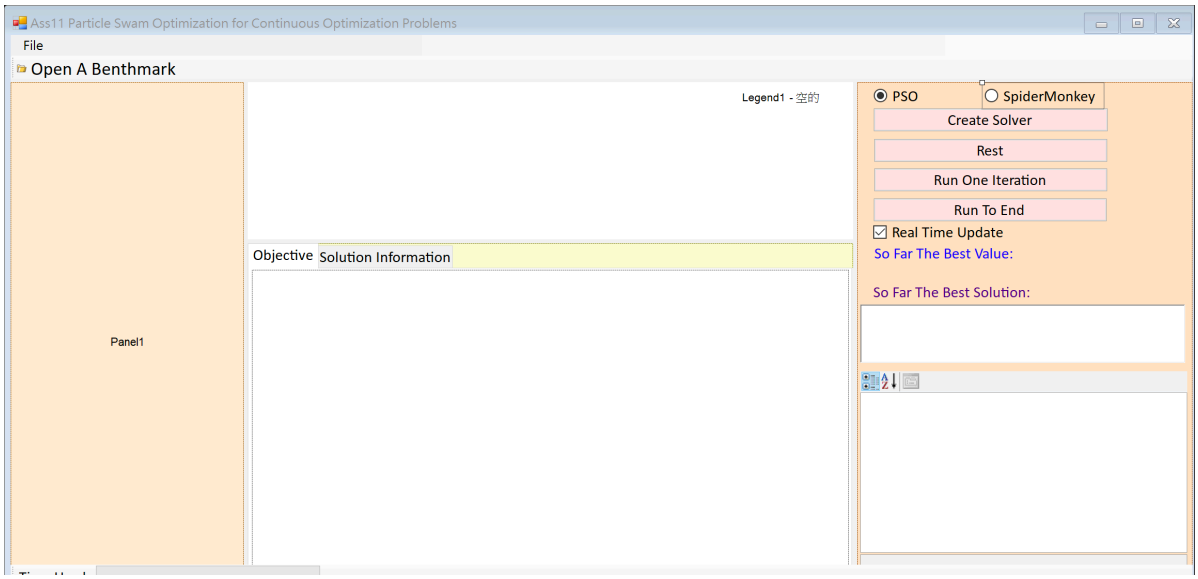
Algorithm 2 Position update process in Global Leader Phase (GLP) :

for $k = 1$ to MG **do**
  $count = 1$;
  $GS = k^{th}$ group size;
  while $count < GS$ **do**
    for $i = 1$ to GS **do**
      if $U(0,1) < prob_i$ **then**
        $count = count + 1$.
        Randomly select $j \in \{1...D\}$.
        Randomly select SM_r from k^{th} group s.t. $r \neq i$.
        $SMnew_{ij} = SM_{ij} + U(0,1) \times (GL_j - SM_{ij}) + U(-1,1) \times (SM_{rj} - SM_{ij})$.
      end if
    end for
    if i is equal to GS **then**
      $i = 1$;
    end if
  end while
end for

Algorithm 4 Global Leader Decision Phase:

if $GlobalLimitCount > GlobalLeaderLimit$ **then**
  $GlobalLimitCount = 0$
  if Number of groups $< MG$ **then**
    Divide the population into groups.
  else
    Combine all the groups to make a single group.
  end if
  Update Local Leaders position.
end if

2. 程式介面（UI）



3. 介面說明

- 左上角點選 Open A Benchmark(File)：匯入連續優化問題(檔名*.cop)
- 左側區：說明問題最佳解位置、objective value，以及問題說明
- 下方區：在 3D 圖中展現代理人的位置及移動方式
- 右側區：操作者控制演算法種類以及參數性質，並且顯現最佳解

4. 程式之執行步驟為

I. 讀檔

II. 產生Solver

III. Reset

IV. 執行演算

Numerical or Example Tests（運行測試結）

Table 2 Experimental results

Test Function	Algorithm	SD	ME	AFE	SR
f_1	DE	1.42E−04	8.68E−04	27378	100
	PSO	6.72E−05	9.34E−04	45914.5	100
	ABC	2.02E−04	7.57E−04	35901	100
	CMA-ES	2.90E−04	7.10E−04	21248	100
	SMO	8.38E−05	8.88E−04	15128.19	100
f_2	DE	3.54E−01	1.00E−01	25858	95
	PSO	2.36E−04	2.53E−04	38273.5	100
	ABC	5.92E−04	6.35E−04	20244	100
	CMA-ES	1.77E+00	1.44E+00	72184	36
	SMO	1.20E−05	2.34E−04	12018.41	100

$$AFE = \frac{\sum_{i=1}^{100} \#\text{ of function evaluations to meet the termination criteria for run } i}{100}$$

由於程式能力不足，所以未能將此演算法實現，因此參照了此文獻中的數據，可以發現到蜘蛛猴演算法在AFE的表現上皆優於其他的演算法，儘管在Mean Error的表現上未必能稱作最好，但是仍有其價值。

Conclusion and Discussion（結論與討論）

結論：

透過文獻我們可以知道蜘蛛猴演算法因為其原理間單、高效、及控制參數少的特性，在解決優化問題上有很大的優勢，而此篇作文研究蜘蛛猴演算法的始祖，其貢獻更是難以衡量，其師法FFSS這種特性的精神在我們日後進行meta-heuristic算法的時候，更應該好好的學習，以免落得為賦新詞搶說愁的問題，也因為這次的報告了解到自己還有很多的努力空間，更透過這次的報告了解到了mta-heuristic的浪漫。

Reference（参考）

Bansal J C, Sharma H, Jadon S S, et al. Spider monkey optimization algorithm for numerical optimization[J]. Memetic Computing, 2014, 6(1): 31-47.

DANG Tingting, LIN Dan. Spider monkey optimization algorithm with dynamic self-adaptive inertia weight. Computer Engineering and Applications

Swami V, Kumar S, Jain S. An Improved Spider Monkey Optimization Algorithm[C]//Pant M, Ray K, Sharma T, et al. Soft Computing: Theories and Applications. Singa- pore:Springer Press, 2018:73-81.