基於基因演算法之三維蜜蜂模擬

彭煜博　陳品中　柯冠宇　王派軒　陳忠義
國立中山大學
b0730400{02, 10, 11, 12, 29}@student.nsysu.edu.tw

摘要

基因演算法（genetic algorithm，GA）為一種演化式演算法（evolutionary algorithm），可用以求解複雜的最佳化問題。本文所使用的蜜蜂模擬程式，由先前期中報告所製作的程式修改而來，透過結合基因演算法，我們可以近似的模擬數個蜂群間的競爭關係。

1　前言

蜜蜂[1]是一種會飛行且具高度社會性的的昆蟲，基本上依角色不同可分為女王蜂、工蜂和雄蜂，前二者皆為雌蜂，但工蜂不具生育能力。

蜂巢的日常工作由工蜂負責，諸如防禦、築巢、育幼等等，相較於女王蜂和雄蜂的工作繁多。至於女王蜂與雄蜂的工作主要為生育，負責使蜂群的數量增加。

相較於期中報告，在期末的程式中，移除了作為掠食者的虎頭蜂，但還是能透過間接的參數設定生成行為類似掠食者的蜂群來達成模擬虎頭蜂。

在本次的期末報告中，我們將能透過基因演算法，來模擬多蜂群彼此間的互動及演化，達成期中報告時所不能達成的效果。

2　相關研究

基因演算法首先由 John Henry Holland 於 1975 年出版的 *Adaptation in Natural and Artificial Systems* 一書中提出，Holland 認為此演算法可以應用在各種領域中，如最佳化、經濟學、賽局理論等等領域。

一般而言，基因演算法可以分為數個部分，包括編碼問題參數、定義適應函數（fitness function）、選擇（select）、交配（recombine/crossover）和突變（mutate）等，針對不同議題，各個部分的實現會有所不同。

```
encode parameters
generate initial population
while termination condition is not satisfied do
    calculate fitness of each chromosome
    select the population
    recombine the population
    mutate the population
find the best chromosome from the population
decode the best chromosome
return decoded chromosome
```

圖一、基因演算法的虛擬碼

3　方法

BeeSimulation 為本次的程式的名稱，我們透過對期中的版本進行兩次的迭代，來達成所要實現的目標。

該程式的兩次版本迭代分別為重構蜜蜂等物件的互動模式和增加相應的管理類別（v1.1.0），以及使用侵入式的修改來增加對基因演算法的支持（v2.0.0）。

以下內容皆默認讀者已閱讀過期中報告，僅敘述新加入的名詞或概念。

3.1 重構（v1.1.0）

本版本為期中程式的改進版本，包含數個類別、函式和變數的修改，提供更加方便的方式來管理蜂群與花，增加不同的蜜蜂角色和移除捕食者（Predator）。

3.1.1 修改

在原先的程式中，部分類別、函式和變數存在表達不精確或其他的問題，因此在本版本中，我們做了適當的修改來改善這個問題，例如將類別 Food 重新命名為 Flower，使其名稱更加符合在模擬中所扮演的角色。

除此之外，我們刪減了一些不必要的成員函數，例如將類別 Bee 和 Flower（原類別 Food）中的 isDead 方法移除，因為它可以利用判斷生命是否為零來替代。

透過這些修改，我們可以使整體程式更加清晰，並讓後續的開發（v2.0.0）變得容易。

3.1.2 新的蜂群及花群管理 API

有鑑於原先的模擬環境中僅能支援一個蜂巢和一叢花朵，擴展性不佳，因此在本版本中特地為此進行優化。

我們使用類別 Colony 來管理類別 Bee 的實例，令使用者可以在模擬的長方形箱子內，任意擺放蜂巢位置，個別 Colony 所管理的蜂群都是獨立的，這使得長方形箱子內可以同時存在多個蜂群。

至於類別 Flower，我們採取類似的方法來管理，將 Flower 的實例使用類別 Flowers 進行控制，讓長方形箱子內存在不同花群成為可能。

因應以上的兩個新類別的加入，我們增加了類別 Environment 來對蜂群與蜂群和蜂群與花群之間的互動進行控制。

3.1.3 新的蜜蜂角色與移除捕食者

為了使蜜蜂的攻擊行為與採蜜行為分離（原本由 sense 統一表現），我們替類別 Bee 引入了兩種新的角色，分別為攻擊者和採集者，這是替日後加入基因演算法做準備，在設定好兩種新角色後，我們讓用於尋找蜜源的 sense 和用來回巢的 memorize（新增）僅適用於採集蜂，而用以攻擊敵人的 attack 僅適用於攻擊蜂。

基於以上的設定，我們可以讓一個蜂巢僅會生成攻擊蜂，如此一來便可以達成先前捕食者所扮演的角色。

3.2 基因演算法（v.2.0.0）

該版本為了加入基因演算法，需要對前一版本類別 Environment 和類別 Colony 與進行相當大程度的修改，令模擬程式與類別 GeneticAlgorithm 之間高度耦合，無法與前一版本之程式相容。

3.2.1 流程

所有有關類別 GeneticAlgorithm 的流程控制，都會由類別 Environment 管理。

首先，類別 GeneticAlgorithm 會根據控制參數進行初始化，初始化的內容包含指定數量的染色體（chromosome）等。

接下來在每一輪基因演算法的更新（evaluate 方法）中，會分別調用 select 方法、recombine 方法和 mutate 方法來達成染色體選擇、染色體交配和染色體突變。

3.2.2 染色體的選擇（select）

染色體的選擇透過 select 方法實現，該方法中使用簡單的循環方法進行選擇，即每一組染色體會與它的前一個和後一個相比，若一開始有 N 條染色體進行選擇，則選擇結束後，會得到 N 條染色體，並且會保證淘汰掉表現最差的那條（若第 i 個染色體的分數最低，則它在兩次的比較中都不可能勝利），不過如果 SELECT_RATE 之值被設定在(0,1]，則表現較差的染色體有可能會勝過較好的染色體，因而被選擇到。

3.2.3 染色體的交配（recombine/crossover）

染色體的交配透過 recombine 方法實現，該方法採取雙點法（two-point recombination），首先會先使用隨機方式選取要交換的起始點和終點，再將該兩點內所包含的所有數值進行交換（包含起始點和終點），交配的機率由 RECOMBINATION_RATE 控制，應設定在 [0,1]。

3.2.3 染色體的突變（mutate）

染色體的突變透過 mutate 方法實現，該方法的實現中，每一個點都有 MUTATION_RATE（應被設定在[0,1]）的機率被修改，每次的修改都會被設定在該點所允許的最大值與最小值之間。

3.2.4 適應函數（fitness function）

我們使用 *擊殺數* × *擊殺分數* + *採集數* × *採集分數* 作為適應函數，其中擊殺數和採集數會在每一次模擬中計算，而擊殺分數和採集分數為預先定義的數字。

4 實驗

在本次的實驗中，我們測試了三種不同的情境。未提及之參數皆按照程式本身的設定，以下情境中的蜂群分別放置在長方形箱子中的左下和右上方（以水平面來看）。

在三個情境中，每個情境我們皆會觀察第一個世代觀察到第一百二十個世代的演化情形，一個世代（generation）的評估時間皆為 1200 影格數（frame），使用影格數作為基本單位的原因是 p5.js 繪圖的單位為影格數，在擁有獨立顯示卡的情況下，以 WebGL 為後端時可以預見瀏覽器能跑滿每秒 60 影格數（60 fps），這也是使用 1200 影格數的原因，因為換算下來相當於 20 秒。

除了用 1200 影格數為單位外，每個蜂巢在每次模擬開始時，皆會生出 40 隻蜜蜂，這是因為我們將每個蜂巢中的染色體設為八條，如此一來每條染色體都能生出五隻蜜蜂，可以用較合理的蜜蜂基數來計算每條染色體的擊殺數和採集數。

在本次實驗中，基因演算法能夠訓練的參數共有九個，其詳細資料如下，前七個參數的單位可以視為各個力的乘數，第八個參數的單位為像素，是蜜蜂認定鄰居的基準之一，第九個參數的單位則是度數（degree），也是蜜蜂認定鄰居的基準之一，關於此二參數可見 Craig Reynolds 的 *Boids: Background and Update*[3]一文。

表一、基因演算法的可訓練參數

參數名稱	數值類型	最小值	最大值
sep	浮點數	0	1
ali	浮點數	0	1
coh	浮點數	0	1
sen	浮點數	0	1
mem	浮點數	0	1
wan	浮點數	0	1
atk	浮點數	0	1
dist	浮點數	50	300
angle	浮點數	10	180

另外，SELECT_RATE、RECOMBINATION_RATE 和 MUTATION_RATE 分別設為 0.1、0.5 和 0.5。

4.1 兩蜂群相互競爭，有三個花叢

在此情況下，我們將兩個花叢個別置於兩個蜂巢的前方，第三個花叢放置在兩蜂巢連線的中點。

在第一個世代時，雙方所產生的蜜蜂行為相當混亂，但在經過數十個世代後，雙方的染色體中會採蜜的染色體都會被挑選出來，當雙方都有一定的採蜜能力後，隨著世代的交替，可以觀察到既能採蜜又能攻擊敵方蜂群的染色體會逐漸勝出，此時具有絕對優勢的一方就會出現，可以觀察到優勢方造成既能把己方花叢完全採集完畢，

前往採集中央花叢，又能達到將敵方蜜蜂完全擊殺的局面，但在經過一定的世代之後，具絕對優勢的一方，可能因為過度發展而導致自身滅亡，讓另一方有崛起的機會，從此循環往復，形成此消彼長的輪迴。

雖然過度發展的情況不常發生，但在這些極少數的案例中，最常出現的情況是將 coh 增加到最大值，我們對這種情況的解釋是蜜蜂發現跟彼此間靠得越近，越容易採集到蜜，因為只要有一隻蜜蜂採到蜜，就能保證剩下的蜜蜂都採到蜜，進而壯大蜂群，但這會有個問題，當所有蜜蜂都跟彼此接近時，代表它們越容易形成一點，降低找到蜜的機率，也因此另一方的攻擊蜂能很容易地的將形成一顆圓球的採集蜂群一舉殲滅，造成原本優勢方滅亡的情況。

4.2 兩蜂群相互競爭，有一個花叢

在此情境中，唯一的花叢被放置在中央，我們觀察到蜂群的優化結果跟前面三個花叢的差異不大，唯一有變化的是兩個蜂群間不太會出現一方輾壓另一方的場面，並且進化所需的世代數明顯增加許多。

4.3 兩蜂群相互競爭，沒有花叢

在此情景下，由於採蜜蜂完全沒有用處，我們透過將 ATK_RATE 設定為 1，使得每個蜂巢生成的蜜蜂皆為攻擊蜂。

在前一兩個世代時，雙方的攻擊性都不高，但經過十幾個世代後，會開始出現一方將另一方完全擊殺的情況，再過幾輪後，情況會交替，開始新的循環，簡單來說，雙方的互動情況就不再有更多變化。

4.4 難以收斂的情境

在實驗的一開始，我們將基因演算法的前七個可訓練參數的範圍設定為$(-50,50]$間的浮點數，並猜想基因演算法最終能得到一個優良的解，但是不管在數種涵蓋上述三種情境的條件下，訓練近百次無法得到堪用的解，說的更精確點，跟初始隨機生成的染色體表現沒有顯著差別。

在兩蜂群相互競爭，有三個花叢的情況下，我們曾經對其進行超過一千五百個世代（約莫八至九個小時）的迭代，但結果還是如同上面提過的一樣，表現令人失望，但是不能確定是否經過更多次的迭代後，不能得到滿意的解。

在經過數次對數值範圍的調整之後，我們發現可變動範圍越小，其解空間越小，越容易在可以接受的時間內，得到不錯的表現，但是這樣設定之後，各個染色體之間的相異程度就會有所降低，這會是一個問題，不過我們認為這是時間與精度之間的取捨了。

5 結論

在本次報告中，我們對原本的蜜蜂模擬程式進行優化與改進，接著在此基礎上與基因演算法結合，進行三種不同狀況的實驗，發現基因演算法確實能優化蜂群的行為，此外還討論了基因演算法在當可調參數之數值範圍過大的情況下，會有難以收斂的情境。

由於程式在一開始設計時並未設想到後續會有如此多的改動，以至於各個類別在互動上發生了相當高程度的耦合，這表示如果未來還要新增新的功能，可能會是件難上加難的事，因此如有必要，在新增功能前，先評估現有程式之架構後重新設計並實現，會是更加合適的方法。

參考資料

[1] Wikipedia contributors. (2020, December 24). Honey bee. In Wikipedia, The Free Encyclopedia. Retrieved 14:49, December 30, 2020, from https://en.wikipedia.org/w/index.php?title=Honey_bee&oldid=996020210

[2] 黃衍明. (n.d.). 基因演算法之基本概念、方法與國內相關研究概況. Retrieved 19:32, December 22, 2020, from http://myweb.ncku.edu.tw/~ftlin/course/CAAD/CourseInformation/document/genetictaiwan.pdf

[3] Craig Reynolds. (2001, September 6). *Boids: Background and Update.* Retrieved 10:40, November 7, 2020, from https://www.red3d.com/cwr/boids/