

基因程式技術於螞蟻族群式搜尋之研究

A Study of Genetic Programming on Foraging in an Ant Colony

陳岳宏
Yueh-Hung Chen

台南師院資訊教育研究所
Institute of Information Education
National Tainan Teachers College
m87518@stu.ntntc.edu.tw

孫光天
Sun Koun-Tem

台南師院資訊教育研究所
Institute of Information Education
National Tainan Teachers College
ktsun@ipx.ntntc.edu.tw

摘要

基因程式設計(Genetic Programming)是由電腦模擬進化機制及過程，將程式自行演化以達成特定目的。在本研究中嘗試以基因程式設計來演化出一個程式，希望藉由此程式在一個模擬的環境中，產生 20 隻螞蟻的搜尋方式，使它們以合作方式，將蟻巢外的兩個食物堆中的食物小球搬運回蟻巢。本研究中捨棄了較複雜及功能較強大的函數與終端點運算，而將它們拆成更基本的組成單元，其結果顯示基因程式設計可成功的利用這些簡單的函數及終端點，在每個時間只有一動作的條件下，完成合作搬運食物小球的工作。

關鍵字：基因程式設計、螞蟻族群、整體行爲、終端點、演化

一、前言：

基因程式設計(Genetic Programming)[5, 6]是電腦科學中一個新興的領域。至今的一些研究成果，已經成功的利用基因程式設計演化出許多的程式，而所得到的程式中，有很多執行的結果幾乎等於或超越了專業研究人員所設計的程式[1]。

基因程式設計是以基因演算法(Genetic Algorithm)[4, 8]為雛型，將程式進行演化，基因演算法中的個體在基因程式設計中則轉變成程式單元可能構成的程式，而如何利用自然界中“物競天擇，適者生存”的法則，以交配(crossover)、複製(reproduction)及突變(mutation)的方式在廣大的程式個體集合中，發現適合的個體(程式)，正是基因程式設計所研究的範圍。基因程式設計的優點，在於我們僅需了解待解決問題的少量資訊，即可利用基因程式設計的機制來嘗試演化出解決問題的程式。當我們

面對一個沒有理想的演算法可以解決的問題時，此特性便顯得特別有用。例如產生一個集體行爲(emergent behavior)中每個個體的規則，或用以設計原本不易設計的非對稱通頻帶濾波器[7]。

其中整體行爲是指，在一個由許多簡單的個體所構成的群體集中，可使用一組適當而簡單的規則，應用在每個個體上，以使這個群體表現出相當複雜的整體行爲[1]。但設計一組適當的規則應用於個體上，使此群體表現出期望的整體行爲則是相當困難的工作，因為表現出整體行爲時群體中的個體是同時在運作，而且個體通常只能感應到四週的環境或其他個體的變化，以做為反應的依據，並不直接與其他個體通訊，因此在設計階段所設計的個體規則，在執行階段會有什麼樣的整體行爲出現，往往是難以預料的。而此現象，無論在自然界中，或是在資訊科學的領域裡，都有許多例子存在，例如螞蟻搜集食物的行爲以及細胞狀態機(cellular automata)的多數判斷問題(majority classification problem)[1]。

在本研究中我們將利用基因程式設計，以簡單的函數與終端點，演化出一個可以分別控制蟻群中每隻螞蟻的程式，使模擬的蟻群的行爲，表現出如同自然界中真實的社會行爲：搜尋、分工及留下費洛蒙(pheromone)訊息等，做為以基因程式設計演化整體行爲規則的嘗試。

二、基因程式設計

基因程式設計的構想是來自基因演算法(genetic algorithm)[4]。基因演算法的特性[3]在於：

1. 基因演算法中，參與運算的是問題解集合的編碼，而非問題的解。
2. 基因演算法同時考慮一群可能解，而非單一解。

3. 基因演算法僅運用可能解的結果資訊（以適合度函數(fitness function)評估），而不需要如導函數或其他輔助知識等額外資訊。
4. 基因演算法利用機率式的轉換規則，而非明確的決定規則，來引導整個搜尋方向。

由於以上特性使得基因演算法不同於以往的搜尋方式，既能在問題空間中找到接近整體的（非區域的）最佳解，在適合度函數上也不需要其他的輔助資訊。使得在許多的應用場合中基因演算法都有極佳的表現[3]。

在基因程式設計中，我們的目的是以進化的方式，產生一個特定用途的程式。首先、一個程式可以化為一個程式樹(program tree)來表示（如 Figure 1 所示），其中終端節點為一個程式中的變數或常數，內部節點為程式的運算元(operator)，會將其子樹傳回的值經此運算元運算後回傳給父節點，或經判斷後選擇其中的一個子樹執行。

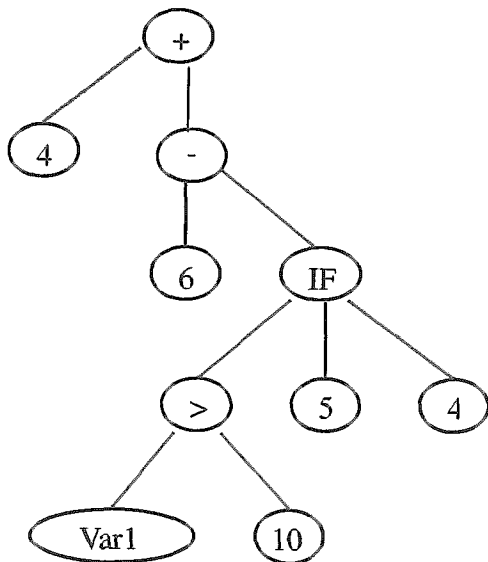


Figure 1、程式樹

利用基因程式設計來產生程式碼時，首先必須定義程式樹中所使用到的函數集合(function set)與終端點集合(terminal set)，其中函數集合中的函數(function)必須有適當的參數運算。在程式樹中，函數節點皆為程式樹的內部節點，而參與運算的參數則為其子樹所傳回的結果。一般來說，函數集合可以包含以下函數[5]：

1. 算術運算子 (+, -, *等)；
2. 數學函數 (sin, cos, exp, log 等)；
3. 布林運算子 (AND, OR, NOT 等)；
4. 條件判斷式 (If-Then-Else)；
5. 迴圈函數 (Do-Until)；
6. 遞迴函數；
7. 其他特殊領域所使用到的函數。

終端點(Terminal)通常為變數或常數，或者是不需參數參與運算的函數，在程式樹中，此種節點皆為程式樹的終端節點或樹葉。

另外，必須定義一個評估每個程式個體優劣的適合度函數，此適合度函數在產生不同用途的程式個體時，有不同的定義。且此適合度函數影響到整個程式族群演化的方向，因此良好的適合度函數定義可以加速理想的程式個體產生，使產生的程式個體體積更小，功能更完備。有了以上對程式組成單元及程式個體評估方式的定義後，尚需定義演化的終止條件、程式族群的個體數及程式個體參與交配(crossover)、複製(reproduction)與突變(mutation)的比例等。之後就可以利用演化的機制，在演化了指定的代數後或出現了理想的程式個體後終止演化過程，獲得所需的結果。

和基因演算法一樣，基因程式設計中有三個產生子代的重要運算：交配(crossover)、複製(reproduction)與突變(mutation)，以下是這三種運算的進行方式：

1. 交配：此運算會在兩個程式個體中，各以亂數選取一個分枝，之後將這兩個程式樹中，被選取的分枝交換，產生兩個新的子代程式個體（如 Figure 2 所示）。
2. 複製：此運算會將一個程式個體複製到下一個子代中。被複製的親代及子代程式個體將完全相同。
3. 突變：此運算會將一個程式個體中的某一個分支，以一個亂數產生的新分支取代（如 Figure 3 所示）。

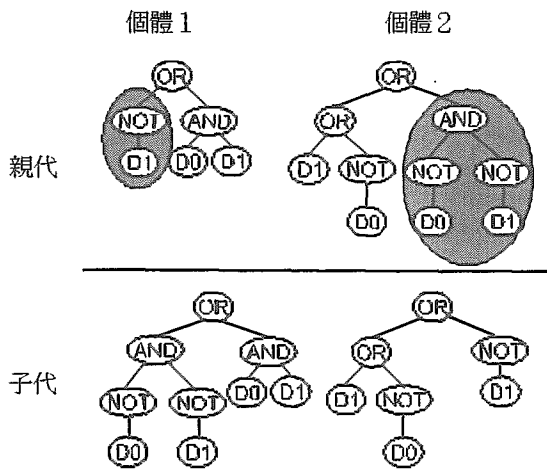


Figure 2、交配運算。(上半部為交配前，灰色部分為亂數選擇的子樹；下半部為交配後)

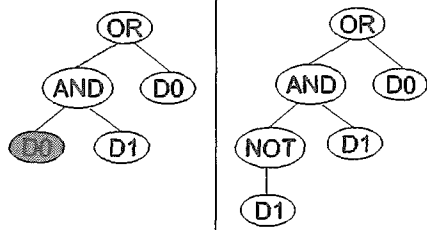


Figure 3、突變運算。左半部為突變前，其中灰色節點為亂數選擇之突變的分支；右半部為突變後結果。

在進行以上的運算時，是以機率式選取族群中的程式個體進行運算，一個程式個體的適合度越大時，被選取的機會也越大，在本研究中以輪盤式(roulette wheel selection)來進行親代程式個體的選取。輪盤式選擇的方法為：設有程式族群 $P = \{p_i \mid 1 \leq i \leq n\}$ ，其中 n 為程式個體數； $fitness()$ 為適合度函數，定義下列函數：

1. $sum = \sum_{i=1}^n fitness(p_i)$
2. $roulette_i = \sum_{k=1}^i fitness(p_k) / sum \quad i = 1 \text{ to } n$
3. 產生一個亂數 r ， $0 \leq r < 1$ ，則滿足 $roulette_i > r$ ， $1 \leq i \leq n$ 條件中最小之 i 即為所選取之程式個體。

整個基因程式設計的流程如下：

1. 產生一個初始的程式族群 (n 個程式)。其中每個程式個體都是亂數選取函數集中的函數或終端點集中的終端點組合而成。其進行的方式為：當產生一個子樹時，若亂數選取到的為函數集中的函數，則繼續為其產生子樹；若為終端點集中的終端點則停止。

2. 重複執行以下的步驟直到滿足終止條件：
 - (1). 執行程式族群中的每一個程式，利用評估函數來評估此程式對解決此問題的適合度。
 - (2). 利用輪盤式選取親代程式個體，以交配、複製與突變運算產生下一個子代。
3. 在所有的子代中，任一子代的程式個體，由適合度函數評估為最佳者即為基因程式設計之解

三、螞蟻族群式搜尋問題

為驗證我們所研究之基因程式設計效能，我們以螞蟻族群式搜尋問題為應用對象，因為在所有的社會性昆蟲中，螞蟻擁有最複雜的社會行為[2]，其中螞蟻搜尋食物也是採取分工合作的方式，發現食物的螞蟻在回巢的途中會沿途留下費洛蒙(pheromone)，其他螞蟻感覺到費洛蒙時會跟隨此費洛蒙而找到食物並加入搬運工作，此為一複雜的群體合作方式，但每一個個體工作又很單純，使螞蟻靠著這樣的機制，在每隻螞蟻都僅能感覺到四週環境的變化，而不直接互相交換訊息下，仍能合作完成目的。而每隻螞蟻在使用相當簡單的規則下，也使整個螞蟻群表現出相對複雜的集體行為(emergent behavior)。本研究即嘗試利用基因程式設計，演化出螞蟻搜尋食物的演算法，此演化而得的程式將應用在螞蟻中的每一隻螞蟻上。在模擬的環境中，每隻螞蟻都可以獨立完成搬運食物的工作，但螞蟻有合作的整體行為者適度將較高，也就是我們所希望的結果。

本研究中所模擬的世界為：在一個 32×32 方格的地區中，蟻巢設在(20, 11)為中心的 3×3 方格內，當螞蟻帶著食物走到蟻巢時便會自動放下食物；在(5, 17)及(15, 29)為中心的兩個 3×3 方格內共有 144 個食物小球，也就是每單位方格中有 8 個食物小球等待搬取。此螞蟻我們設有 20 隻螞蟻，開始時，20 隻螞蟻分散在以蟻巢(20, 11)為中心的 5×5 方格內，以亂數決定每隻螞蟻的位置及面對的方向。每隻螞蟻同時只能搬取一個食物小球。另外，模擬世界的邊界是相連的，因此當螞蟻走出模擬世界的邊界時它將出現在相對一邊的邊界上，螞蟻所遺留的費洛蒙可以維持一百個時間單位。整個模擬的世界將如 Figure 4 所示。

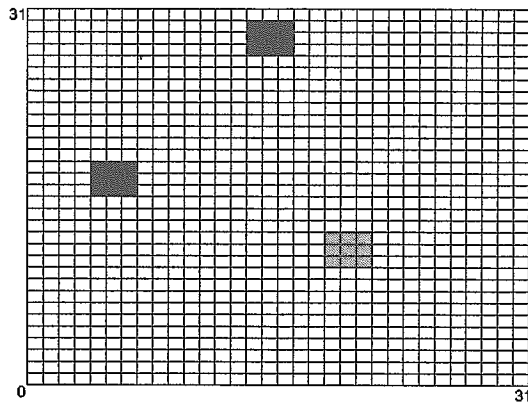


Figure 4、所模擬的螞蟻世界。其中淺灰色為蟻巢，深灰色部分為食物小球所在。

選取程式族群中的一個程式做為每隻螞蟻的控制程式，並開始模擬後，在每個時間單位中，每隻螞蟻執行此程式一次，在 4000 個時間單位後，或蟻群搬取完 144 個食物小球後，模擬即結束。

對此問題之前的研究中，有利用 PROGN 函數來做為函數集合中的元素之一[5]，此函數可以循序執行它的各子樹；另外也有研究者利用多代理架構 (multi-agent architecture)[2]，在每個代理所執行的終端點中，由優先權最大者先執行，而各終端點執行的優先順序則由研究者來排定，其程式樹如 Figure 5，當利用 Figure 5 的程式樹來控制螞蟻時，若此螞蟻所在方格中有食物，且螞蟻不是在搬著食物的狀態，則程式會在一個時間單位中依序執行搬取食物、向所面對的方向移動一格以及向東、南、西、北任意一個方向連續移動兩格等三個動作。這

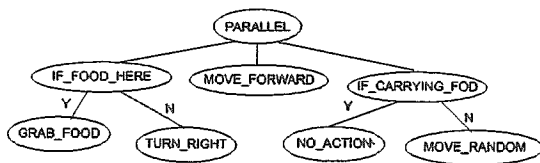


Figure 5、多代理架構的程式樹

些增加的函數可以產生更強有力的程式，使得螞蟻在同一個單位時間內，可執行多個動作。在本研究中我們捨棄了較強力的函數，而將它拆散成更簡單的幾個函數，來嘗試進化出解決此問題的程式。

四、函數集合、終端集合與評估函數

本研究中使用的函數都是做為判斷之用，並不會影響螞蟻位置或週圍環境，所有的函數都是雙參數的判斷式，當判斷條件成立時會執行第一個參數(左子樹)，不成立時執行第二個參數(右子樹)，這些函數有：

1. IF-FOOD-HERE：若螞蟻所在位置上有食物，則此判斷成立，否則不成立。
2. IF-FOOD-FORWARD：若與螞蟻所在的方格鄰接的方格中，螞蟻所面對的方向上的方格有食物則此判斷成立；否則不成立。
3. IF-CARRYING-FOOD：若螞蟻正搬運一個食物小球則成立，否則不成立。
4. IF-SMELL-FOOD：當螞蟻所在的方格其鄰接的四個方格(東、南、西、北)內有食物時，則成立，否則不成立。
5. IF-FACING-NEST：當螞蟻向前移動一步(一格)時並不會增加它與蟻巢的距離 d 時成立，否則不成立。此距離 d 定義為：

$$d = \text{abs}(x - x_{\text{nest}}) + \text{abs}(y - y_{\text{nest}})$$

其中 $\text{abs}()$ 為絕對值函數；

(x, y) 為螞蟻所在的座標；

$(x_{\text{nest}}, y_{\text{nest}})$ 為蟻巢的中心座標

6. IF-SMELL-PHEROMONE：當螞蟻所在的方格其鄰接的四個方格(東、南、西、北)中有費洛蒙時成立，否則不成立。
7. IF-PHEROMONE-FORWARD：若與螞蟻所在的方格鄰接的方格中，螞蟻所面對的方向上的方格有費洛蒙則成立，否則不成立。
8. IF-NEST-HERE：若螞蟻位於蟻巢的 3×3 方格中則成立，否則不成立。

所有的終端點，都是螞蟻實際的行動，將會影響螞蟻本身以及週圍的環境。我們定義有：

1. MOVE-FORWARD：此終端點會讓螞蟻向著面對的方向前進一格。
2. MOVE-RANDOM：此終端點會以亂數改變螞蟻的方向，成為東、南、西、北任一方向，再向前移動兩格。
3. TURN-LEFT：此終端點會讓螞蟻面對的方向逆時針轉 90 度。
4. TURN-RIGHT：此終端點會讓螞蟻面對的方向順

時針轉 90 度。

5. GRAB-FOOD：若此螞蟻並沒有搬著食物，則螞蟻會嘗試在所在方格內搬起食物小球，若所在方格內確實有食物小球，則食物小球數減 1 而螞蟻成為搬著食物的狀態。
6. DROP-PHEROMONE-AND-MOVE：此終端點會讓螞蟻在所在的方格內留下費洛蒙後，再向著面對的方向前進一格。

相似的研究中 [5] 有許多功能更強的函數，如 MOVE-TO-ADJACENT-FOOD-ELSE，包含了測試週圍方格內有無食物小球，以及移動到有食物小球的方格內兩部分功能；DROP-PHEROMONE 則會在螞蟻搬運食物的狀態下自動執行；MOVE-TO-NEST 則結合了判斷蟻巢方向與移動兩個功能，在本研究中都將它們分開，希望能藉由演化產生相似的結果。

在程式中採用較簡單的適合度評估方法，適合度評估函數如下：

$$f(x) = (4000 - time) / 10 + pellet \times 3 - size / 10$$

$$fitness(x) = \begin{cases} f(x) & \text{if } f(x) > 0 \\ 0 & \text{if } f(x) \leq 0 \end{cases}$$

其中 *time* 為經過的時間單位，而 *pellet* 則為成功搬回蟻巢的食物小球數，*size* 為程式的大小（=程式中所使用的函數總數加上終端點總數）。Table 1 為整個演化過程的參數設定。

Table 1、整個演化過程的參數設定

Function set	IF-FOOD-HERE, IF-FOOD-FORWARD, IF-SMELL-FOOD, IF-CARRYING-FOOD, IF-FACING-NEST, IF-PHEROMONE-FORWARD, IF-SMELL-PHEROMONE, IF-NEST-HERE
Terminal set	MOVE-FORWARD, MOVE-RANDOM, TURN-LEFT, TURN, RIGHT, GRAB-FOOD, DROP-PHEROMONE-AND-MOVE
Fitness case	單一測試樣本
Raw fitness	$f(x) = (4000 - time) / 10 + pellet \times 3 - size / 10$ $fitness(x) = \begin{cases} f(x) & \text{if } f(x) > 0 \\ 0 & \text{if } f(x) \leq 0 \end{cases}$ 其中 <i>time</i> 為所花費的時間單位，4000 為時間上限 <i>pellet</i> 為成功搬回的食物小球數 <i>size</i> 為整個程式的大小
Fitness	同 Raw fitness
Hits	成功搬回的食物小球數

Parameters	程式族群大小 M=100 (程式個體) 進化代數 G=100 產生子代時，各基因運算所占的百分比為：交配：80%；複製：17%；突變：3%
------------	---

五、系統模擬

本研究以表一的設定做九次演化，其中以第六次及第八次的結果最好，第六次的適合度及成功運回食物小球之曲線如 Figure 6 與 Figure 7 所示；第八次的適合度及成功運回食物小球之曲線如 Figure 8 與 Figure 9 所示。

其中第六次嘗試演化所得最佳的演算法如 Figure 10，其程式樹表示如 Figure 11 所示。由 Figure 10 及 Figure 11 可以發現，第六次嘗試演化所得的程式主要是以 MOVE-RANDOM 移動到有食物小球處，搬起食物後再以 MOVE-RANDOM 亂數移動回到蟻巢。因為此程式架構十分簡單，容易由演化而得，是一個區域極小值，因此在嘗試演化時經常可以發現此程式或類似的架構出現，而且因為其程式小，在適應度上佔相當優勢，很容易擴散而使整個程式族群的型式趨於一致，而落入區域極小值中。

IF-FOOD-HERE(IF-CARRYING-FOOD(MOVE-RANDOM, GRAB-FOOD), MOVE-RANDOM)

Figure 10、第六次演化所得最佳演算法

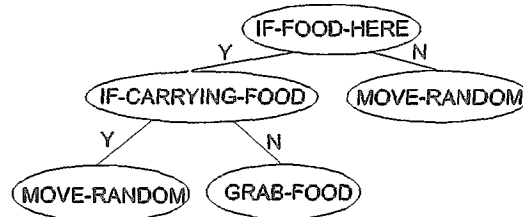


Figure 11、第六次演化所得最佳演算法的程式樹

第八次演化所得的最佳演算法如 Figure 12 所示，將不可能被執行到的分支去除後繪成程式樹如 Figure 13 所示。此次所演化出的程式碼除了具有類似 Figure 11 中程式的判斷外，此螞蟻族群利用費洛蒙做為引導其他螞蟻搜尋食物的工具，當有螞蟻發現四週有遺留的費洛蒙

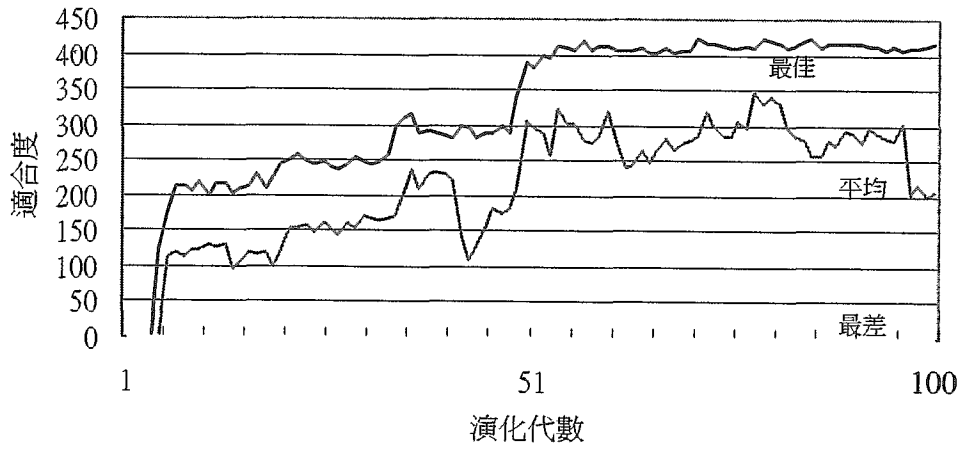


Figure 6、第六次演化中各子代之適合度曲線

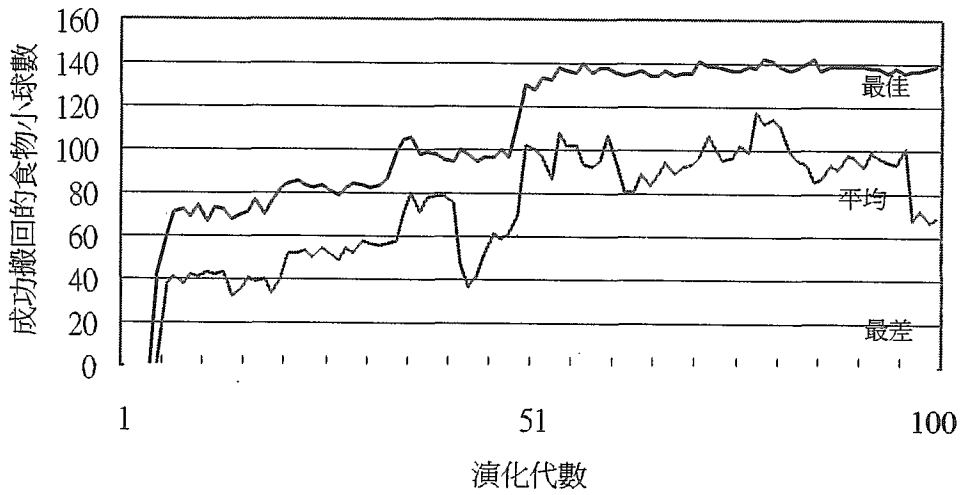


Figure 7、第六次演化中各子代成功搬回的食物小球數曲線

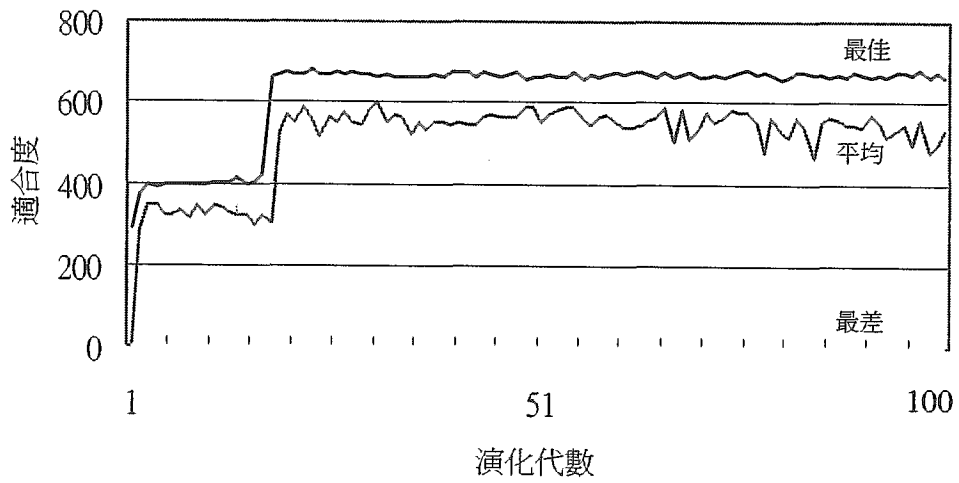


Figure 8、第八次演化中各子代之適合度曲線

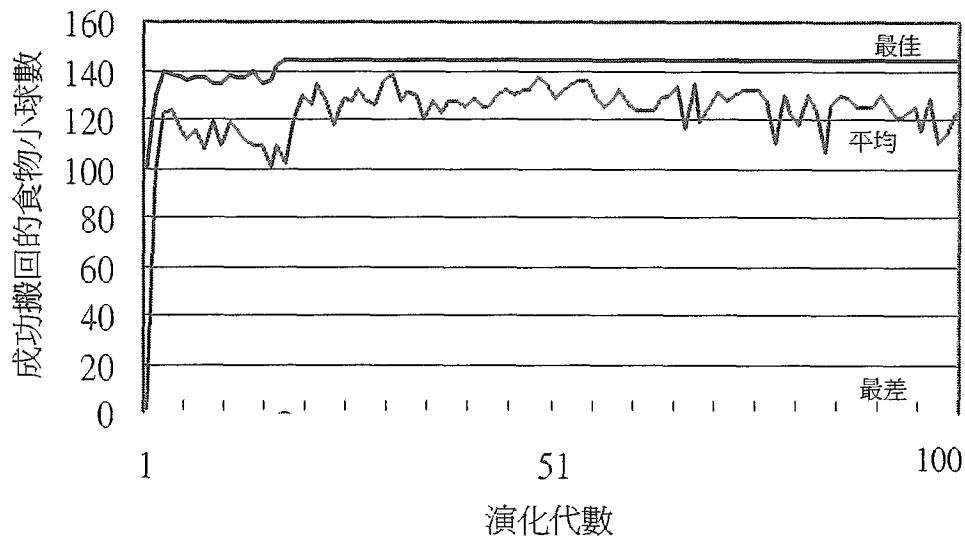


Figure 9、第八次演化中各子代成功搬回的食物小球數曲線

```

IF-CARRYING-FOOD(IF-CARRYING-FOOD(IF-FACING-NEST(IF-FOOD-FORWARD(IF-SMELL-PHEROMONE(IF-PHEROMONE-FORWARD(MOVE-RANDOM, GRAB-FOOD), IF -CARRYING-FOOD(TURN-RIGHT, IF-FACING-NEST(IF-PHEROMONE-FORWARD(MOVE-RANDOM, TURN-RIGHT), GRAB-FOOD))), MOVE-FORWARD), TURN-RIGHT), IF-PHEROMONE-FORWARD(IF-PHEROMONE-FORWARD(MOVE-FORWARD, MOVE-FORWARD), TURN-RIGHT)), IF -FOOD-HERE(IF-FACING-NEST(IF-PHEROMONE-FORWARD(MOVE-RANDOM, TURN-RIGHT), GRAB-FOOD), IF -SMELL-PHEROMONE(IF-CARRYING-FOOD(IF-FACING-NEST(MOVE-FORWARD, TURN-LEFT), MOVE-RANDOM), IF -CARRYING-FOOD(IF-FACING-NEST(IF-CARRYING-FOOD(TURN-LEFT, DROP-PHEROMONE-AND-MOVE), IF-PHEROMONE-FORWARD(IF-FOOD-FORWARD(GRAB-FOOD, IF -NEST-HERE(MOVE-RANDOM, MOVE-RANDOM)), GRAB-FOOD))), IF -CARRYING-FOOD(TURN-LEFT, DROP-PHEROMONE-AND-MOVE))))))

```

Figure 12、第八次演化所得最佳演算法

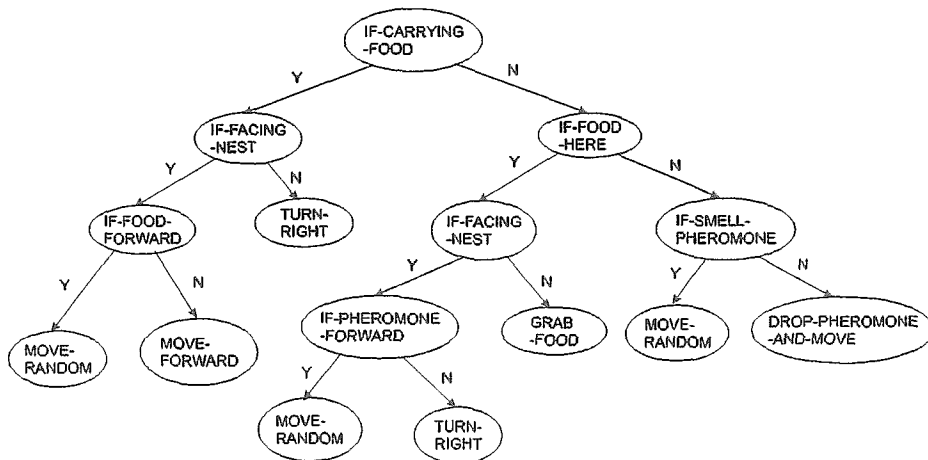


Figure 13、第八次演化所得最佳演算法化簡後之樹狀結構

時，即會以 MOVE-RANDOM 來搜尋鄰近的其他方格，因此當有一隻螞蟻成功的在食物小球附近留下費洛蒙時，附近的螞蟻便會開始聚集協助；但在搬取一個食物小球回巢後，又必須重新找尋食物堆的所在；另外，本次程式成功的演化出螞蟻在搬取食物小球後，即利用 IF-FACING-NEST 辨認蟻巢方向自動返回的功能，因此其效率遠較前幾次演化而得的程式更好。

六、結論與建議

在本研究中，我們嘗試以基因程式設計來演化出一個螞蟻群搜尋活動，在此搜尋過程中，將控制 20 隻螞蟻，使它們以合作方式，將蟻巢外的兩個食物堆中的食物小球搬運回蟻巢，本研究中捨棄了較複雜及功能較強大的函數與終端點，而將它們拆成更基本的動作單元，其結果顯示，基因程式設計可成功的利用這些簡單的函數及終端點，在每個單位時間只有一個動作的條件下，完成合作搬運食物小球的工作。

在研究中我們也發現，雖然基因程式設計與基因演算法一樣，較能避免落入區域極小值中，但一旦落入區域極小值而接近收斂時，突變的機制也很難讓整個搜尋跳出區域極小值，因為突變機制是以亂數決定程式個體突變的子樹，因此，突變出的程式子樹，在大部分的形情下並不能使整個程式個體的適合度比區域極小值小。

由許多例子及本研究中可以發現，基因程式設計可以在不需有相關背景知識下，演化出完成特殊目的的程式，而演化出的程式並不比專業人員所開發的程式遜色。此外，在以傳統程式設計來解決問題時，要設計一個群體中每個個體的規則，使整體行為表現出期望的效果並不容易，此點正可利用基因程式設計來加以實現。在本研究中，我們很成功的利用基因程式設計，將功能簡單的函數或終端點組合成功能更強大的程式，解決了螞蟻族群搜尋問題，使蟻群表現出合作搬運食物小球的整體行

為，肯定了此項技術之價值，進一步可應用到更複雜之族群社會模式。

參考文獻

- [1] David Andre & Forrest H Bennett III & John R. Koza, "Discovery by Genetic Programming of a Cellular Automata Rule that is Better than any Known Rule for the majority Classification Problem," *Genetic Programming: Proceedings of the First Annual Conference*, pp. 1-11, July 1996.
- [2] Forrest H Bennett III, "Automatic Creation of an Efficient Multi-Agent Architecture Using Genetic Programming with Architecture-Altering Operations," *Genetic Programming: Proceedings of the First Annual Conference*, pp. 30-38, July 1996.
- [3] Mitsuo Gen & Runwei Cheng, *Genetic Algorithms Engineering Design*, John Wiley & Sons. inc., 1997.
- [4] John H. Holland, *Adaptation in Natural and Artificial System*, second edition, MIT Press 1992.
- [5] John R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.
- [6] John R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, 1994.
- [7] John R. Koza & Forrest H Bennett III & David Andre & Martin A. Keane, "Automated WYWIWYG design of both the topology and component values of analog electrical circuits using genetic programming," *Genetic Programming: Proceedings of the First Annual Conference*, pp. 123-131, July 1996.
- [8] Melanie Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, 1998.