基於適應值分享之改良式基因演算法

游峰碩
崑山科技大學資訊管理系
助理教授
fsyu@mail.ksu.edu.tw

吳昀珊
崑山科技大學資訊管理系
研究生
suika72713@yahoo.com.tw

摘要

本研究將原始基因演算法(Genetic Algorithms, GA)進行二度改良，以菁英保留法與 Bad Genes 選擇機制互相配合，以增加染色體之間的異質性，再加入本研究所提出適應值分享機制，在演化代數初期時達到讓演算法延遲收斂之成效，而隨著代數增加演算法持續進行空間跳躍，透過此方式改善初步改良之演算法落入區域最佳解之問題，並以初步改良之基因演算法與二度改良之基因演算法運用於大專院校排課問題(University Course Timetabling Problem)進行效能、效率之比較。實驗結果証明在搜尋空間龐大且無法預測之情況下，本研究所提出二度改良之基因演算法能增進找尋趨近於最佳解之效能，並讓演算法執行效能更穩健、並有效提升演算法之效率。

關鍵詞：基因演算法、排課問題、適應值分享。

Abstract

In this research, we propose a new version of genetic algorithm based on fitness sharing in order to improve the search result. In previous study, a Bad Genes selection mechanism is implemented which improve the performance of our study of timetabling problem. Nevertheless, it still suffers the situation that the search results fall into local optimal which is common for evolutionary computations. In order to enhance the diversity of genes selection, the proposed algorithm delays the convergence of genes during evolution process by using fitness sharing. The experimental results show that our approach can avoid the local locking situation in traditional genetic algorithm. Furthermore, as the generation increased, the result shows that the proposed genetic algorithm can effectively finding the optimal solution for the timetabling problem under consideration.

Keywords: Genetic Algorithms, Timetabling Problems, Fitness Sharing.

1. 前言

基因演算法(Genetic Algorithm, GA) 為較早出現的演化式演算化之一，其基本模式源自於達爾文的物競天擇理論，生物的進化過程本身就是一個關於環境適應性的搜尋過程，模擬生物界的自然選擇與進化機制所發展出之高度平行、隨機、自適應的隨機搜尋法，並透過演化的機制隨機地逐步淘汰較差的個體，保留較好的個體，而被保留的個體使用交配(crossover)，突變(mutation)等遺傳運算子來產生優良的下一代，經過世代演化機制達到最佳化的目標。

傳統的基因演算法在演算法收斂時往往會有陷入區域最佳解的情況發生，其因可能是選擇複製運算子常常挑選到同樣的染色體，或是染色體之間異質性（heterogeneous）不高，此些現象皆是傳統基因演算法常見的缺點。因此本研究以先前作者[1]所開發之大專院校排課系統來進行效能、效率之改良，以基於多目標問題所採用適應值分享（fitness sharing）的方式，提出一種二度改良式的基因演算法，以延遲演算法收斂的速度，讓演算法有更多的可能性去搜尋其它的問題解，以期望能更趨近於最佳解，改善演算法搜尋空間不足的問題，並以未使用適應值分享機制與加入適應值分享機制之二種基因演算法進行比較。

以下各章節內容摘要分述如下。第二節文獻探討。第三節描述本研究所應用之問題。第四節介紹本文所提出之基因演算法之設計架構與研究方法。第五節對本研究之實驗結果做一歸納。第六節為本研究之結論。

2. 文獻探討

在真實的世界中，一個問題往往包含許多個目標，這些目標不能簡化且可能具有相互衝突的特性，例如當我們決策需要購買多少部機器及雇用多少人力，以達到成本最小化之情況，則需要同時考慮到機器成本與人力成本，若是單獨考慮到機器的成本，則會有人力剩餘

沒有機器可操作之情況；若是只考慮到人力成本，則可能造成沒有人員操作機器造成機器閒置之情況。因此如何在多目標的衝突情況下權衡取捨，即為多目標最佳化的重要研究課題。

演化式計算（evolutionary computation）一直廣為運用於解決這類多目標最佳化的問題，有些學者將演化式計算歸類於軟計算（soft computing）、人工智慧（artificial intelligence）或自然計算（natural computing），到近幾年則將演算式計算所建立的模式稱之為演化式演算法（evolutionary algorithms, EA）。而將多目標問題應用於基因演算法的多目標基因演算法（Multi-Objective Genetic Algorithms, MOGA）也漸漸在各領域中廣泛被應用，但演算法時常容易落入區域最佳解之缺點，針對此一問題過去有許多學者皆以不同方式改善此情況，於 1995 年 Fonseca 與 Fleming[5]將幾種演化式演算法不同適應值函數（fitness function）的給定方式做比較，如聚集方式（aggregation approaches）是考量到所有的目標值以求出一數量，再以此數量來比較解的優劣，此方式優點是可以產生一個最佳解，但使用此方式時需對於相關領域有很深的了解，而這通常是不可得的。Non-Pareto 方式是使用權重的方式將所有的目標值合併為一個單一的目標函數，使得基因演算法中的選擇運算操作能依據這個函數進行評估。另外 Goldberg[6]以建立在利基（niching）概念上的適應值分享方式，於搜尋空間之中將某一個體周圍的幾個個體歸為同一個鄰近區域（neighborhood），而此鄰近區域的每一個個體將隨著區域中的個體數量降低其適應值。

使用基因演算法於排課問題之研究首先由 Colorni 等學者[4]於 1990 年提出。Colorni 使用基因演算法嘗試建構義大利一所高中的課程表並且獲得不錯的結果。在問題的表達上，Colorni 使用一個二維矩陣表示一個課程表。在矩陣中，每一列代表一位教師，每一行代表一個時段，而矩陣中之元素值表示某課程。利用基因演算法，他們發現對於其所討論之排課問題中之硬性限制均可被滿足，但在演化過程中需要使用某些修補運算子來進行修補染色體的工作。Abramson 等學者[3]則是先考慮將教師、教室、班級、課程這四項因素預先排好，然後再使用基因演算法以求得授課時段之排程結果；研究結果顯示當面對龐大的搜尋空間時，採用基因演算法的技術在搜尋問題之極小值的表現上相當良好。Wilke 等學者[8]採用一種混合式的基因演算法求解德國某高中之排課問題；此混合式基因演算法會於演化過程中引進一種所謂的重構 (reconfiguration) 運算子。在一般情形下，該運算子不會被執行，只有當基因演算法在幾個世代後仍然無法產生高於目前最佳適應值之染色體時才會被啟動。

3. 問題描述

排課問題的研究起步於 60 年代[7]。此問題之研究重點在於如何將特定之資源分配至特定的時段，並且滿足某些特定的限制式，使得排課結果是可行且有效。許多的因素以及限制條件影響著一週課表之排定，譬如說，同一個教師不可於同一個時段排定不同的課程。類似這類造成不合理的排課限制我們稱之為硬性限制；另一方面，有些教師可能基於個人本身因素考量偏好於將其課程排定於上午或是下午，類似這類的因素我們稱之為軟性限制。一個滿足硬性限制的課程表我們稱之為可行的課程表。因此，一個可行的課程表基本上是可以被採用為實際之課程表，雖然其結果可能不會滿足所有教師的喜好。一個高品質的排課結果，不僅需要滿足硬性限制的要求，也必須盡可能地完全滿足軟性限制的要求。本研究探討之硬性限制以及軟性限制如下：

硬性限制：

HC1	每一節課只能排定一門課程
HC2	每一節課只能排定一位教師

軟性限制：

SC1	盡量避開排課於第一節與第八節
SC2	有關理科、數科之課程，盡量排於上午
SC3	同一個教師，一天排定之授課時數盡量不要超過六小時
SC4	所有課程必需連排（不跨中午）
SC5	盡量滿足教師對於時段之滿意度
SC6	同年級的共同選修課，避免各班級衝堂

另外，為了方便演算法的運算，我們將一週課程表的時段分為 40 個時段，其分配方式如表 1 所示：

表 1　一週上課之時段

星期 / 節次	一	二	三	四	五
一	1	9	17	25	33
二	2	10	18	26	34
三	3	11	19	27	35
四	4	12	20	28	36
五	5	13	21	29	37
六	6	14	22	30	38
七	7	15	23	31	39
八	8	16	24	32	40

3.1 模式建構

於本節將對本研究所要應用之問題，以線性規劃的方式來描述，以下說明本研究模式所使用的符號：

I：課程所成之集合

J：所有時段（1,2,…,40）

P：時段喜好度

i：第 i 個課程

j：第 j 個時段

t：第 t 位教師

k：第 k 個班級

L：屬於上午的時段之集合

M：屬於數理科目編號之集合

D_n：第星期 n

N：可以修課程 i 的班級之集合

T 位教師，I 個課程，40 個時段的排課問題，以教師喜好度最大化為目標：

目標函數：

$$\text{Max} \sum x_{ijtk} p_{ijtk}$$

限制式：

$$(1)\sum_{j\in J} x_{ijtk} \le 3$$

$$(2)\sum_{i\in I} x_{ijtk} \le 1$$

$$(3)\sum_{j\in J, j\notin M} x_{ijtk} \le 3 \quad where\ L=\{1,8,9,...,40\}$$

$$(4)\sum_{j\in T} x_{mjtk} \le 3 \ and \ \sum_{j\notin T} x_{mjtk} = 0$$

$$where\ T=\{1,2,3,...,40\}\ and$$

$$M=\{3,4,7,13\}$$

$$(5)\sum_{j\in D_n} x_{ijtk} \le 6 \text{ where } n\in\{1,2,3,4,5\}$$

$$(6)\ if\ x_{ijtk_1}=1\ then\ x_{ijtk_2}=0$$

$$where\ k_2 \in N\setminus\{k_1\}$$

模式說明：

(1)式為對於每一位老師於每一個班級教授每一個課程最多為課程時段上限三個時段之限制。

(2)式為對於每一位老師於每一個時段只能教授一個課程之限制。

(3)式為對於每一個課程避排於第一節課與第八節課的時段之限制。

(4)式為有關數理的科目排於上午時段之限制。

(5)式為對於每一位教師在一天授課時段不超過六小時之限制。

(6)式為對於每一共同科目，若開在班級 1，則班級 2 該時段不排課之限制。

4.研究方法

在先前作者[2]以菁英保留法與 Bad Genes 選擇兩種選擇運算子機制互相配合，而我們將此種做法稱之為隨機選取法，目的是在於讓交配運算操作時能夠產生異質性較高的染色體，其結果顯示使用此初步改良的基因演算法對於演算法之效率、效能都有很大的改良，但當問題空間再增大數倍時，若是演算法沒有在終止條件前搜尋到最佳解，則演算法往往會出現易於過快收斂，造成最佳解的適應值極低的情況，因此我們在此演算法的選擇運算子操作之前，加入適應值分享的機制，此目的在於延遲適應值的收斂，讓演算法能有更多的機會產生空間跳躍，對於搜尋空間能有較分散的搜尋。

本研究提出使用適應值分享的機制，其目的是為了避免演算法過早收斂落入區域最佳解，並且與未使用適應值分享機制之基因演算法來進行執行效能、效率之比較。其加入適應值分享之基因演算法基本的運作流程如下及圖 1 所示。

```
public void mainloop() {

      產生初始群體();
      找適應值最好的染色體();
      選擇染色體();
      進行交配();

      while (!終止代數()) {
         代數++;
         SharingFunction.sharing ();
         theSelect.select();
         交配();
         突變();
         找適應值最好的染色體();
       }
}
```

```
private void sharing () {

      for(int i=0;i<中心點個數;i++){
          center=中心點[i];
          radius =radius();//取得範圍

           If(i==0){
       //計算每條染色體與中心染色體的距離
              indexDis1=distance (center);
               //取得在範圍內的染色體
              scope1=getScope();
           }else if(i==1){
              indexDis2=distance (center);
              Scope2=getScope();
           } else if(i==2){
              indexDis3=distance (center);
              Scope3=getScope();
           }
      }

      將中心點範圍內重覆的染色體刪除();

      doSharing();

}
```

```
static int radius() {

      if(generationNum<5000){
          radius =intRandom(1, 5);
      }else{
          radius = intRandom(1, 3);
      }

return radius;
```

```
}
```

```
private int distance() {

      for(int j=1;j<單條染色體.size();j++){

          st=單條染色體該時段的排課資訊();
          t=st.該時段教師編號();
          P=教師t在時段j喜好值;

          c= center.該時段的排課資訊();
          Pt=c.該時段教師編號();
          cP=教師Pt在時段j的喜好值;

              if(P!=cP){
                  sum=sum+1;
              }
      }

      return sum;//距離

}
```

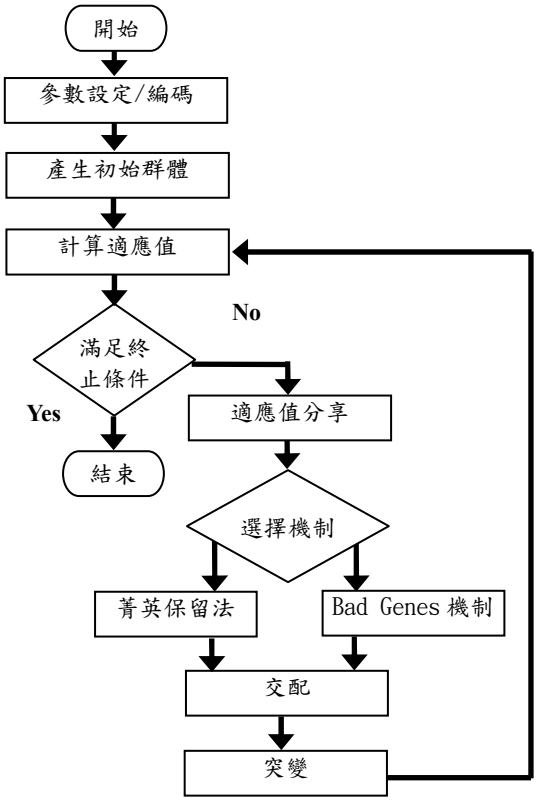

圖 1 使用適應值分享之基因演算法運作流程

傳統的基因演算法收斂速度很快，若恰巧隨機搜尋到更好的空間解，則適應值就能大幅提升，反之，若是都未搜尋到更好的空間解，而又因染色體族群之間的異質性又不高，就會造成演算法已過度收斂，演算法產生陷入區域最佳解的情況。

而我們使用適應值分享機制的目的是要降低演算法收斂的速度，把優良染色體的適應值分享給其它的染色體，一方面可把優良的染色體給壓下來，另一方面可讓其它染色體也有機會可以保留至下一世代，並且以此方法提升染色體族群之間的異質性，但若是過度分享，則又有可能造成很差的染色體卻一直留存在族群中，而沒有辦法突破區域最佳解，因此我們的適應值分享機制將以染色體之間的距離做為是否分享的基礎，避免造成優良的染色體適應值過度分享給太多的染色體。

由於本研究所應用問題的編碼非單純的數值型編碼，因此在適應值分享機制中，距離計算的方面將較為不同。我們計算族群裡每條染色體與中心點染色體的距離，再依據距離的範圍來取得欲被分享適應值的染色體編號。染色體距離計算的方式是以每一個基因內存放排課資料中的教師編號，對應到該教師編號於該時段的喜好值，以同樣的方式，取得目前中心點染色體該時段教師編號對應的喜好值並做一比對，假如喜好值相同(同為 10 或同為 0)，則略過之時段，假如喜好值不相同則做累加的動作，並且將染色體依照距離的大小從大到小做排序，接下來以設定的範圍做為基準，依序取得各中心染色體範圍內的候選染色體編號。

為了不要大幅破壞初步改良基因演算法之架構，在適應值分享選取染色體之個數的步驟時，我們在演化代數的初期設定隨機取得一至五之個數，而到後期則將範圍縮小為隨機取得一至三之個數。

在實際做適應值分享的步驟之前，需要先對於每群候選染色體做篩選之動作，若是候選染色體編號中有包括中心染色體的編號，則直接刪除該候選染色體的編號，若是候選染色體的編號有一起出現的情況，則用隨機的方式選擇一個編號刪除，以此方法來避免中心染色體或是同一候選染色體重覆分享適應值，造成機制混亂難以預測之情況，其篩選動作如圖 2 所示。

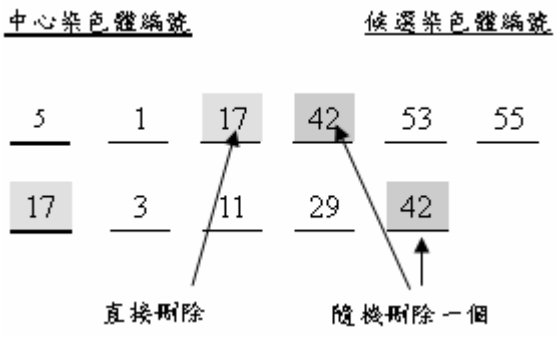

圖 2 染色體編號篩選示意圖

由於本研究所應用之問題所考量到教師喜好度較為重，因此距離是以教師喜好度來計算，若是單純以適應值來計算則可能造成優良染色體的適應值相互分享，則對於延遲演算法收斂的效果不大。

在加入適應值分享機制之後，我們預期此演算法初代會因適應值被分享產生效能會低於原來的演算法之情況，而我們期望演算法隨著代數增加能夠持續突破區域最佳解，達到提升演算法效能之效果。

5.研究成果

本研究實驗方式於染色體初始化（initialization）步驟後，複製兩份同樣初始族群的染色體，以初步改良的基因演算法與加入適應值分享機制之基因演算法分別執行，讓兩種基因演算法立足點相同再予以進行比較。

本實驗以崑山科技大學資訊管理系某學期日間部派課結果，課程為三十六門，年級數為四、每年級二個班級，並以不同教師喜好度（Preference）做測試，其演算法之相關參數設定如表 1 所示。

表 1 演算法之參數設定

	數值	
參數	初步改良基因演算法	適應值分享基因演算法
族群數目	60	60
交配率	0.5	0.5
突變率	0.5	0.5
複製菁英數	2	2
演算次代教	20000	20000
Bad Genes 選擇率	不定，由隨機選取法決定	不定，由隨機選取法決定
中心染色體個數		3

演算法程式以 JAVA 撰寫，使用之電腦配備 CPU 為 3.40GHz 及 1GB RAM。以表 1 之演算法參數執行 50 次。將使用初步改良式的基因演算法與加入適應值分享機制的演算法以各種喜好度之數量予以比較。

當各教師喜好度給予最佳喜好的總數與各教師自己授課的時數（例如教師 1 教授 3 門 3 學分的課程，則最佳喜好的總數則為 9）相等的情況稱之為 PD（Preference Degree）。圖 3 為各教師喜好度給予最佳喜好的總數比各教師自己授課的時數都多 5 門課（即為 PD5）。圖 4 為各教師喜好度給予最佳喜好的總數比各教師自己授課的時數都多四門課（即為 PD4，其餘依此類推）。圖 5 為各教師喜好度給予最佳喜好的總數比各教師自己授課的時數都多三門課的數量。而圖 6 為最佳喜好的總數多二門課的數量。圖 7 為最佳喜好的總數多一門課的數量。

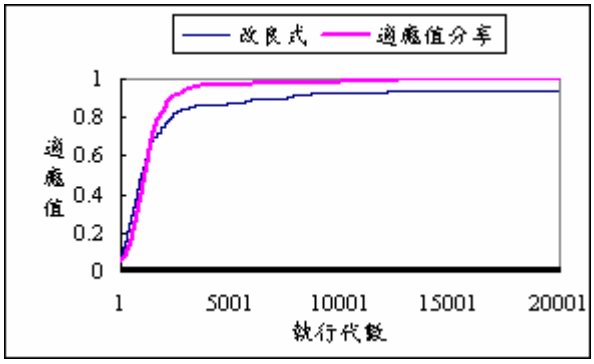

圖 3 平均適應值演化曲線圖（PD5）

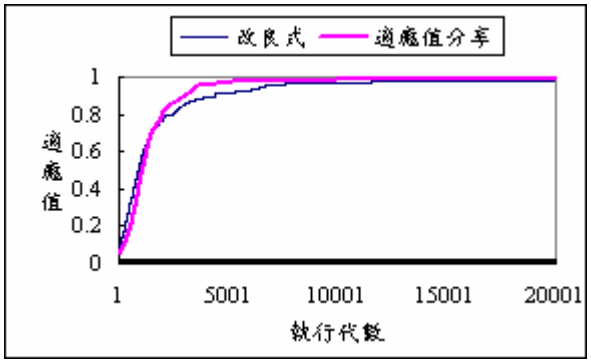

圖 4 平均適應值演化曲線圖（PD4）

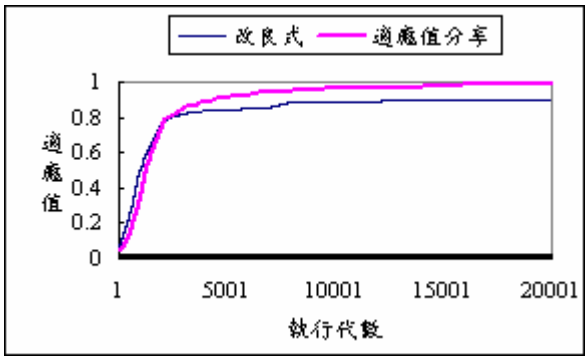

圖 5 平均適應值演化曲線圖（PD3）

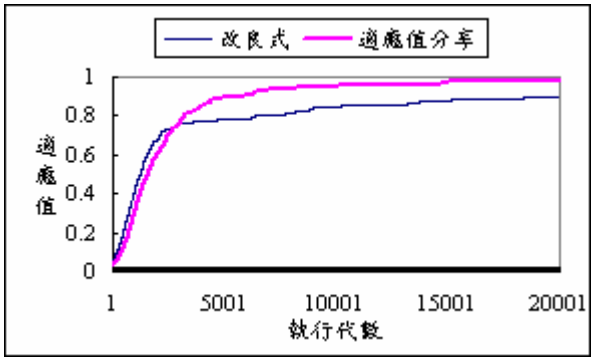

圖 6 平均適應值演化曲線圖（PD2）

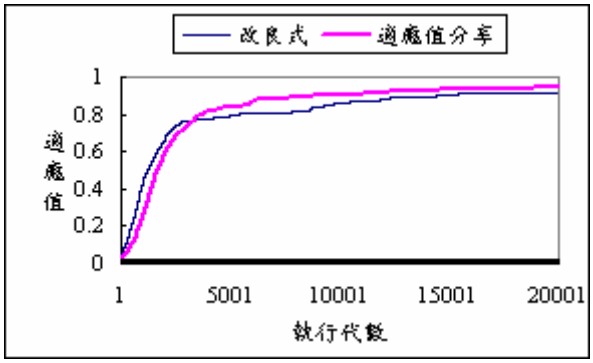

圖 7 平均適應值演化曲線圖（PD1）

由圖 3 至圖 7 可得知，使用適應值分享機制的演算法在不同情況 PD 的代數初期適應值都比未使用適應值分享機制的要低，這是由於適應值分享機制降低每代的適應值，延遲演算法適應值收斂所造成之結果，而隨著代數增加，延遲演算法收斂的效果讓演算法產生更多的搜尋空間跳躍。在 PD 限制條件越來越緊的情況，雖平均適應值都漸漸降低，但加入適應值分享機制的演算法的平均適應值都還是比未使用適應值分享機制要來的高。

另外我們以 PD3 的情況執行演算法 200 次，演算次代數為 20000 代，統計兩種演算法

到達終止代數時適應值分佈。其統計為下表所示。

表 3　兩種演算法適應值比率

	改良式	適應值分享
fitness=1	74.0%	73.0%
1>fitness>=0.9	8.5%	26.5%
0.9>fitness>=0.8	0.0%	0.0%
0.8>fitness>=0.7	2.0%	0.0%
0.7>fitness>=0.6	5.5%	0.5%
>0.6fitness>=0.5	5.0%	0.0%
0.5>fitness	5.0%	0.0%

由表 3 可看出，雖然改良式基因演算所能找尋到最佳解的次數比較高，但若未找到最佳解時則適應值會過度偏低，而加入適應值分享的基因演算法改善改良式基因演算法的此種缺陷，讓演算法能更趨於穩健、更有效率，且實驗結果証實與本研究所預測情況相同。

6. 結論

針對本研究所應用之問題空間，原始的基因演算法經過本研究的二度改良，先以菁英保留法與 Bad Genes 選擇機制互相配合，以增加染色體之間的異質性，再加入本研究所提出適應值分享機制，讓演算法延遲收斂每代的最佳解，以改善初步改良的基因演算法雖能快速找尋到最佳解，卻不平穩的情況。

在搜尋空間龐大且無法預測的情況下，本研究所提出適應值分享機制讓問題解能更趨近最佳解之成效，達到有效提升演算法的平穩性，而我們在實驗過程中發現適應值分享範圍的選取對於演算法有很大的影響，未來若能對於範圍的選取做更一步的探討與實驗，可望使演算法更增進求解之效益。

參考文獻

[1] 游峰碩、吳昀珊、王子夏（民 97）。運用基因演算法於大專院校排課之研究，全國現代管理與創新研討會（CCMI）。

[2] 游峰碩，吳昀珊，王子夏（民 97）。選擇運算子對於基因演算法效能之改良，資訊科技理論與應用國際研討會(ICTAIT)。

[3] Abramson, D. and Abela, J.,（1991 April）A parallel genetic algorithm for solving the school timetabling problem. Techincal report, Division of Information Technology, C.S.I.R.O.

[4] Colorni, A., Dorigo, M., and Maniezzo V.,（1990） Genetic algorithms nd highly constrained problems: The time-table case. In G. Goos and J. Hartmanis, editors, Parallel Problem Solving from Nature（pp. 55-59）.Springer-Verlag.

[5] C. M. Fonseca and P. J. Fleming, “An overview of evolutionary algorithms in multiobjective optimization,” Evol. Comput., vol. 3, no.1, pp. 1–16, 1995.

[6] D. E. Goldberg and J. Richardson, “Genetic algorithms with sharingfor multimodal function optimization,” in Genetic Algorithms and TheirGenetic Algorithms, J. J. Grefenstette, Ed. Hillsdale, NJ: LawrenceErlbaum, 1987, pp. 41–49.

[7] Gotlieb, C. C.,（1963） The construction of class-teacher time-tables, Information Processing, Proceedings of IFIP Congress 62 （pp. 73-77）.

[8] Wilke, P., Grobner, M., and Oster, N.,（2002） A Hybrid Genetic Algorithm for School Timetabling, Advances in Artificial Intelligence: 15th Australian Joint Conference on Artificial Intelligence, Canberra, Australia, LNCS 2557 （pp 455-464）.