基因演算法於排課問題上之研究
Research on Course Scheduler Problem By Genetic Computing

葉耀明
國立臺灣師範大學資訊教育系
台北市和平東路一段 162 號
ymyeh@ice.ntnu.edu.tw

王富民
國立臺灣師範大學資訊教育系
台北市和平東路一段 162 號
fmwang@ice.ntnu.edu.tw

摘要

排課問題是一個 NP-Complete 的問題，一直都沒有很好的解決方法。但是此問題又是每一個學校在每個學期都會碰到的一個必要行政作業，其問題的條件限制包括課程、教師、教室、班級、以及設備等複雜因素。很不容易將其轉換成電腦可以解決的程式系統。本論文提出一種改良後的基因演算法運算機制來解決此限制條件非常複雜的排課問題。我們的改良後運算機制可以自然地將排課問題在課程、教師、教室、班級、以及設備等複雜的限制條件融入基因演算法中並有效地求解。接著為了提高此系統的執行效率，我們進而將此基因演算法運算機制導入可以做並行處理的基因代理人計算的概念，並提出兩種基因代理人運算模式：Message Queue、Collection。經過效率執行評估後，我們發現多執行緒和多處理程序下的基因演算法程式確實可以提高本系統的執行效率。

關鍵詞： 排課問題、基因演算法、基因代理人計算。

Abstract

Course Scheduling Problem is an NP-Complete Problem, however, it is also a necessary administration task for every school in every semester. The constraints of a Course Scheduling Problem include complicated parameters such as courses, teachers, classrooms, classes and facilities in a school. It is very difficult to develop an efficient computer system to solve this kind of problem. This paper proposes a modified genetic algorithm to solve the Course Scheduling Problem, which can adapt these complicated parameters very easily and solve the problem efficiently. In order to improve the execution performance of the system, we also introduce genetic agent computing concept into our computing mechanism, which can provide concurrency computation through distributed system. We propose two genetic agent computing models: Message Queue and Collection. We find that the multi-thread and multi-process versions of genetic agent computing indeed can improve the execution performance of our system.

Keywords: Course Scheduling Problem, Genetic Algorithm, Genetic Agent Computing.

一． 簡介

排課的主要功能在於將教育構想有計畫的轉變為實際的教學活動。影響排課的因素歸納起來有四項：即教師、課程、學生、和教學設施。其他如法規、習慣或個人偏好等，也都構成排課的限制條件。故排課問題相當複雜，而且是動態的[1]。而 N.L.Lawrie,D.C. Wood 和 S.Even 等人證明了排課問題是一個 NP-Complete 的問題[1]13][14][15]。

電腦排課在 60 年代即有學者投入研究，如 Appleby、Csima 與 Gotlish 等人利用一些簡單的經驗法則（Heuristic Rule）來解決一些規模不是很大的問題。到了 80 年代 Selim 採用的方式[2][16][17]主要是對課程科目進行處理，首先將固定時段的課程優先排入，然後依序填入其他課程。Dowsland 的作法[2][18][19]是採用模組化，將整個排課系統分成資料輸入模組、指定模組、課表顯示模組、改進模組四個模組。Loo 的作法[2][20]是將教師、教室、及學生的可排時段設置成矩陣方式來表示，然後先將指定時段的科目先排入課表，再指定科目的優先權，並依優先權的順序將科目排入課表之中。

國內也有許多的學者研究電腦化的排課問題。例如包冬意等人[3]以啟發式法則來處理排課先期作業，再利用排課群體決策支援系統讓各相關人員來共同協商未排入的相關課程；盧坤勇[4]則是利用指派法則來處理排課問題；唐學明[1]所採用的是利用經驗式回溯搜尋（heuristic backtracking）法來解決此類問題；而邱孟佑、廖鴻圖[5]等人則是將電腦排課系統

建立於 Internet 之上。

本論文提出一種改良後的基因演算法(Genetic Algorithm)運算機制來解決此限制條件非常複雜的排課問題。基因演算法是一種多點推測的搜尋方法且可保證函數最佳化的方法。這方法可使函數收斂於廣域的最佳解，而不會掉入區域最佳解中。

近年來的基因演算法的研究中，有了和原來不同的變革，例如合作式演化的基因演算法（co-operative co-evolution genetic algorithm, CCGA），就是一個族群之中包含了許多種（species）或子族群(sub-populations)，其中每一個種類代表了問題的一個變數或是一部份需要被最佳化的。當所有的種被整合在一起即為整個問題的完整解[6]。AGUIRRE 等人則提出基因運算的合作式模組（Cooperative Model for Genetic Operations）去改善基因演算法[7]。T.Matsumura 等人也針對多處理機提出了在平行分散式的基因演算法（Parallel and Distributed Genetic Algorithm, PDGA）中染色體遷移的影響[8]。K.Kojima 等人則提出了網路的基因演算法（Network based Genetic Algorithm）[9]，將其加入了代表團管理模組（delegation management model），目的是希望降低網路通訊的負荷。而 Nicolas Barnier 等人則針對限制滿足的問題[10]中，將每個限制滿足的條件分成不同的變數，針對不同的變數再另作處理。

為了提昇基因演算法的執行效率，我們提出結合代理者計算(Agent Computing)的概念到基因演算法中。所謂代理者（Agent）[11]，是一個實體（可能是電腦或人類）具有完成我們需求的能力，而代理者和代理者間可相互影響。代理者主要概念在於其具有某部分的自主能力且彼此間能夠相互影響。並且可能同時存在於一個處理機中或是多處理機中。例如 Etzioni 和 Weld 所提出的 Sofobt 提供了網路使用者在網際網路上搜尋資料的個人智慧化輔助[11]。Krulwich 則設計 BarginFinder 幫助 CD 購買者在網路上搜尋最好的價錢。而 Maes 則描述一系列的代理者包含了 Maxime（電子郵件過濾代理者）、MCL（會議排程代理者）、NewT（新聞過濾代理者，可讓使用者從網路上獲得所感興趣的新聞資源）。

二、 排課問題

所謂排課問題就是針對教學的對象(即學生)，安排所需研習的課程且上課的時間與場地，在同一時間上不會發生授課教師或上課教室上的衝突，並且可以滿足教師、班級、課程的需求。換句話說，如何解決教師、課程、班級、教室、與課程的設備需求間的相互關係以求得最好的結果即為所謂的排課問題，以下針對這五項主要因素的需求做一個描述：

（1）教師：必須考慮到教師的專長（也就是可任教的科目）、教師所必須任教的最低鐘點數、教師是否習慣連堂上課及教師上課的合適時段。

（2）班級：必須考慮該班的學生人數（以便找尋可容納該班級所有學生的教室）及該班所必須修習的課程。

（3）教室：必須考慮教室的大小及教室所包含的設備。

（4）課程：必須考慮課程的合適時段、課程是否連堂上課、課程是否為許多班級一起修習的共同課程、課程間的是否避免時間互斥性及哪些課程需要哪些設備。

（5）設備：必須考慮到學校中所擁有何種設備及數量。

在排課之初，排課者必須要知道上述的需求，但是為了方便排課的進行，我們先將一些條件予以事先處理，得到（1）課程-教師、（2）課程-課程、（3）課程-班級、（4）教室-設備的四種關係。而由此可以很明顯的知道，哪門課可由那個老師任教且可被安排在哪一間教室，並且不可和哪些課程排在同一時段。除此之外，我們將需求分成必要性條件(necessary condition)及滿足性條件(satisfied condition)兩種條件來討論。

1、必要性條件（necessary condition）：

所謂的必要性條件就是在現實情況之下，絕不能違反的原則，茲列於下：

1.1 教師必要性條件（NT）

（NT1）教師鐘點數： 教師每週上課的時數必須大於或等於其系所要求的最低時數。

（NT2）教師時間非重疊性： 除共同課程時間外，同一位教師不可同時間對一個班級以上授課。

1.2 課程必要性條件（NO）

（NO1）共同課程一致性：修習共同課程的班級必須在同一時間、同一教室上課。

1.3 教室必要性條件（NR）

（NR1）教室大小。

（NR2）教室時間非重疊性。

（NR3）教室設備必要性。

2、滿足性條件(satisfied condition)：

所謂的滿足性條件即是受限在現實環境之下，排課的角色仍可忍受的條件。茲列於下：

2.1 教師滿足性條件（ST）

（ST1）教師合適時段。

（ST2）教師不合適時段。

（ST3）教師連堂上課需求。

2.2 課程滿足性條件（SO）

（SO1）課程合適時段。

（SO2）課程不合適時段。
（SO3）課程連堂需求。
（SO4）相異課程時間互斥性：在同一時間內互斥的兩門課不可排入同一時段，而該互斥性沒有遞移性的關係。

所以整個排課的最終結果是希望能夠得到每個班級所修習的每一門課的上課時間、上課教室及上課的教師，並且彼此間不互相衝突。換句話說，我們希望在符合上述必要性條件的解中去找尋較能符合滿足性條件的解。

三・基因演算法與其運算模組

基因演算法是 John Holland 在 1975 年所提出。剛開始時，是在解集合中選擇一組被稱為族群（population）之隨機解，在族群中的每一個個體（individual）均被稱為染色體（chromosome），而這染色體所表達的就是我們所求問題的一個解。染色體透過一連串的疊代（iteration）的程序，形成了新的世代（generation）。在每一世代當中，每個染色體利用一些標準來做評估，以產生相對應的適應值（fitness value），利用此適應值來篩選父代（parent）及子代（offspring）中較優良的個體來作為下一世代。所謂子代（offspring）的染色體，是藉由兩種方式來形成：一是藉由交配（crossover）的機制，結合目前世代中的兩個染色體；一是藉由突變的機制，改變染色體中的字串。在篩選過程中，擁有較好的適應值就會有較高的機率被選擇為下一代的染色體，以符合生物學上優勝劣敗的原則。經過多次世代演化之後，其結果會向較佳的染色體收斂，而這就是我們所希望的較佳解或次佳解 [12]。

所以我們針對上述的問題提出一個改變原基因演算法的機制，將修正過後的演算法分成初始化機制、基因運算機制、評估機制、調整機制、遷移機制及演化機制。在討論這六個機制之前，先介紹如何將排課問題中所需之要件化成染色體的形式。

A. 染色體的結構

在運用基因演算法的時候，在一個染色體就是一個問題解的原則下，將一個染色體安排為所有班級的一週課表。所以其安排方式如下：

（一）染色體的表示法：

每一個染色體所代表的是一種排課方式，而基因則是由每個班級來表示。以下式子顯示染色體的結構，其中 X 所代表的是一個染色體：

$$X = \{x_1, x_2, \ldots, x_n\},$$

其中 x_i 所代表的是第 i 個基因(班級 i) (公式一)

（二）基因的表示法：

每個基因所代表的是一個班級一週的總課堂數。每堂課 t_j 代表為該班（該基因）在第 j 個時段的上課課程，故每個基因 x 的結構可以由下列式子表示：

$$x = [t_1, t_2, \ldots, t_m],$$

其中 t_i 所代表的是該班的第 i 個時段 (公式二)

在染色體中每個不同基因的時段，是以字元來表示，其順序是以大寫字母 A～Z、小寫字母 a~z、數字 0~8 的順序記錄了課程（數字 9 表示該堂為空堂）。

B. 運算模組

以下則針對改良後的基因演算法的六個模組化機制做一個說明，其演算法如圖 1，架構圖則如圖 2：

(一)、初始化機制(Initiation operation)：

在染色體的初始化上，對於其選擇染色體中的基因是採用隨機選擇，但在其他要素的互動上，所採用的是經過有選擇性的選出能配合這染色體的要素。

以排課來例，排課主要為課程、班級、教室、教師四個要素的互動關係，若染色體的形式以班級及課程為主，教室及教師就成為需要配合的要素。所以首先和傳統的基因演算法一樣必須隨機選擇出一組染色體，也就是隨機取出一組合理的班級與課程的對應關係。再來則必須針對這組隨機選擇的班級課程，有選擇性的找出合理的教師和教室與該課程對應。

(二)、基因運算機制(Genetic operation)

基因運算機制主要分為交配（crossover）與突變(mutation)兩部分。在我們的排課系統中的交配及突變運算如下所述：

（1）交配

在基因演算法中，one-cut-point 交配的方式通常是以基因為單位做交換的動作，以產生新的子代。但在此提出的方法，和原有的方式略有不同，因為在本系統中的基因並非只是單純的一個值而是一個字串，若只是單純的對基因做交換的動作，在基因數少的情況之下，可能不會產生多大的演化效用，故本交配的方法在隨機取分割點（cut point）的時候，並不以基因為單位，而以課堂為單位，取的分割點可能位於基因之上，造成切割基因的效果，以達到改造基因的目的。此法的表示如下：

假設兩個父代分別為 X 及 Y，其中

X＝{[a, b, c],[d, ↓e, f],[g, h, i]}

Y＝{[c, b, a],[f, ↓d, e],[g, i, h]}

若取的分割點剛好位於第 2 個基因的第一個時段之後，則其子代分別為 X'及 Y':

X'={[a, b, c],[d, ↓f, e],[g, i, h]}

Y'={[c, b, a],[f, ↓d, e],[g, h, i]}

上述情形，針對有完整基因的部分，和原有的 one-cut-point 一樣，純粹做交換的動作，但對被切割的基因，採用部分保留而另一部份則參照另一父代染色體所沒有的部分依序填入。

（2）突變

突變是針對要突變的基因做改變，其改變方式為將該基因的字串重新做一次隨機排列。突變主要根據兩個數值：突變代數及突變個數。所謂的突變代數是指多少代突變一次，而突變個數是指所有可能突變的基因個數乘上突變機率即為要突變的基因個數。在本系統中，為了要獲得較好的染色體，也就是要獲得較好的適應函數值，故保留最好的染色體不予突變。

(三)、評估機制(Evaluation operation)

評估機制包含了兩個部分，一個為適應函數（fitness function），另一個為限制函數(constraint function)。適應函數的計算是針對染色體去求出適應值（fitness value），根據這適應值的大小得知此是否為較佳解。限制函數的計算是針對染色體去求出限制值（constraint value），以得知此染色體是否為合理染色體(解集合中的一個元素)。以下就本排課系統為例，說明適應函數及限制函數的做法：

（1）適應函數

適應函數主要針對合理染色體的去計算適應值，並對滿足性條件作分析。如果該染色體符合其條件的話，則其適應值為該適應函數值之累加。其定義如下：

若欲解決之問題有其非必要限制條件 $S_1,S_2....S_n$ 且相對應的適應函數為 $f_1(X), f_2(X), ..., f_n(X)$，其中 X 代表的是欲評估的染色體，則該染色體的適應值(fitness)如公式三。

$F(X)=w_1*f_1(X)+w_2*f_2(X)+...+w_n*f_n(X)$，其中 $w_1, w_2,...,w_n$ 分別代表各條件之權重。(公式三)

就排課系統的七個滿足性條件（ST1～ST3、SO1～SO4）相對應的適應函數分別為 f_1~f_7，針對每一個染色體 X，其適應值 F（X）為 $w_1*f_1(X)+w_2*f_2(X)+...+w_7*f_7(X)$，其中 w_1 ～w_7 為使用者所輸入相對應的滿足性條件的權重值。以下分別說明各函數之作法：

（f_1）教師合適時段的適應函數：針對滿足性條件（ST1）。計算方式為該染色體中，有多少時段是該課程的任課教師的較合適時段，其值為符合時段（time slots）數,其值為正。

（f_2）教師不合適時段的適應函數：針對滿足性條件（ST2）。計算方式為該染色體中，有多少時段是該課程的任課教師的不合適時段，其值為符合時段數乘上-1,故其值為負。

（f_3）教師連堂上課需求的適應函數：針對滿足性條件（ST3）。計算方式，該染色體中的每一個課程的任課教師是否滿足該教師的連堂需求，其值為滿足連堂需求的課程數,故其值為正。

（f_4）課程合適時段的適應函數：針對滿足性條件（SO1）。計算方式為該染色體中，有多少時段是課程位於較合適時段，適應函數值為符合時段數,故其值為正。

（f_5）課程不合適時段的適應函數：針對滿足性條件（SO2）。計算方式為該染色體中，有多少時段是課程位於較不合適時段，其值為符合時段數乘上-1,故其值為負。

（f_6）課程連堂需求的適應函數：針對滿足性條件（SO3）。計算方式為該染色體中的每一個課程是否滿足連堂需求，換句話說，適應函數值為滿足連堂需求的課程數,故其值為正。

（f_7）相異課程時間互斥性的適應函數：針對滿足性條件（SO4）。計算方式為針對該染色體中的每個基因中的每個 time slot，計算不同基因同一相對的 time slot 位置的互斥的課程數。其適應函數值即為該課程數乘上（-1），故其值為負。

（2）限制函數

限制函數主要針對所有子代染色體的去計算限制值，也就是針對必要限制條件作分析。若該染色體滿足其必要性的限制條件，則可得知該染色體為合理的染色體，同時也是該問題的解集合中的一個元素。其定義如下：

若欲解決之問題有其必要條件 $N_1,N_2,...,N_n$ 且相對應的限制函數為 $g_1(X), g_2(X),..., g_n(X)$，其中 X 代表的是欲評估的染色體，則該染色體的限制值(constraint value)如公式四。

$G(X)=w_1*g_1(X)+w_2*g_2(X)+...+w_n*g_n(X)$ ，其中 $w_1, w_2,...,w_n$ 分別代表各條件之權重。而一個合理的染色體 X 必須滿足為 $G(X)<C$ （C 為一個定值）。(公式四)

就排課系統的六個必要性條件（NT1、NT2、NO1、NR1～NR3），相對應的限制函數分別為 g_1~g_6，針對每一個染色體 X，其限制值 G（X）為 $w_1*g_1(X)+w_2*g_2(X)+...+w_6*g_6(X)$，其中 w_1 ～w_6 為使用者所輸入相對應的限制性條件權重值。以下分別說明各函數之作法。

（g_1）教師鐘點數的限制函數：針對必要性條件（NT1）。計算方式為計算該染色體中，有多少教師的開課鐘點數小於其下限，則其限制函數的值為該教師數。

（g_2）教師時間非重疊性的限制函數：針對必要性條件（NT2）。計算方式為計算該染色體中的每個基因的相對位置，出現其同一教師的個數。

（g_3）共同課程時間一致性的限制函數：針對必要性條件（NO1）。計算方式為針對該染色體的每一個班級（基因），若該班級具有共同課程但該課程的上課班級數不滿必須上課的班級數，則其值為該課程所欠缺的班級數。

（g_4）教室大小的限制函數：針對必要性條件（NR1）。計算方式為計算該染色體中，有多少課程的教室大小無法容納預定的修課人數，其值為該課程數。

（g_5）教室時間非重疊性的限制函數：針對必要性條件（NR2）。計算方式為計算該染色體

中每個基因的相對位置，在非同一課程卻出現同一教室的個數。

（g_6）教室設備必要性的限制函數：針對必要性條件（NR3）。計算方式為對每一個的已開之課程計算其教室的設備是否滿足其需求，其值為不滿其需求的設備數目。

所以在此我們可以很明確的得知，所謂的排課問題，就是要在限制值最小的染色體中，找出最大的適應值的染色體，此染色體就是我們要找的解。

(四)、調整機制(Adjustability operation)

通常在過多的限制條件的情況下，很容易產生不合理的染色體(irregular chromosome)，故針對此類交配或突變之後的不合理的染色體或針對較不容易收斂的限制條件，利用經過設計好的調整函式(adjust function)來改良染色體中基因的排列方式，以求得較優良的染色體，也就是運用調整函式來幫助我們盡快求出所需的解，

調整機制中的調整函式大體上可分成兩種，一種是將不合理的染色體調整為合理的染色體，另一種則是針對染色體中不易收斂的限制條件，來做調整，使該染色體較符合其問題需求。調整函式 Ad 的定義為 Ad（X）= X’，其中 X 為經過評估機制後被認為有需調整的染色體，X’為調整後的染色體，在調整機制中不同的調整函式並沒有相關性的運算關係。以下針對排課問題來說明調整機制。

在本排課的系統之中，除了不合理的染色體子代（也就是不合理解）需要調整外，就是本系統處理連堂的問題，因為基因演算法大部分採用隨機的運算方式，較難將一門課的所有時數排在一起。故在此將會使用調整機制的功能，將其調整至理想的染色體。以下就說明各調整函數的作法。

（1）教師的調整

針對教師的必要性條件有兩個，一個是教師的鐘點數(NT1)，一個是教師時間非重疊性(NT2)。

對於教師開課的最低鐘點問題，首先藉由評估機制去判斷該染色體是否存在有教師開課的鐘點數小於到其最低鐘點數。調整函式的作法為尋找該教師的可任教科目中那一個已開課的任課教師開課的鐘點數最多，在不影響其他衝突的情況之下，將其取代原開課教師，否則往下找另一個課程。

對於教師時間非重疊性的問題，一樣是藉由評估機制去判斷該染色體是否存在有時段具有教師衝突的情況。調整函式的作法為針對該染色體中的時段分別比較具有教師衝突的課程，選擇教師衝突的堂數較多的課程，則另外擇一可任教的教師，若其衝突課程的的衝突堂數一樣多時，則選擇其可任教教師較多的課程，擇另一可任教之教師。

（2）教室的調整

針對教室的必要性限制條件有三個，一個是教室大小(NR1)，一個是教室時間非重疊性(NR2)，另一個則為教室設備必要性(NR3)。

對於教室大小的問題，找出該染色體中教室大小無法容納該課程學生人數者，替換成其他可容納該課程人數的教室。

對於教室時間非重疊性的問題，將要調整的染色體，依次比較每個基因的每個時段，看其教室是否相同，若在該基因的時段發現其教室相同，則在其衝突的基因中，隨機取出一基因調整該時段的教室，去選擇其另外符合需求的教室。

對於教室設備必要性的問題，利用評估函式可得知那些時段，教室設備無法滿足該課程，找尋該時段可滿足此課程的教室來替換原教室。

（3）共同課程的調整

其限制性條件是共同課程時間一致性（NO1），作法是將要調整的染色體隨機依次取出基因，且將該班的共同課程與其他班級比較是否具有相同一門課，若有，則調整其他班共同課程的時段與該班相同，而被調整的基因其被置換掉的字元代碼，則填入原來的共同課程字元代碼的位置。

（4）連堂問題的調整

針對連堂問題所針對的滿足性條件有兩個：一個是教師連堂上課需求（ST3）；另一個是課程連堂需求（SO3）。採用的方式為假定該課程為要求連堂方式，且包含了教室的要求或課程的要求，此時視其課程是否上一堂或下一堂是否為空堂，若是則計算將該課程位於其他的時段移入該空堂位置的適應值，再去比較原來的適應值何者較高，若為後者較高，則移動該課程。

(五)、遷移機制(Migration operation)

多族群的基因演算法，在 80 年代中，即有學者提出，認為多族群的基因演算法效果會比單一族群的基因演算法效果較好，這好比近親通婚所生出的下一代會生出畸形兒的機率會遠大於非近親通婚所生出的下一代的道裡一樣。在此針對融入了一個新的機制-遷移機制（ Migration operation）。主要作用在接受其他族群的較優良個體，融入本族群之中，在該機制存有遷移函式（Migration function）有兩種形式，一種是選擇其他族群的優良染色體進入本族群中進行演化，另一種是將本族群之優良染色體給其他族群所用，如下所示：

（1）將本族群第 t 代較好的 num 個染色體送至族群 i 的 Queue 中。其函式表示為 $MF_{out}(Generation(t),num\ ,Population_i)$

（2）將本族群接獲至其他族群的 Queue 中的染

色體取出，和第 t 代的染色體一起加入演化並清除該 Queue。其函式表示為 MF_{in}(Generation(t))。

在圖 2 的架構圖中，遷移機制與演化機制是以雙箭頭表示，是因為演化機制所選出之優良個體還是要給其他的族群所用。

(六)、演化機制(Evolution operation)

在演化機制中，主要是從父代及子代的染色體中選擇出下一世代的染色體。在本排課系統中，從父代及子代中選出染色體限制值不超過我們的上限，且適應值較大者進入下一世代演化，用此方式的留下個數和原父代相同的且限制值符合我們的需求且適應值會越來越好的染色體。

```
Procedure: Multiple_Thread Genetic Algorithm
begin
    t←0;
    Initiation operation [ Generation(t) ];
    Evaluation operation [ Generation(t) ];
    while (not termination condition)
    begin
       Genetic operation [ Generation(t) ]
                          result Offspring(t);
        Evaluation operation [ Offspring(t) ];
        Adjustability operation [ Offspring(t) ];
        Migration Operation[ random select other
              population ]result Foreign(t);
        Evolution operation
          [ Generation(t),Offspring(t),Foreign(t)]
                          result Generation(t+1);
        Migration Operation[ Generation(t+1) ]
        t←t+1;
    end
end
```

圖 1.改良後的基因演算法

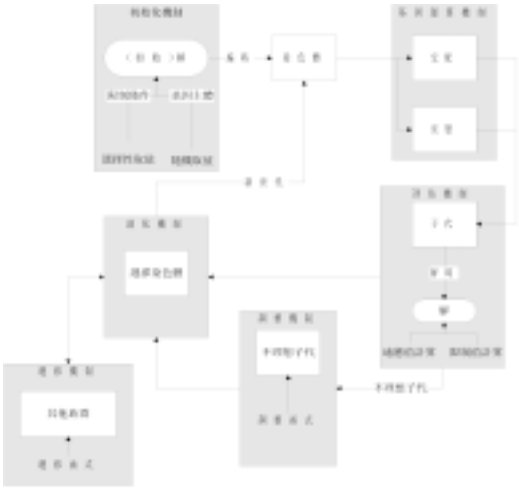

圖 2.改良後的基因演算法的架構圖

四．基因代理人計算

為了將多族群的基因演算法充分應用在分散式系統的環境之下，我們提出一種基因代理人的方式，也就是將基因演算法分別給予不同的執行緒或是不同的機器上來執行，給個執行緒或是每台機器中都至少有一個可獨立運作的基因演算法在執行。在此，我們將針對遷移機制來詳加討論如何處理族群間的合作。

遷移機制中的個體遷移，我們是主要是以 QUEUE 的方式來處理，其處理的方式為在每一個基因演算法中，設有一個 QUEUE，對其外來族群的個體要進入本族群中，會先進入 QUEUE 中等待，當該族群到了需要外來個體一起進入本族群演化之時，就從 QUEUE 中，抓取其中的染色體，進入本族群的演化機制，並清除 QUEUE 中的值。而針對本族群的個體要遷移至其他的族群演化，則在一定代數後將本身較優良的染色體，送至其他族群的 QUEUE 中等待。換句話說，對於本族群外的基因運算，其運算結果都可幫助本身族群的演化，故都可稱為本族群的基因代理者。對於這種基因代理人的機制我們將其分成兩種，分別為 Message Queue Process 及 Collection Process，以下分別討論：

（1）Message Queue Process:

每一個基因運算代理者都和本身族群站在平等的地位，本身族群必須將自己的優良染色體提供給其他的代理者，同時也獲得其他基因運算代理者的優良染色體，換句話說，不管是本身或是其他代理者都存有一個 message queue 以處理外來的個體。此種方式，對於採用單機上的多執行緒效率較為好，因為若是溝通於網路之上，可能因為個體的遷移過於平凡而使的整體的效能下降。

（2）Collection Process

對於整體系統而言，只有一個主要族群中含有 Queue 來處理外來個體，其他的基因運算代理者，只是一個幫助主要族群演化的助手，換句話說，主要族群只單方面接收基因代理者的優良個體，而不將本身的優良染色體遷移出去，且除了主要族群之外，其他的基因代理者並不互相溝通。此種方式，較適合於分散式環境的多處理機，因為會比 message queue 的方式所需的網路的負荷還要小很多。

為了將基因代理人計算的方式運作於分散式系統上，我們在此先提出一個代理者系統（RPOS），此系統的環境的主要目的是將主要族群的基因代理人複製到其他主機之上，換句話說，代理者系統的主要功能在於負責傳遞基因運算物件（將主機中的排課代理者複製到其他機器上），並啟動該物件。使得基因代理人計算具有自動自發的功能。基因代理者系統本身是以 CORBA 及 JAVA 來實作，其架構主要分成三部分，架構圖如圖 3：

（1） RPOS.Server.RPOServer：接收 Client 端的物件並複製於本機上，並依 Client 的需求，在本機上啟動該物件。

（2） RPOS.Client.RPOClient：向 Server 端傳送物件，並傳送啟動該物件的訊息給 Server 端。

（3） RPOS_CORBA：使 PROS 的 Client 端能使用 Server 端的物件，也就是藉由 Server 的物件使得 Client 的訊息及物件能傳送到 Server 端。

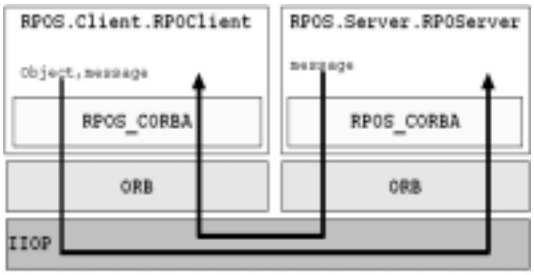

圖 3.RPOS 架構圖

有了 RPOS 的系統之後，對於我們的分散式下的環境的基因運算的步驟如下：

步驟 1：將代理者複製到可運作的機器上。

步驟 2:主機及代理者分別啟動基因演算法(其運算所需的資料透過 http 協定向主機讀取 XML 檔案)。

步驟 3：每個代理者每代運算最佳的染色體傳回主機的存放染色體的 Queue 中。

步驟 4：主機的基因演算法每經過一定代數，比較本身及 Queue 的染色體，選擇較優者染色體進入下一代演化，並清除存放染色體的 Queue。

整個系統的架構圖如圖 4 所示：

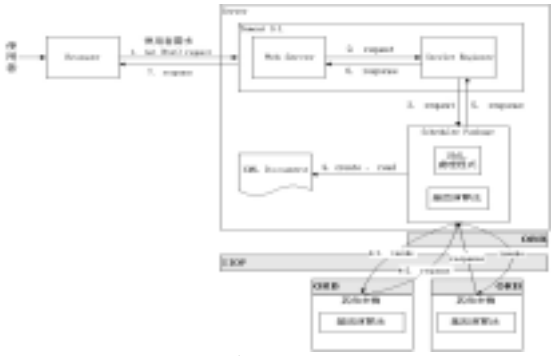

圖 4.系統架構圖

五・效能評估

我們發展的排課系統中針對運用單執行緒、多執行緒及多處理程序環境下的基因演算法進行評估與分析。

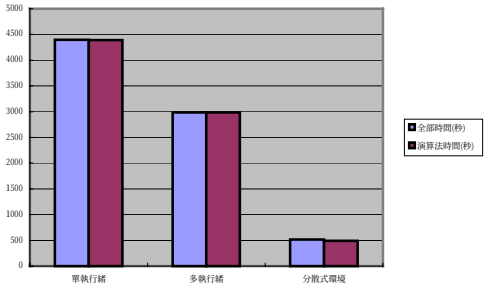

圖 5.單執行緒、多執行緒及多處理程序執行時間比較

由圖 5 可明顯看出在染色體個數＝40 時，分成 10 個族群的情況之下，每 5 代做一次遷移時，當個體越大時，其時間差距越大。其中單執行緒與多執行緒的時間差距大約 1～1.5 倍，而多執行緒與分散式環境的時間差距為 5～6 倍。

接著我們針對執行效率及適應值的收斂來兩方面來加以討論。

(一) 執行效率

1.分散式環境的執行效率大於多執行緒。多執行緒的執行效率大於單執行緒。

2.當個體越大時，其運算時間越長，但是其單執行緒、多執行緒及分散式環境下的時間差仍成一定的比例，不會因個體大小影響。

3.當每一世代的染色體個數越多，則運算時間越長，但對單執行緒與多執行緒的時間比例上仍是相同的差距。如在對多執行緒及分散式環境下的運算結果，則會因為染色體的個數增加，而有所成長。

4.對遷移越頻繁而言，當越頻繁，則所需的時間越長，但是對多執行緒或是分散式的環境而言，時間差距並沒有改變多少。

(二) 適應值的評估

1.對單執行緒、多執行緒及分散式環境的適應值影響最明顯的是染色體個數，當染色體個數越多時，則其值越快收斂。

2.多執行緒及分散式環境下的適應值在剛開始的時後，可能會比單執行緒較差，但是在經過一段的演化之後，反而會比單執行緒得出較好的結果。

六・結論

本論文提出一種改良後的基因演算法運算機制來解決限制條件非常複雜的排課問題，進而將此基因演算法運算機制導入可以做並行處理的基因代理人計算的概念，並提出兩種基因代理人運算模式：Message Queue、Collection。

針對修正後的基因演算法及基因代理人運算的系統，我們提出幾點結論：

1.利用將基因演算法的模組化機制方便於討論及修改原來基因演算法。
2.在評估機制中，利用兩種不同的評估方式（適應值及限制值），可方便篩選出優良的染色體。
3.利用調整機制的修正染色體將可使得染色體演化的過程中不易出現不合理的解。
4.在多族群的環境下，利用遷移機制，方便於族群之間的染色體交換。5.在分散式環境下的多族群合作方式，採用了區域網路中 Agent 合作的方式求出解，在未來發展上，可透過 SOAP 的方式，將合作群擴展至 Internet 上。
6.在本研究中的 Agent，在未來發展上，應可擴展為自行整合資料的能力。
7.在本研究中的 Agent，在未來發展上，可擴展為具有自行複製的能力，以減少通訊的時間。
8.本研究的解決排課問題是建立在基因演算法上，由於基因演算法本身並不具有回饋的機制，故未來發展上，可以利用具有回饋機制的螞蟻演算法，應可更快速解決排課問題。

七・參考文獻

[1]. 唐學明,"軍事學校電腦排課問題之探討", *復興崗學報 59 期，1996*。

[2]. 劉明淵,"電腦在排課作業上的應用-問題的性質與幾個系統的作法", 資訊與教育雜誌 1993.4.

[3]. 包冬意,賴永進,吳智輝,"大專院校排課自動化之研究",大葉學報 2（1）：p.135-144,1993。

[4]. 盧坤勇,"電腦化排課系統指派法則探討", 聯合學報 12 期. p107- 116 。民國 83.11。

[5]. 邱孟佑,廖鴻圖,"植基於 Intranet 上之電腦輔助排課系統",德明學報 12 期,頁 1-13,, 民 86.3。

[6]. N.Chaiyaratans and A.M.S.Zalzala, "Recent Developments in Evolutionary and Genetic Algorithm:Theory and Applications", *Genetic Algorithms in Engineering Systems: Innovations and Applications,2-4 September 1997, Conference Publication No.446, @IEEE,1997*

[7]. Hayatt Regency Bethesda, "Cooperative Model for Genetic Operators to Improve GAs", *IEEE International Conference on Intelligence, Information and Systems Nov. 1-3, 1999.*

[8]. T.Mastsumura, M.Nakamura, J.Okech ,"Effects of Chromosome Migration on a Parallel and Distributes Genetic Algorithm", *Proceedings of the 1997 International symposium on Parallel Architectures, Algorithm, and Network, 1997. (I-SPAN '97).*

[9]. Kazunori Kojima, Wataru Kawamata, Hiroshi Matsuo and Masaaki Ishigame , "Network Based Genetic Algorithm", Proceedings of the Seventh International Conference on Parallel and Distributed Systems: Workshops (ICPADS'00 Workshops) Copyright (c) 2000 Institute of Electrical and Electronics Engineers, Inc. All rights reserved.

[10]. Nicolas Barnier, Pascal Brisset, "Optimization by hybridization of a genetic algorithm with constraint satisfaction techniques", *1998 IEEE http://iciis99.cs.unr.edu/schedule.html.*

[11]. Caroline C. Hayes, ”Agents in a Nutshell-A Very Brief Introduction”, IEEE Transactions on Knowledge and Data Engineering, VOL 11, NO.1,January/FEBRARY 1999.

[12]. M.Gen , R.Cheng,"Genetic Algorithm & Engineer Design",*Willey-Interscience Publication,pp.2-4,1997*

[13]. N.L.Lawrie, "An Integer Linear Programming Model of a School Timetabling Problem", The Computer Journal, Vol.12, pp307-316. 1969.

[14]. D.C.Wood, "A Technique for Colouring a Graph Applicable to Large Scale Timetabling Problem", The Computer Journal, Vol.12, p317-319. 1969.

[15]. S.Even, A.Itai, A.Shamir, "On the Complesity of Timetable and Multicommodity Flow Problems", 16th IEEE Annual Symposium on Foundatiions of Computer Science, pp.184-193, 1975.

[16]. Selim,SM,"An algorithm for constructing a University faculty timetable". Comput. Edue., 6. pp.323-332. 1982.

[17]. Selim,SM, "An algorithm for producing course and lecture timetable". Comput. Edue., 6. pp.323-332. 1983.

[18]. Dowsland,W.B. and Lim,s"Computer aided school timetabling - part1: the history of computerised timetabling", Compute Education,pp22-23. 1982,Nov.

[19]. Dowsland,W.B. and Lim,s,"Computer aided school timetabling - part2: the micro-computer for school timetabling", Compute Education,pp2-4. 1982,Nov.

[20]. Loo,E.H. Goh,T,N. and Ong, H.L., " A heuristic approach to scheduling university timetables.", Comput. Edue. 10(3),pp.397-388.1986.