運用改良式基因演算求解無界限背包問題

陳榮靜
朝陽大學資訊管理研究所副教授
crching@cyut.edu.tw

簡程輝
朝陽大學資訊管理研究所研究生
s9454605@cyut.edu.tw

摘要

背包問題 (Knapsack Problem) 是一個 NP-Complete 問題 ， 而無界限背包問題 (Unbounded Knapsack Problem)更趨複雜，其困難度更甚一般的背包問題。在本文論文中，我們將運用基因演算求解無界限背包問題。我們將基因演算法做了一些改良 ，採用菁英策略（elitism strategy），克服傳統基因演算法中收斂速度過慢的缺點。在菁英策略中它保證了每個子世代至少會和原世代的表現一樣好。另一方面我們依問題空間複雜度 調整母體大小與執行次數，將每次的最佳值留置在精華群。最後在這精華群中取出佼佼者，以這種多重的選秀策略在廣大的搜索空間真正找到最佳者，實驗證明本方法能找出問題的最佳解。

關鍵詞：背包問題、基因演算法、演化式演算法、菁英策略。

Abstract

Knapsack Problem is one kind of NP-Complete problem. Unbounded Knapsack Problems are more complex and hard than general Knapsack problem. In this paper, we apply genetic algorithm to solve Unbounded Knapsack Problem,. The system we uses an elitism strategy to overcome the defect of the slowly convergence rate of general genetic algorithm. The elitism strategy keeps good chromosomes and not to be eliminated through the mechanism of crossover and mutation, and to make sure the features of offspring chromosomes are as good as their parents at least. On the other way, we adaptive the initial population of chromosomes of size and the executing times. The system will take the optimal value from the chromosomes of each finished round and put the value into an elitism group. Therefore, we can get the optimal values from the elitism group to find our real solution. The experiment results showed our method could find the real optimal solution of the problem.

Keywords : Knapsack problem 、 Genetic algorithm、Evolutionary algorithm、elitism strategy

1. 前言

背包問題 (Knapsack Problem)是作業研究領域中相當著名的探討主題[15];該問題敘述登山者的背包，以及多個可供選擇之項目，雖然每一個項目都可以為登山者帶來效用，但同時亦將佔用背包空間；因此，如何在背包容量限制下選擇某些項目，並為登山者帶來最大效用，即為背包問題所探討之主題。背包問題主要在討論如何從多件物品中，在符合限制條件前提下選出數件物品，其中每項物品只有 0 與 1 兩種選擇，使得得到最大總效益。而我們將背包問題以下列公式題說明之:

$$\text{Maximize} \quad \sum_{i=1}^{n} x_i p_i$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_i w_i \le C \qquad (1.1)$$

$$x_i \in \{0,1\},\ \text{i=1....n}$$

其中 p_i:項目 i 所帶來之利潤 (i=1,2…,n)

w_i :項目 i 的重量

C : 總重量限制

x_i : 項目 i 是否被選擇之決策變數，如果項目 i 被選入適合解 (feasible solution)內，則 x_i=1;

其他，則 x_i=0。

如式(1.1)所示之單一限制條件背包問題，其數學型式雖然相當簡單，但其複雜度為 2^n，當 n 較大時，若欲求其最佳解,則需耗費相當多的計算時間。

圖 1、無界限背包問題

無界限的背包問題 (Unbounded Knapsack Problem)其定義為[16]:在每一項目下都沒有界限，其唯一的適應條件是適應函數，而它具有以下三個特性

(1)它是個決策性問題

(2)它是個 0/1 問題

(3)對每一項目而言總重量必須小於等於C也就是適應函數的限制條件。

而無界限的背包問題如圖 1 所示，是在討論如何從多件物品中，選出數件物品，每項物品無限量供應，但必須符合限制條件即最大載重量，使得得到最大總效益 。如公式 (1.2)所示，雖與 (1.1)相同，但 x_i 的條件卻大有不同，x_i 是大於或等於零 的正整數 ，其複雜度為 $\prod_{i=1}^{n}(M_i)$，其中 M_i 是每一種選項可選擇的個數。*M_i 值大於等於 1*。

$$\text{Maximize} \quad \sum_{i=1}^{n} x_i p_i$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_i w_i \le C \qquad (1.2)$$

$$0 \le x_i, x_i \in \text{integer} \text{ , i=1.....n}$$

背包問題 應用 在企業管理方面亦 比比皆是，且其廣泛應用在如資本預算問題上、計畫排程上以及投資組合選擇上；而無界限背包問題也被應用於其它地方，譬如投資方案選擇方面，假設有數項投資方案可選擇、每項投資方案可得到的報酬不同，在有限的總投資金額情形下，從中選擇出數項投資方案以得到最大報酬，這類問題相當於求解背包問題。其它又如VLSI積體電路排列上 、平行排程上 、網路routing上、貨櫃裝載問題上、業務派遣問題上、財務管理...等問題，都能以背包問題表示。背包問題為一NP-hard 問題，當選項多時便無法在有限時間內求出最佳解，所以許多文獻提出各種不同的啟發式解法 ， 如 Genetic Algorithm[12][15] 、 模擬退火法 (Simulated Annealing)[14] 等 演化式演算法 (Evolutionary Computation) 求解方式，雖然演化式演算法比以往的啟發式解法花費較多的計算時間，但通常能得到較佳的解。自從演化式演算法被提出來，便被廣泛應用在各類不同的問題上。本篇論文 提出 如何針對無界限的背包問題的特性，以基因演算法設計產生並嘗試找到真正的最佳解。

基因演算法是模擬生物基因演化所建立的演算模式，根據達爾文進化論“適者生存，不適者淘汰”的道理，做為在廣大空間的一種搜尋機制，藉此來求解各種困難的組合以找到問題的最佳解。在1975年Holland與其學生提出自然界的演化是發生在生物染色體的基因中，它仿效生物界中的物競天擇的自然進化法則，由物種中選擇較好特性的母代，隨機相互交換位元資訊，以期產生更為優秀的子代，重複下去，產生適應性最強的最佳物種；其中每一種生物的特徵係來自於該物種上一代的基因排列，演化是指每一代基因所發生的變化情形，所謂適者生存是指這一代的基因排列優於上一代的基因排列，而產生比上一代更能適應環境生存的世代。因此基因演算法(GAs)是強調基因型的轉變 ，將問題的參數經過編碼成為基因格式，利用遺傳運算進行演化來找到問題的最佳解。這些遺傳運算演化過程 ，包括產生世代(population)、再生(reproduction)、交配(crossover)與突變(mutation) 等等。GAs不斷重覆循環的演化方式可視為是在龐大的搜尋空間中，選擇可行的區域做有系統的多點搜尋。這種平行多點找尋方式，與其他技巧所使用的搜尋方法有極大的不同，這也是 GAs 被視為是有效率搜尋方法的原因之一。應用 GAs 求解組合最佳化問題的基本精神是：將欲搜尋問題的參數編

碼成0與1的二進位碼型式，根據問題的條件或目標函數來設計適應函數。通常問題的目標函數可直接用來評估個體的優劣好壞，但是有些問題由於屬性的不同，它們的目標函數要經過適度的修正才能夠拿來當成適應函數，根據適應函數選出來的染色體將構成子代，而週而復始地產生新的population，直到有最合適的結果被產生出來。

在相關研究中，有關於在求解背包問題方面D. Pisinger(2005)[13]提出許多修正方法，他利用 profit 與 weight 關係做 weakly、strong、Inverse strong…等修正，使得解以線性方式呈現不致於分散於各處；C. Zhao(2005)[12]等人將 Stack filter 問題轉化為 0-1 的背包問題用 MMSE準則以基因演算法來求解。在基因演算法改良方面，蘇純繒及翁瑞聰(2002)[3]以啟發式方式加入洞悉法求解以避免落入不必要的解當中；顏榮政及鄭道明(2005)[6]自動的建構不同的同類型資源分配型態以達到自動最佳化求解目的；莊秉文及阮議聰(2001)[2] 運用決策樹理論並加入了學習機制，以基因演算法來提出一種方式演化五子棋的策略；C.Zhao(2005.5)說明基因演算法要調整突變機率及交配機率以期找到較適的參數以求最佳解；吳泰熙等(2005) [7] 以基因演算法求解單原片方形物件排列問題提出菁英策略。

論文其他章節包括第 2節基因演算法及問題描述，第3節是我們提出的菁英式演算法，第4節是實驗結果與數據，第5節是結論及未來建議方向。

2. 基因演算法及問題描述

本文利用案例探討的方式將基因演算做一實際的應用，再加入事先選擇項的預估複雜度，以衡量所需母體的個數，及所需執行的次數，以求得真正的最佳解；基本上的GAs還是運用 Holland 所提出的步驟。依據 M. Negnevitsky(2004)[10]所提步驟，設計 GA 求解問題可分成下列十個步驟進行之。

(1) 以固定長度 設計 問題 domain 的編碼方式(每個個體表示法)，選取染色體的母體N，交配率 P_c 及突變率 P_m。
(2) 定義適應函數以量測執行後的狀態。
(3) 隨機產生初始的染色體母體個數N個。
(4) 計算出每個獨立染色體的適應函數值 $f(x_1)$、$f(x_2)$、$f(x_3)$‧‧$f(x_N)$。
(5) 在目前的母體中選擇一對染色體作配對，在母代染色體中以機率方式隨機選出交配
(6) 藉由基因交配及突變運作產生子代染色體。
(7) 將產生的染色體放於新的母體中。
(8) 重覆步驟(5)直到新的染色體數等於初染色體數。
(9) 新一代的染色體取代原染色體。
(10) 重回步驟(4) 直到整個過程滿足終止條件。

本論文首先將無界限背包問題 轉成基因染色體表示，接著產生隨問題 Domain 的大小調整初始染色體的個數及執行的代數，接著執行菁英式基因演算法，求取最佳解，我們重複以此改良式基因演算法多次，並從中取得最佳結果。由於本研究的主要目的，乃在於探討無界限的背包問題，找出其組合在重量限制下最大價值的選擇模式，為了選取最大價值下的組合，本研究將運用改良式基因法尋求最佳化的選擇。

假設問題描述如下: 背包的載重量選定為 C 公斤後，共有 n 種物品，每一項的重量分別為 W_1、W_2、‧‧‧Wn，在此假設每項價值也不同，分別為 P_1、P_2、‧‧‧Pn，假設 n=10 時如表 1，我們舉例假設背包總載重量容量有 C 公斤，每件重量與價值設定下，在最大重量限制之下，我們運用基因演算法求出最佳解。

表 1: 假設 n=10 其項次、每件重量與價格

項次編號	1	2	3	4	5	6	7	8	9	10
每件公斤	22	50	20	10	15	16	17	21	8	25
每件單價	1078	2350	920	440	645	656	816	903	400	1125

3. 菁英式基因演算法

上述的基因演算法 步驟我們將之歸納成四部份。這四部分包括基因編碼方式 (gene coding)、交配與突變機制 (crossover and mutation mechanism)、適應函數 (fitness function)、挑選機制(selection mechanism)，另外再加上一個過程，即執行次數(execute times)而整個基因的演算流程如圖 2所示，首先根據選擇項的大小選擇母體數N值，接著依照基因演算法的程序，先編碼後產生母體初值，然後在整個循環中交配、突變、檢驗、評估、挑選

中運作，直到第一個代數門檻後，先取出最佳值置於菁英群中，最後到第二個次數門檻後，從菁英群中取出最佳值，茲分述如下：

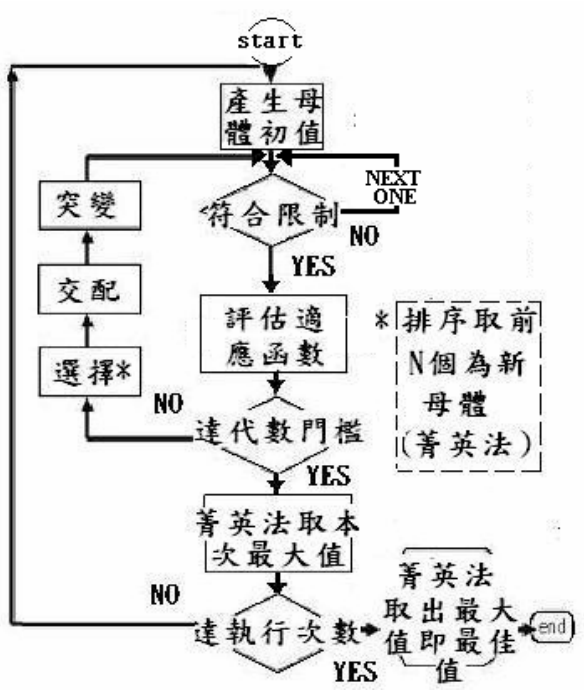

圖 2 基因演算流程

(1)具縮減位元數的基因編碼方式(gene coding)

如何把問題轉成染色體方式 ？我們由個別的項次重量特性來著手，也因為在重量特性下來處理，也相對的考量其限制，在最大重量小於等於C公斤的基本限制下，每種項目個別的件數就被限定了。

①個別定義：如第 1 項其重量為W_1kg 下的選擇組合最多可為l_1即 C/W_1 件，其選擇件數則為由 0 至l_1件亦即選擇項有l_1+1；所以其它各項的選擇個數依序為 (l_2+1)、(l_3+1)、・・・(l_k+1)，數學模式描述如(2.1):

$$S_i = \{x | x \in Integer; 0 \le x \le N_i + 1\}$$

$$\prod_{i=1}^{k} (l_i + 1) = \prod_{i=1}^{k} (C / W_i + 1) \qquad (2.1)$$

其中 S_i 是物品 i 可選擇的個數集合；x 是選擇個數。

例如我們 100kg 限制為例的選擇如表 1 其可能的件數如表 2 所示，其總選項共有 94,594,500 個選項。

②定義轉為 2 進位：依據表 2 我們將每件可能的組合情況轉為 2 進位，表示如表 3 所示。

③縮減位元數(bits)：由表3觀察位元變化，我們發現，除了第4與9項需用4 bits表示外，大部分的項目表示法並不需要用 4 bits，第2項因為只有0、1、2三個可能，所以只要2個bits即可，而第1、3、5、6、7、8、10等項只需3 bit；每種項目可能的件數我們以常用的2進位表示法來代表，因為數量可能的不同我們將其 2進位元表示方式依據實際可能刪減為如表 4，如此表示所佔的位元數比較少可以提升效能。

表2、 每種項目可能的件數

項 目	公斤/件	每種項目可能的件數											
1	22	0	1	2	3	4							
2	50	0	1	2									
3	20	0	1	2	3	4	5						
4	10	0	1	2	3	4	5	6	7	8	9	10	
5	15	0	1	2	3	4	5	6					
6	16	0	1	2	3	4	5	6					
7	17	0	1	2	3	4	5						
8	21	0	1	2	3	4							
9	8	0	1	2	3	4	5	6	7	8	9	10	11	12
10	25	0	1	2	3	4							

表 3 、每種可能的件數 2 進位表示法

項目	每種項目可能的件數用 2 進位表示法											
1	0000	0001	0010	0011	0100							
2	0000	0001	0010									
3	0000	0001	0010	0011	0100							
4	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	
5	0000	0001	0010	0011	0100	0101	0110					
6	0000	0001	0010	0011	0100	0101	0110					
7	0000	0001	0010	0011	0100	0101						
8	0000	0001	0010	0011	0100							
9	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
10	0000	0001	0010	0011	0100							

表 4、 每種項目可能的件數 2 進位較適表示法

項目	每種項目可能的件數用 2 進位 較適表示法											
1	000	001	010	011	100							
2	00	01	10									
3	000	001	010	011	100							
4	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	
5	000	001	010	011	100	101	110					
6	000	001	010	011	100	101	110					
7	000	001	010	011	100	101						
8	000	001	010	011	100							
9	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
10	000	001	010	011	100							

④染色體形成：將每一項次數量個別表示組合起來，形成基因表示形式，來代表選擇項目可能的組合，轉換成染色體如圖 3，依順序排列如圖 3(a)表示者為 0010、0000、0000、0000、0000、0000、0010、0000、0011、0000 代表第 1 項選 2 件、第 7 項選 2 件、第 9 項選 3 件，而圖 3(b)表示者為較適表示法。

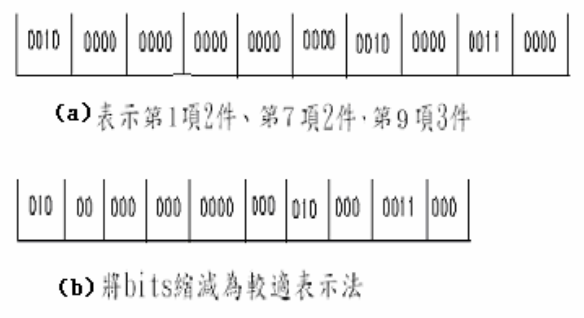

圖 3、項目選擇之染色體表示法

(2)交配與突變機制 (crossover and mutation mechanism)：

交配：每種的交配形式，其主要觀念都是相同的。即是以隨機的方式產生亂數，再與事先預設的交配機率進行交配。一般交配的方式有單點交配(One-Point Crossover)、兩點式交配(Two-Point Crossover)與字罩式交配 (Uniform Crossover)。

①單點交配(One-Point Crossover)：在所選出的兩字串中，選取一交配點，並交換兩字串中此交配點後的所有位元。

②兩點式交配(Two-Point Crossover)：在所選出的兩字串中，選取兩個交配點，並交換兩字串中兩個交配點間的所有位元。

③字罩式交配(Uniform Crossover)：產生與物種字串長度相同的字罩當作交配時的位元指標器，其中字罩是隨機地由 0 與 1 所組成，字罩中為 1 的位元即是兩物種字串彼此交換位元資訊的位置。

本論文選擇交配的機制為單點交配，將染色體由母體中選出，做單點交配以產生出子代。但為了交配均勻，不致產生固定點，我們採用以亂數的方式產生 0-9 數字，由最右邊到亂數值加 1 之項互相置換，如圖 4 所示以後 3 項為交配點，交配機率:設定為一般狀況 70%為主。

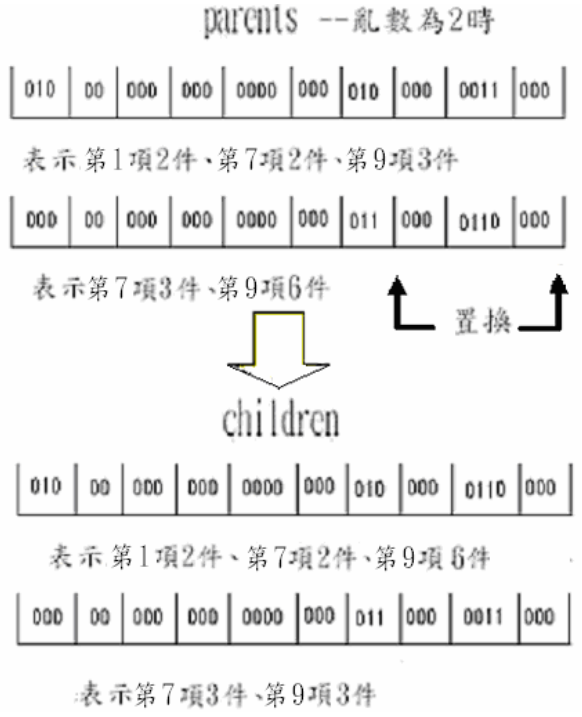

圖 4、染色體之單點交配表示法

突變(mutation)的方式，依每種可能的單項項目的選項做突變，換言之第 6 項的組合為 0、1、2、3、4、5、6，雖然它擁有 3 bits，依 2 進位元的方式它本應有 111 的存在，但在本案例僅有七種選項，突變的範疇不外乎這 0 到 6 其中之一選擇，111 不屬於範疇內故不在突變的選取項中，突變的方式如圖 5 所示。

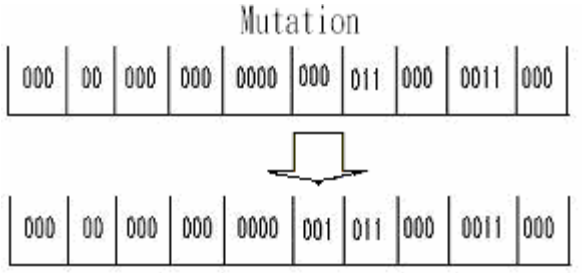

圖 5 染色體突變的方式

(3)定義適應函數值(Fitness function):

對每一個獨立的染色體我們的適應函數如 (3.1)式定義適應函數,(3.2)式是定義限制範圍，而 x 的項次在本例中依項目的不同有 0,1,…12 等選項，而 i 的項目有 n 種，每一項 i 具有 p_i 的價值，含有 w_i 的重量。

$$(KP)\max imize\sum_{i=1}^{n} p_i x_i \qquad (3.1)$$

$$f(s)=\sum_{i=1}^{n} w_i x_i \le c=100 \qquad (3.2)$$

$$0 \le x_i, x_i \in \text{integer}，\text{i=1…n}$$

(4)挑選機制 (selection mechanism):

我們選用的法則為精英法 則 (elite method) 或稱之競爭選擇法 (tournament selection method)。在這個條件下交配後或者突變後產生的子代 ，有些就不符合重量限制，留下符合者當子代，不符者則淘汰，而目標為留下來的子代中總價較高者。依問題的複雜度選取不同個數 染色體當 初始個數群體規模 N，在選項數目較低時 N 值較小，反言之在選項數目較高時 N 值則較大，我們分別以不同的 N 值來觀察，以適當的交配率 P_c 與突變機率 P_m 作觀察，先由目標及適應函數做評估，接著執行選擇、交配、突變藉以產生新的母體，再循環藉由限制重量及適應函數做評估，判斷是否達到代數門檻值，若未達到則又繼續執行選擇、交配、突變...，若已達交配代數門檻值，便找到最佳之染色體。

4.實驗結果與數據

4.1 實驗安排

經過這樣的基因演算法，我們以 VB 程式在 Intel Pentium(R) M processor1.6GHZ 電腦執行，系統畫面如圖 6 所示；項目的每件價值，單位重量，及限制重量可依問題需要更動，基因演算的條件方面，母體數、世代數、交配機率與突變機率，也可依需要更動，直接在畫面下輸入相當的數據。如本文前所提的十種項目以 100 公斤下，同樣的價格、每件重量、總重限制、交配機率假設 70%、突變機率 50%、母體數定為 200、代數設為 100 代，以菁英選擇法，在這程式下某次執行以基因演算法第 19 代就已經找到最佳解，如圖 6，這項目配置總價最高的最佳解為：第 1 項 2 件，第 9 項 7 件，其重量分別為 22KG、8KG 合計為 22KG X 2 + 8KG X 7= 100KG，組合方式為 20000000070，轉 換成 二進制 基因 的模式組合 為 010.00.000.000.000.000.000.000. 0111.000，而其價格為 1078 x 2 元＋400 x 7 元＝4956 元，是最佳解。

圖 6 程式參數說明

4.2 實驗結果與討論

一般而言 基因演算法 都是調整突變機率及交配機率 以期找到較適的參數以求最佳解；普遍都是調整這兩個因子，而在本文中乃更動母體數及執行次數為主，本實驗中依本文所闡述，以不同的選項來更動母體數與執行次數，在其它參數均相同的情況下，選項越多母體數當增多，執行數亦當隨之增多，我們把每一次執行隨機選的母體數當成廣大搜尋的網子布撒在搜索空間中，再利用基因演算法的局部收斂情形，透過菁英法由每代新產生的基因裡都選最佳者 ，在所有的搜尋範圍內取最大值，以這樣的方式去真正找到最佳值之機率就變得非常高；我們以載重量 55 公斤、82 公斤

及 100 公斤的數據，其選項分別為 435456、11404800、94594500 以得到以下圖 7 之數據，我們以選項預先算出選項的多寡，以 3%、90%且母體數分別為 20、50、100 執行 30 次得到結果如圖 7 所示，當載重量越大時選項越多，要得到最佳值的機率越少，故當選項越多時，欲得到一定的最佳值數量，相對的母體數就得增多，或者執行的次數增多調整這兩個指標使得達到一定的最佳值數量，我們以前面的方式項目載重量不同得出選項與每項價格及重量不變的情況，以 70%交配機率與適合的突變率及均勻分佈的交配點以菁英法選秀後，再將之應用於變化不同的重量與單價及 整體的限制重量之中。

在本實驗下 當然列舉法每次一定可以找到最佳解，但以基因演算法一般卻有一個現象即收斂到區域的最適解，這樣的情況在項次較少下，列舉法或許優於基因演算法，在項次多搜尋維度較大的時候 ，基因演算法優於列舉法，我們為了讓最佳解之搜尋能等同於列舉法之必定找到最佳值，所以加上了母體數與執行次數的參數調整，當選擇數越多時我們將母體數與執行次數加大，最佳值的出現也成正比的關係成長，在圖 7 的各項圖表裏面我們可得到這樣的結果。

在圖 7(a)為在載重量 55kg 突變率 3% 執行 30 次下最佳值出現的個數，當然母體數越大出現最佳值的次數越多；在圖 7(b)為同在載重量 55kg 執行 30 次下但突變率改為 90%最佳值出現的個數，在這兩個圖比較下為求最佳值的出現，以菁英法突變率越高越佳，且母體數設定越多最佳值出現個數越多。相同地在圖 7(c)、(d)有同樣的狀態，出乎預期的在載重量 100kg 下在突變率 3%與 90%的變化卻不大，為了探究這樣的情形，將執行次數增加為 80 次來觀察突變率與載重量變大(選擇項加多)與突變率之間的關係，卻發現在載重量變大(選擇項加多)突變率 90%情況下最佳值反而出現較少，在 3%與 90%之間差入了突變率 30%的觀察點卻發現在突變率 30%下最佳值出現次數最多。

一般而言 突變是為了使原先基因 組合群中的基因可以具有較多的變化性，以擴大搜尋空間的範圍，而搜尋的範圍與突變率成正比，當突變率過高時 ，將會遺失掉原先 較佳的基因，但用菁英法我們將這樣的缺點修正了保留住較有用的染色體，所以我們在圖 7(a)、(b)、(c)、(d)因選擇項在一定程度下對於這樣的最佳化問題，高突變率方式確實可以比較低的突變率所求得之解來的有效 且最佳解出現個數來得多；但在較高的選擇項下，高突變率使基因演算法流於隨機搜尋而較難收斂，也因此使求解的過程中，族群適應函數值產生振盪情形，雖然用菁英策略使其調整，但其最佳值出現頻率不及較低的突變率如圖 7(e)、(f)。

當然在傳統的基因演算法中，因為前一代的基因群被後一代所取代，如果增高其突變率的話，會造成擁有較 佳值的染色體的數目減少，但我們用菁英法克服了且反而增加了搜尋範疇；當減低突變率太多的話，卻縮小了搜尋的空間，所以必須在找尋最佳值的收斂過程中和擴展搜尋的空間之間做一個平衡調適。由圖 7(g)我們觀察到這樣的結果。

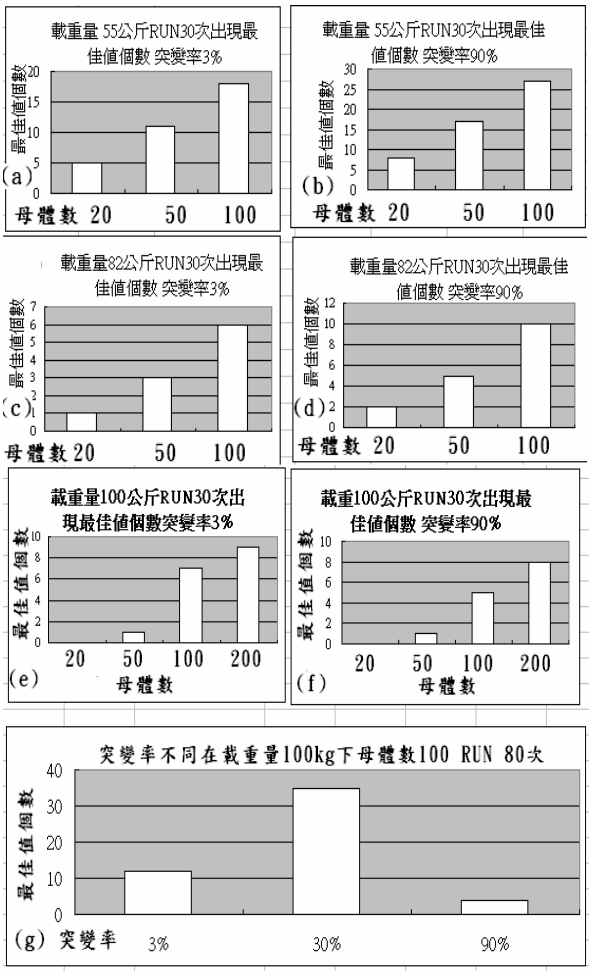

圖 7 實驗結果與數據

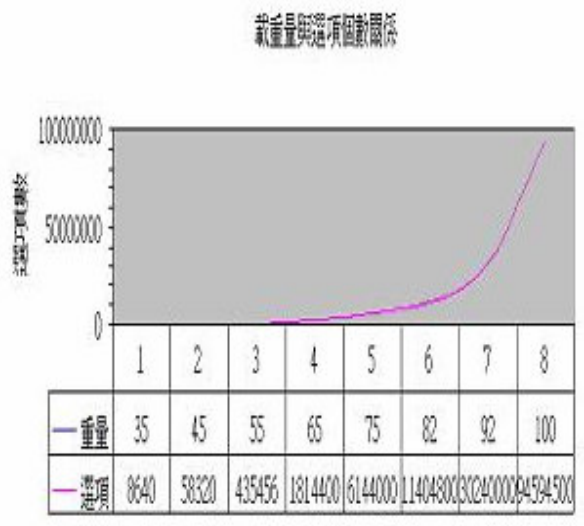

圖 8.載重量與選項相關影響曲線

依據圖 8 當載重量越大時選項越多，且呈現約重量的 3 次方強的曲線往上攀升。當搜尋空間大時，列舉法已無法計算，我們不知最佳解為何?我們和貪取(Greedy)演算法比較。在圖 9 顯示的是依據多重菁英式基因演算法與貪取(Greedy)演算法的比較。圖 9 基因演化前值代表隨機產生進行演化前的初值，實驗結果，我們提出的基因演算法的優於貪取演算法。

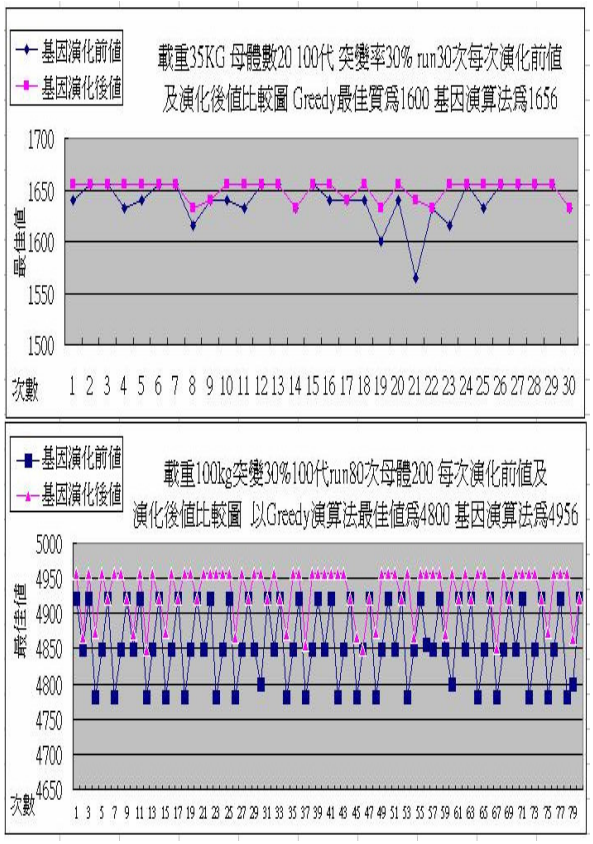

圖 9 執行基因演算與貪取演算比較

背包問題可以將之應用在投資理財上，而要求最大的報酬額度，以無限背包問題做基本的研究，在過程中我們透過基因演算的程序觀察所發生的變化，找到它具有這樣的特性，其最佳解我們利用列舉法則先依目標函數測試在所有搜尋空間中全部搜尋過所有解，先找到全域最佳解，藉以與基因演算法比較；初期本來考量應該用較高的突變率，如此在基因演算的過程中變化較多，在較小的載重量下也的確使得產生較多的最佳解次數，但在較大的載重量限制下卻成反向關係，因為在載重量較高的情況下較高的突變率變化過大如 90%在 100 代內不易收斂，要到 500 代以後才收斂，而在較低的突變率下 3%變化又不夠大，當取 30%時卻有較高的最佳值情形，如何在選項的多寡上衡量應設定多少母體數，或者執行多少次之間的比例？之間有甚麼樣的函數關係 ？如何是在這最適化下花的時間最少 ？當我們將之找出來之後 N-P Hard 將不再是問題，針對基因演算法的不保證最適化的問題也將迎刃而解。

5.結論與未來研究建議

無界限背包問題複雜度，更甚一般的背包問題。在本文論文中，我們運用菁英策略改進基因演算求解無界限背包問題。克服傳統基因演算法中收斂速度過慢的缺點。菁英策略保證每個子世代至少會和原世代的表現一樣好。另一方面我們 依問題空間複雜度 調整母體大小與執行次數，將每次的最佳值留置在精華群。最後從精華群中取出最佳者，以多重的選秀策略在廣大的搜索空間真正找到最佳解，實驗證明本方法能找出問題的最佳解。

為了再改善搜尋效能，未來擬再加入動態基因組合配置法，運用貪婪演算法的精神:每一步面臨選擇時，都做眼前最有利的選擇。如此則取 Greedy 的優點，增加最有利方向的交換率，以使基因演算法的效率更加提昇。

參考文獻

[1] 蘇木春、張孝德，"機器學習：類神經網路、模糊系統以及基因演算法則"，全華科技圖書股份有限公司，1997。

[2] 莊秉文、阮議聰，“五子棋棋略的演化學習法”，中原大學資訊工程研究所 ，2001.06。

[3] 蘇純繒、翁瑞聰，“以基因演算法求解最小化設置時間單機排程問題”，雲林科技大學工業工程與管理研究所 ，商管科技季刊 ，Vol. 5, No3, pp. 275 ~ 288，2004。

[4] 周鵬程，"遺傳演算法原理與應用"，全華科技，2005。

[5] 林豐澤，"演化式計算下篇：基因演算法及三種應用實例"，文化大學應用數學系，智慧科技與應用統計學報 2005.06。

[6] 顏榮政、鄭道明，“應用快速混亂基因演算法於營建資源分配模擬之研究”，朝陽科技大學營建工程研究所，2005.07。

[7] 吳泰熙、吳奕樺、張欽智，"以基因演算法求解單原片方形物件排列問題"，科學與工程技術期刊 vol.2,No.3,p p.75-83 ,2006。

[8] Holland, J. H.,.“Adaptive in Natural and Artificial Systems,” Ann Arbor, MI: Univ. Michigan Press. 1975.

[9] Negnevitsky, M.. Artificial Intelligence: A Guide to Intelligent Systems, ISBN 0-201-71159-1, 2002

[10] M. Negnevitsky, Artificial Intelligence: A Guide to Intelligent Systems, 2nd ed., Essex: Addison Wesley,2004.

[11] Hans Kellerer、 Ulrich Pferschy, and David Pisinger.” Knapsack Problems”, 2004.

[12] C. Zhao, W.Zhang, "Using genetic algorithm optimizing stack filters based on MMSE criterion "，in image and vision Computing, College of Information and Communication Engineering, pp. 853–860 , 2005.5.

[13] D. Pisinger,"Where are the hard knapsack problems?",Computers and Operations Research, Vol. 32,Issue 9，pp. 2271-2284. 2005.

[14] W. P. Su, “Simulated annealing as a tool for Ab initio phasing in X-ray crystallography”, Acta Cryst. A51 (1995) 845-849.

[15] KEN-LI LI 、GUANG-MING DAI 、QING-HUA LI1,"A GENETIC ALGORITHM FOR THE UNBOUNDED KNAPSACK PROBLEM",Computer School, Huazhong University of Science and Technology, Wuhan, 430074, China,Departtnent of Computer, China University of Geo Science,2003

[16] WiKipedia ,”Knapsack problem”, http://en.wikipedia.org/wiki/Knapsack problem