參、 蟻群優化演算法

蟻群優化演算法最早在 Dorigo(1992)的論文中被提出，其靈感源於螞蟻在尋找食物過程中發現路徑的行為。是一種求組合最優化問題的新式後設啟發式演算法；且該方法具有正面反饋、分散式計算和富於建設性的貪婪啟發式搜索的特點。是一種基於資訊量(費洛蒙)鋪設的人工智慧技術，可以透過它來尋求通過圖形最優路徑等極其複雜的問題找到解決辦法。

如上所述，透過螞蟻覓食時，若找到食物在搬運食物回程的途中會分泌一種特殊的費洛蒙(Pheromones)，以告訴其它的螞蟻可以循著這條路徑來搬運食物。雖然這些費洛蒙會隨時間的經過而蒸發，但若有其他螞蟻也循該路徑回來則會繼續分泌費洛蒙。因此，這些費洛蒙成為良好的指標，讓螞蟻得以同心協力搬回更多的食物。

螞蟻演算法基本上是利用群體智慧(Swarm Intelligence)演算的方法。所謂的群體智慧即是藉由觀察社會型昆蟲(Social Insects)的行為模式，從中發掘其特性並加以應用。此種策略的特別之處是群體中的每個獨立個體之行為皆會對環境的狀態造成影響；但另一方面，環境的狀態也會影響群體中每個獨立個體的行為。而在蒐集個體的行為進一步累積經驗之後，整個群體所表現出來的複雜行為即是透過個體的交互過程產生出的群體智慧，因此整個群體具有組織性，就算單一個體的任務失敗，群體也會繼續前進完成整體目標，並不會受到太大的影響。

蟻群優化演算法已應用於許多組合優化問題，包括蛋白質設計或路由車輛的二次分配問題。另外有許多派生的方法已經應用在實變數動力學問題、隨機問題與多目標並行的環境。同時，它也被用於解決 TSP 規劃問題的最適解。蟻群演算法可以連續運行並適應變化，為外，它也被用於解決網際網路跳點問題，當網路流量或基地站負載改變時，透過此規劃模式背後的改良式蟻群演算機制搜尋優化路由組合，可預見其能力將會優於目前 WiMax 所使用的傳統跳點規劃方法。

前面提到我們可以藉由制定一些規則來突顯或加強蟻群行為的特性，然後就可以利用這些特性進行編碼，一般來說，規則制定大約可分為兩大類。

1. 路徑的選擇規則(Pseudo Random Proportional Rule)

Dorigo(1997)在蟻群演算法尋路規劃文章中提及，利用路徑選擇規則選擇初始路徑以縮短計算時間，而且只在最好的解更新費洛蒙的濃度，經實驗證明，確實有助於螞蟻能以最快搜尋找到最適解。

2. 費洛蒙更新規則(Pheromones Update Rule)

蟻群演算法中的費洛蒙更新規則也分為兩大類，一類是費洛蒙及時(Online)更新，另一類則是費洛蒙離線(Offline)更新，前者用在每隻螞蟻走過該路徑後，後者是在該回合結束後，針對費洛蒙值做一些適度的調整。當然，這個機制不僅影響了之後螞蟻的路徑選擇決策，還會對最後解的優劣有著緊密的關係。

(1)費洛蒙即時更新規則：費洛蒙的濃度，與路徑上經過的螞蟻數量成正比與此路徑的距離長度成反比。這項規則所著重的是路徑的熱門程度，也就是費洛蒙濃度對螞蟻做出選擇時的影響，未對距離因素加以考量。

(2)費洛蒙離線更新規則：在該流程結束後，對於最佳路徑解來進行費洛蒙濃度的調整、變更，這就稱為費洛蒙離線更新。同時，對其餘非最佳路徑解做費洛蒙蒸發的動作，如此一來，在下次行程執行時，最佳路徑解被選擇的機率就會增加，得以縮短演算法的時間且增加其效能。

在完成規則的制定之後，蟻群間就沒有什麼直接的關聯性，牠們處在這個大環境下利用散發費洛蒙來進行溝通。如果有一隻螞蟻找到了食物，那牠就利用費洛蒙在走過的路徑上來告知其他的蟻群這裡有食物！在其他蟻群經過附近時，發現費洛蒙的存在自然就會聚集在一起。理所當然，這個時候費洛蒙的濃度也就會越來越高。

直至目前為止，我們的討論都是利用費洛蒙來判斷該路徑是否為最佳路徑解。但第一個螞蟻牠是如何找到食物的呢(初始路徑選擇)？實際上，一開始所有路徑上的費洛蒙濃度都是一樣，也就是說選擇任何一條路經的機率都是相同地。既然沒有外部因素的影響，那麼螞蟻就基於自身本能做移動，螞蟻會朝著一個方向前進，但並不一定保持直線前進，可能會走的歪歪區區且有一定的隨機性。因此，雖會儘量保持同一方向性，但是卻又有新的試探，尤其當碰到障礙物的時候它會立刻改變方向，這也可以看成是一種選擇的過程，也就是環境的障礙物讓螞蟻的某個方向正確。這就解釋了為什麼螞蟻在複雜像迷宮的現實世界中仍然能找到隱蔽的食物了。

透過以上的文字解說，相信讀者對於蟻群演算法的基本概念有一定程度上的理解。而蟻群優化演算法它最初的目地亦在解決 TSP 路徑規劃問題，希望找到最短路徑的路線。這個演算法相對簡單，它基於一組螞蟻，每隻完成一次城市間的遍歷，且每個階段，螞蟻會根據一些規則選擇從一個城市移動到另一個城市，而它必須訪問每個城市至少一次，當兩個城市間的距離越遠則它被選中的機會越小，當兩個城市邊際的一邊形成的費洛蒙越濃烈，則它被選擇的機率越大。

最後，完成旅程的螞蟻會在所有走過的路徑上沉積費洛蒙，但每次迭代後，費洛蒙將會被揮發。透過不斷進行這個過程，最後蟻群會選擇一條費洛蒙濃度最強的路徑行走，這也就是我們希望獲得的最適解。以下，我們將再進一步利用程序及相關的數學模式來加深讀者的印象，讀者亦可經由這些步驟轉換成為實際的代碼。如下

1、 每隻螞蟻隨機選擇一個城市出發，費洛蒙濃度 τ 一開始為零，透過初始化訓練可得到費洛蒙的初始濃度為 $\tau_0 = \frac{1}{NL^k}$ 。其中，N 表示城市的數目，L^k 為使用最佳鄰近法解得的總路徑長度。

2、 每隻螞蟻移動至下一個節點則會根據狀態轉換規則(State Transition Rule)，來決定下一步的目的地，此時費洛蒙濃度會對於路徑選擇造成影響。

$$p_{ij}^k(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^\alpha (\eta_{ij})^\beta}{\sum_{l \in J^k(i)} [\tau_{il}(t)]^\alpha (\eta_{il})^\beta}, & if\ j \in J^k(i) \\ 0, & otherwise \end{cases} \quad \textbf{(9-29)}$$

3、 更新路徑上的費洛蒙濃度，如下

$$\tau_{ij}(t+n)=\rho\tau_{ij}(t)+\Delta\tau_{ij}$$

(9-30)

其中，$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}$ 而表示如下

$$\Delta\tau_{ij}^{k}=\begin{cases}\sum_{k=1}^{m}\dfrac{Q}{L^{k}(t)}, & if\ kth\ ant\ uses\ edge\ ij\ in\ its\ tour\\ 0, & otherwise\end{cases}$$

(9-31)

4、 重複迭代後，距離越短的路徑上費洛蒙濃度上升；距離越遠的路徑上費洛蒙發散，最後該路徑遭到放棄。當產生的解不再變動或迭代到達我們所設定的次數時，結束蟻群優化演算法且獲得出最佳路徑解。

其中，$p_{ij}^{k}(t)$ 表示在 *t* 迭代時，第 *k* 隻螞蟻由城市 *i* 到 *j* 的概率，$\tau_{ij}(t)$ 表示在 *t* 迭代時，由城市 *i* 到 *j* 的費洛蒙濃度，η_{ij} 表示城市 *i* 到 *j* 距離的倒數，$J^{k}(i)$ 表示第 *k* 隻螞蟻在城市 *i* 時，尚未走過城市的集合。另外，α 、β 、ρ 及 Q 為控制參數。

範例 VBA_9_5_3 為了使程式碼如易閱讀，我們將程式分別使用三個[illegible]模組來運行它們不同的功能，其中包含Ants、Phes 及 SAC 等三個部份。其中，SAC 部份主要在生成模擬的城市距離數[illegible]顯示每隻螞蟻初始的路徑選擇及最後最適解的路徑選擇，其代碼如下：

```
Dim rand As New Random
Function makeGraphDistances(ByVal numberCities As Integer)
    Dim inner()
    ReDim inner(numberCities - 1)
    Dim i As Integer, j As Integer
    For i = 0 To numberCities - 1
        inner(i) = 0
    Next
    Dim dists()
    ReDim dists(numberCities - 1)
    For i = 0 To numberCities - 1
        dists(i) = inner
    Next
    For i = 0 To numberCities - 1
        For j = i + 1 To numberCities - 1
            Dim isValue As Double
            isValue = rand.next_2(1, 9)
```

```
16.                 dists(i)(j) = isValue
17.                 dists(j)(i) = isValue
18.             Next
19.         Next
20.         makeGraphDistances = dists
    End Function
    Sub displayView(trail, ByVal sheet As Worksheet, ByVal rowCount As Integer)
1.          Dim i As Integer, str As String
2.          For i = LBound(trail) To UBound(trail)
3.              str = str + CStr(trail(i)) + "       "
4.          Next
5.          sheet.Cells(sheet.UsedRange.Rows.Count + 1, 1).Value = str
6.          sheet.Cells(sheet.UsedRange.Rows.Count, 1).Font.ColorIndex = 7
    End Sub
    Sub showAntsView(ants, dists, ByVal sheet As Worksheet, ByVal rowCount As Integer, ByVal ph
As Phes)
1.          Dim i As Integer, j As Integer
2.          For i = LBound(ants) To UBound(ants)
3.              sheet.Cells(rowCount + i, 1).Value = CStr(i) + ":["
4.              Dim str As String
5.              str = ""
6.              For j = LBound(ants(i)) To UBound(ants(i))
7.                  str = str + CStr(ants(i)(j)) + CStr(" ")
8.              Next
9.              sheet.Cells(rowCount + i, 2).Value = str
10.             sheet.Cells(rowCount + i, 3).Value = "] Length=" + Format(CStr(ph.Length(ants(i),
    dists)), "#0.0")
11.         Next
    End Sub
```

注意，這裡我們再次引用 mscorlib.dll 組件，故讀者必須親自勾選這個引用，應用程式方能順利運行。另外，本列中我們所使用的陣列表示方法與之前常用的方法不同，這裡的用法可進一步表示成不規則矩陣，運用上將更為方便。

而 showAntsView 與 displayView 兩個程序，只是將運行後的陣列顯示於工作表中，並沒有什麼特別之處；讀者可以注意如何使用 LBound 與 UBound 來計算陣列的大小，以及使用 Format 表達浮點數或其它數值的方式。之後，我們說明另外兩個主要的類別模組，他們分別實際運行上面所述步驟中的第 2 步驟及第 3 步驟。

首先，在 Ants 類別中，看到全域變數α與β。所以很清楚地這裡主要在計算步驟 2 中的狀態轉換機率。另外，包含了每隻螞蟻如何初始化選擇路徑、追蹤路徑及選擇下

一個城市等代碼。這裡的 RandomTrail 函式我們實現了 Fisher-Yates 洗牌演算法，代碼如下

```
    Dim rand As New Random
    Const maxValue = 1.79769313486232E+308
    Const alpha = 3
    Const beta = 2
    Function initAnts(ByVal numberAnts As Integer, ByVal numberCities As Integer, ByVal ph As Phes)
1.      Dim inner()
2.      ReDim inner(numberCities - 1)
3.      Dim i As Integer
4.      For i = 0 To numberCities - 1
5.          inner(i) = 0
6.      Next
7.      Dim ants()
8.      ReDim ants(numberAnts - 1)
9.      For i = 0 To numberAnts - 1
10.         ants(i) = inner
11.     Next
12.     For i = 0 To numberAnts - 1
13.         Dim start As Integer
14.         start = rand.next_2(0, numberCities)
15.         ants(i) = RandomTrail(start, numberCities, ph)
16.     Next
17.     initAnts = ants
    End Function
    Function RandomTrail(ByVal start As Integer, ByVal numberCities As Integer, ByVal ph As Phes)
1.      Dim trail() As Integer
2.      ReDim trail(numberCities - 1)
3.      Dim i As Integer, idx As Integer, temp As Integer
4.      For i = 0 To numberCities - 1
5.          trail(i) = i
6.      Next
7.      For i = 0 To numberCities - 1
8.          Dim r As Integer, tmp As Integer
9.          r = rand.next_2(i, numberCities)
10.         tmp = trail(r)
11.         trail(r) = trail(i)
```

```
12.             trail(i) = tmp
13.         Next
14.         idx = ph.IndexOfTarget(trail, start)
15.         temp = trail(0)
16.         trail(0) = trail(idx)
17.         trail(idx) = temp
18.         RandomTrail = trail
    End Function
```

在 RandomTrail 代碼第 7 至 13 行中，我們由第 *i*(*i* 由 0 開始)個陣列元素開始，將其與其後陣列中隨意一個元素進行交換，來取得第 *i* 個陣列元素值，如此遞迴來取得每一個洗牌後的陣列值。

另外，在 nextCity 函式與 moveProbability 函式中，**我們**實現了輪盤選擇法來做為機率計算的基礎

```
    Function nextCity(ByVal isCount As Integer, ByVal cityX As Integer, visited, pheromones, dists,
ByVal ph As Phes)
1.          Dim probability() As Double, cumulate() As Double
2.          probability = moveProbability(isCount, cityX, visited, pheromones, dists, ph)
3.          ReDim cumulate(UBound(probability) + 1)
4.          Dim i As Integer
5.          For i = LBound(probability) To UBound(probability)
6.              cumulate(i + 1) = cumulate(i) + probability(i)
7.          Next
8.          Dim p As Double
9.          p = rand.nextdouble
10.         For i = LBound(cumulate) To UBound(cumulate) - 1
11.             If p >= cumulate(i) And p < cumulate(i + 1) Then
12.                 nextCity = i
13.             End If
14.         Next
15.         If nextCity < 0 Then
16.             Err.Raise vbObjectError + 513, "Ants.MyObject", "Failure to return valid city in
    nextCity."
17.         End If
    End Function
    Function moveProbability(ByVal isCount As Integer, ByVal cityX As Integer, visited,
pheromones, dists, ByVal ph As Phes)
1.          Dim adjacent() As Double
```

```
2.      ReDim adjacent(UBound(pheromones))
3.      Dim sum As Double, i As Integer
4.      sum = 0
5.      For i = LBound(adjacent) To UBound(adjacent)
6.          If i = cityX Then
7.              adjacent(i) = 0
8.          ElseIf visited(i) Then
9.              adjacent(i) = 0
10.         Else
11.             adjacent(i) = pheromones(cityX)(i) ^ alpha * (1 / ph.distance(cityX, i, dists)) ^
    beta
12.             If adjacent(i) < 0.0001 Then
13.                 adjacent(i) = 0.0001
14.             ElseIf adjacent(i) > (maxValue / (UBound(pheromones) * 100)) Then
15.                 adjacent(i) = maxValue / (UBound(pheromones) * 100)
16.             End If
17.         End If
18.         sum = sum + adjacent(i)
19.     Next
20.     Dim probability() As Double
21.     ReDim probability(UBound(pheromones))
22.     For i = LBound(probability) To UBound(probability)
23.         probability(i) = adjacent(i) / sum
24.     Next
25.     moveProbability = probability
End Function
```

在 moveProbability 代碼第 4 至 19 行計算訪問下一個城市的可能加總值，之後由代碼 22 至 24 行取得參訪各個城市的機率值。基本想法為 $p_i = \frac{F_i}{\sum_{i=1}^{n} F_i}$，其中 F_i 為根據向每個城市移動路徑中費洛蒙濃度計算而得，如代碼 11 至 16 行所示。這裡的 maxValue 為一個無限大的值。

而這裡的 updateAnts 函式、buildTrail 函式與 Phes 物件類別中的 updatePheromones 函式為更新蟻群、追蹤程序與費洛蒙濃度的主要程序。透過更新的過程，每隻螞蟻都可分配到一個新的、 隨機的起始城市而不保留舊的起始城市，而且重新更新每個城市間的費洛蒙濃度(信息量)，在新的迭代中計算一個新的移動路徑。

```
Function updateAnts(ants, pheromones, dists, ByVal ph As Phes)
```

```
1.      Dim numberCities As Integer, i As Integer
2.      numberCities = UBound(pheromones) + 1
3.      For i = LBound(ants) To UBound(ants)
4.          Dim start As Integer
5.          Dim newTrail() As Integer
6.          start = rand.next_2(0, numberCities)
7.          newTrail = buildTrail(i, start, pheromones, dists, ph)
8.          ants(i) = newTrail
9.      Next
10.     updateAnts = ants
    End Function
    Function buildTrail(ByVal isCount As Integer, ByVal start As Integer, pheromones, dists, ByVal
ph As Phes)
1.      Dim trail() As Integer
2.      ReDim trail(UBound(pheromones))
3.      Dim visited() As Boolean
4.      ReDim visited(UBound(pheromones))
5.      trail(0) = start
6.      visited(start) = True
7.      Dim i As Integer
8.      For i = LBound(pheromones) To UBound(pheromones) - 1
9.          Dim cityX As Integer, nexti As Integer
10.         cityX = trail(i)
11.         nexti = nextCity(isCount, cityX, visited, pheromones, dists, ph)
12.         trail(i + 1) = nexti
13.         visited(nexti) = True
14.     Next
15.     buildTrail = trail
    End Function
    Function updatePheromones(pheromones, ants, dists)
1.      Dim i As Integer, j As Integer, k As Integer
2.      For i = LBound(pheromones) To UBound(pheromones)
3.          For j = i + 1 To UBound(pheromones(i))
4.              For k = LBound(ants) To UBound(ants)
5.                  Dim islen As Double, dec As Double, inc As Double
6.                  islen = Length(ants(k), dists)
7.                  dec = (1# - decrease) * pheromones(i)(j)
8.                  inc = 0#
```

```
                    If edgeInTrail(i, j, ants(k)) Then
                        inc = increase / islen
                    End If
                    pheromones(i)(j) = dec + inc
                    If pheromones(i)(j) < 0.001 Then
                        pheromones(i)(j) = 0.001
                    ElseIf pheromones(i)(j) > 100000# Then
                        pheromones(i)(j) = 100000#
                    End If
                    pheromones(j)(i) = pheromones(i)(j)
                Next
            Next
        Next
        updatePheromones = pheromones
    End Function
    Function edgeInTrail(ByVal cityX As Integer, ByVal cityY As Integer, trail)
        Dim lastIndex As Integer, idx As Integer
        lastIndex = UBound(trail)
        idx = IndexOfTarget(trail, cityX)
        If idx = 0 And trail(1) = cityY Then
            edgeInTrail = True
        ElseIf idx = 0 And trail(lastIndex) = cityY Then
            edgeInTrail = True
        ElseIf idx = 0 Then
            edgeInTrail = False
        ElseIf idx = lastIndex And trail(lastIndex - 1) = cityY Then
            edgeInTrail = True
        ElseIf idx = lastIndex And trail(0) = cityY Then
            edgeInTrail = True
        ElseIf idx = lastIndex Then
            edgeInTrail = False
        ElseIf trail(idx - 1) = cityY Then
            edgeInTrail = True
        ElseIf trail(idx + 1) = cityY Then
            edgeInTrail = True
        Else
            edgeInTrail = False
        End If
```

```
End Function
```

BuildTrail 函式中，代碼 8 至 14 行我們維護訪問過的布林值陣列，以便創造新的追蹤，且並不包括重複的城市。而在 updatePheromones 函式中，透過代碼 6 至 12 行來更新費洛蒙濃度，這裡比處理蟻群追蹤程序要來的簡單許多。

當費洛蒙的值減少時，類比發散；增加時，則類比濃度增加。發散是將當前的費洛蒙濃度乘以係數小於 1.0 的值，取決於全域參數 decrease (ρ)。增加則更近蟻群選路之總路徑長度，所占比例由全域參數 increase (*Q*)的比例計算而得。最後，經由 edgeInTrail 與 IndexOfTarget 函式可以知道兩個城市間是否有關聯，也就是決定是否增加其費洛蒙濃度的主要依據。

另外，包含了初始化費洛蒙函式 initPheromones、計算兩城市之的距離函式 distance 與計算總路徑長度函式 Length 等，讀者可以參加實例中之代碼，這裡我們不再另作說明。讀者可透過本章眾多在解決 TSP 問題的演算法中，評估看看那一種是對您未來的工作或待解決的問題上可以提供較佳的適用性。

肆、 蜂群演算法