遺傳演算法於 Job Shop 排程問題上的研究

陳宜欣　陳稼興

國立中央大學資訊管理系

許芳誠

真理大學資訊管理系

摘　　要

近年來具高問題獨立性、強大解答搜尋能力的遺傳演算法，已成為各領域的新寵，本研究以 NP-hard ordering**問題中的經典問題** JSP(job shop problem) **來印證遺傳演算法可以在搜尋解答上擁有很好的表現。我們除了針對排程問題的特性提出一個合適的染色體表示法，並證明其解答空間保證包含最佳解外，還提出一個借自於大自然運行觀念的改良式退火函數，用來調整替換條件，並以實驗驗證這些方法的效果。**

關鍵詞：遺傳演算法、job shop **排程問題、退火方法。**

APPLICATION OF GENETIC ALGORITHMS ON THE JOB SHOP PROBLEM

Yi-Shin Chen　Jiah-Shing Chen

Department of Information Management
National Central University
Chungli Taiwan 320, R.O.C.

Fang-Cheng Hsu

Department of Information Management
Aletheia University
Tamsui Taiwan 251, R.O.C.

Key words: genetic algorithms, job shop problem, annealing.

ABSTRACT

This paper studies the application of genetic algorithms on one of the classical NP-hard ordering problems, namely, the job shop problem. We propose a chromosome representation for the scheduling problem and several new annealing functions for replacement. Experiments on benchmark problems are performed to demonstrate their effectiveness.

一、前　言

提到最佳化的相關研究問題，就會讓人聯想起有名的旅行銷售員問題(traveling salesperson problem, TSP)**、以及排程問題**(scheduling problem)**。在排程問題中，**JSP(job shop problem)**是公認最為困難、複雜的** NP-hard ordering**問**

題之一，它至少與 TSP 一樣困難，因為 JSP 特例之一的 FSP(flow shop problem)其複雜度和 TSP 之複雜度相當。

簡單的說，一個 job shop 包含了 m 個不同功能的機器(machine)及 n 個工作(job)，每個工作又可分為數個步驟(operation)，每個步驟只能在一個特定機器上完成，每個工作的步驟都有其特定的執行順序，JSP 就是要排定每個機器的排程表，以使完成所有工作的時間最短。同時，排程表要滿足以下的限制：

1. 每個步驟在進行過程中不能被中斷。
2. 每個工作的執行順序不能被違反，對同一個工作來說，每一個步驟必須等到前面的步驟完成後才能開始。
3. 每一個工作只能去同一個機器一次。
4. 同一個工作在同一個時間內，只能有一個步驟在機器上執行。
5. 同一台機器在同一個時間內只能執行一個工作步驟。

目前沒有任何多項式等級的演算法可以保證找到 NP-hard ordering 問題的最佳解，這類問題通常是學界亟欲克服的的挑戰。由於其重要性以及實用上的需求，JSP 問題的研究數量非常的可觀。處理 JSP 的問題除了使用傳統方法(例如，分枝界限法、優先規則法、移動瓶頸啟發法等)外，近年來開始有一些研究探討遺傳演算法(genetic algorithms, GA)在 JSP 的應用[1,2,3]，從這些研究中我們發現很少有研究能完整的證明其解答空間包括最佳解，而解答結果很好的研究也非常罕見[4]。因此，本研究認為 GA 在 JSP 排程問題上還有進一步發展的空間。

目前利用 GA 來解決 JSP 的研究，通常都需再加上局部搜尋的方法，才能有較好的解答。在 GA 的機制下，本研究提出一個適用於排程問題的「比較型編碼」，而且證明其解答空間的確包含最佳解。此外，研究中還提出一個借自於大自然運行觀念的「改良式退火函數」，用來調整替換條件，最後並以實驗驗證這些方法的效果。

二、遺傳演算法與模擬退火法

1. 遺傳演算法

遺傳演算法(genetic algorithms, GA)是 Holland [5]受達爾文的天擇說啟發而發展的演算法則。GA 採用了自然界中生物與生物之間競爭求生存的觀念，以一組特別的字串模擬各種生物的染色體(chromosome)，並根據染色體對環境的適應情況計算出各別染色體的適應度(fitness)，在每個世代之間讓各個染色體以隨機的方式進行交配(crossover)與突變(mutation)來產生下一代，而大環境會再根據該染色體的適應度選擇(selection)是否讓其生存。這個演化交替的動作會一直持續到達成最終目標為止。在 Holland 的文獻中所描述的最基本演算法則，稱為 simple genetic algorithm (SGA)。SGA 大致的演算法如下：

表一、模擬退火與最佳化問題類比對照表

物理系統	最佳化問題
狀態 (State)	可行解(Feasible solution)
內部能量 (Energy)	成本 (Cost)
基態 (Ground State)	最佳解 (Optimal solution)
快速淬火 (Rapid quenching)	局部搜尋 (Local search)
慢速退火 (Careful annealing)	模擬退火

```
SGA ( )
{
    隨機設定初始群組
    計算染色體的適應度
    當尚未達到最終目標
    {
        選擇好的個體
        進行染色體間的交配以及突變
        計算染色體的適應度
    }
}
```

GA 特別適用於搜尋解答空間很大、非線性、複雜、可能有雜訊、且無法預測可能解的問題空間，這是傳統決定性最佳化技巧(deterministic optimization)或是貪婪法則(greedy heuristics)無法做到的。GA 靠著群組間的各點可以同時探索不同的區域，再伴隨著世代演化交替、隨機搜尋的特性，這種平行處理的能力使它不容易陷入局部最佳解(local optimum)的困境，而向整體最佳解(global optimum)收斂；這些特點都是 GA 目前受到各領域重視的主要因素。

2. 模擬退火法

模擬退火法(simulated annealing)的原始概念來自於熱力學中熱平衡過程。退火是結晶物質從高溫液態冷凝至低溫固態的一種物理過程。根據熱力學原理，要使物質達到最低能量，可先將物質加溫融化，讓物質中的粒子可以自由活動；然後讓物質慢慢降溫，如果仔細控制降溫程序（也就是以緩慢退火替代急速淬火）則物質最後將可達最低（或非常接近最低）能量的結晶狀態。

1953 年 Metropolis 等人[6]為了找出特定溫度下物質內各原子的平衡配置，首先提出模擬熱力學的退火演算法。作法是將物質目前狀態 i 的能量設為 Ei，然後施以該物質一個微小的擾動使產生狀態 j，能量設為 Ej。如果 Ej – Ei 小於或等於零，狀態 j 被設定為新的目前狀態；否則狀態 j 被設定為新目前狀態的機率為 $\exp[-(Ej-Ei)/k_BT]$，其中 T 為該物質的溫度，k_B 為波茲曼常數(Boltzmann constant)。

模擬退火法可視為 Metropolis 退火演算法的一個類比。Kirkpatrick 等人[7]曾從組合問題最佳化的技術基礎上提出組合最佳化與 Metropoli 退火演算法的類比（表一）。

模擬退火法優於其他方法之處在於，可以避免解答落入局部最小的陷阱中。此法採取隨機搜尋策略尋找目標函

數的全域最佳解。這種搜尋策略可使其求解狀態在樣本空間中往目標函數值較低處移動，但透過隨機過程，仍能使其求解狀態有 P [= exp($-df/T$)，其中df 為目標函數 f 的增值，T 為退火溫度] 的機率往目標數值較高處移動。這樣的特性使得它具有跳越局部最佳解，在 Nonconvex 空間中求得全域最佳解的能力。

三、文獻探討

由於 JSP 的困難性，截至目前為止累積的相關研究數量非常可觀，所使用的方法也相當多元。本節分別就傳統方法及 GA 在 JSP 之應用加以探討。

1. 傳統方法

(一) 分枝界限法

分枝界限法的原則是列舉所有的可能解使成為一個搜尋樹；每個搜尋樹的分支均定義了所有可能解的一個子集合，任何的子集合都不會有交集，因此所有子集合就等於所有的可能解。每個子集合中，最佳解的目標函數是用下限值(lower bound)來估計，如果有分支的下限值大於最小的上限值，此分支就可以丟棄，如此找尋，一直到有滿意的答案為止。

（二）優先規則法

優先規則法是利用一些經驗法則（如：找執行時間最短的步驟、找執行時間最長的步驟等），調整目前尚未排定的步驟到排程表上，直到所有的步驟都放到排程表為止。

（三）移動瓶頸啟發法

移動瓶頸啟發法[8]是解決 JSP 最有力的經驗法則，包含兩個部分，第一部分(SB1)是重複用來解決第二部分在建立搜尋樹時所發生的問題，如同其名，它總會優先解決某一個機器上的瓶頸問題。

由於移動瓶頸啟發法是建立在最佳解可能是各機器都是最佳排程的假設上，如果有問題不滿足這樣子的假設，利用移動瓶頸啟發法的方法可能就無法找到最佳解，而陷入局部最佳解。

（四）模擬退火法

模擬退火法的基本觀念是不斷修改一個可行解，先在機器上隨機選擇不同的步驟（或工作），反轉它們的順序而變成另一個解答，如果新的解答是不可行的，就再調整相關的步驟，一直到調整成可行解，如果這個調整可以改善原來的目標函數，這個新解就會被接受，然後重複以上的步驟；如果不能改善目標函數，有時候會被接受（特別是在初期的時候），但是大部分的時候會被拒絕。

（五）禁忌搜尋法

禁忌搜尋法在搜尋的過程中，會記錄最近修改、移動的方式，以防止下次修改重複這些動作，因此比較不會在同一個地方打轉，而有可能可以跳出局部最佳解[8]。

2. 遺傳演算法在 JSP 的應用

GA 的問題獨立性很高，唯一用來表示問題特性的就是染色體編碼，在 JSP 問題上，已提出的染色體編碼方式有下列八種：

（一）Operation-Based 表示法

這種表示法是將排程表當成是一串有順序的步驟，而每個基因用來表示一個步驟，最自然的方式是用不同的自然數來表示不同步驟，類似在 TSP 問題中用的排列法 (permutation)，但是因為 JSP 本身有一些步驟上的限制，所以不是所有的排列順序都是可行解，Gen、Tsujimura 和 Kubota 提出一個改良方法，將同一個工作上的所有步驟都用同一個代號[9]。

（二）Job-Based 表示法

在這個表示法裡面，染色體是一串工作的排列順序，這個排列順序就是指工作排列的優先順序，在解碼的過程中，先排列優先順序較高的工作，排完這個工作的所有步驟後，再接下去排下一優先的工作，以此類推。因此在這個表示法中如果有 m 個工作，則染色體裡就只有 m 個基因[10]。

這種表示法和移動瓶頸啟發法是建立在同一個假設上，即當各機器上的排程都是最佳化的時候就是最佳解，如果有問題不滿足這樣子的假設，就會陷入局部最佳解，因此這種表示法並不能保證包含最佳解於其解答空間內。

（三）Preference List 表示法

這種表示法是由 Davis 先提出的[11]，之後才由 Falkenauer 及 Bouffouix 將其用到 JSP 的問題上[12]。

在這個表示法裡，如果是 n 個工作和 m 個機器的問題，就會有 m 個次染色體，每個次染色體代表一個機器，每個次染色體有 n 個基因，每個基因裡面的代號就代表工作，整組次染色體看起來也就是一個機器裡面要執行哪個工作的優先順序表。這個方法和 operation-based 表示法有相同的限制，也就是相同的代號只能出現一定的數目。

這個方法只能排出無延遲(non-delay)的排程表[13]，而最佳解可能不在其中。

（四）Job Pair Relation-Based 表示法

Job pair relation-based 表示法[14]用染色體來表示一對工作在機器上的關係，每個基因代表的關係如下式：

$$X_{ijm}=\begin{cases}1, & \text{if job } i \text{ precedes job } j \text{ on machine } m\\ 0, & \text{otherwise}\end{cases}$$

這個表示法可能是所有表示法裡面最為複雜的一種，重複性很高，也很容易產生不可行的解。

（五）Priority Rule-Based 表示法

Priority rule-based 表示法用染色體來表示一組規則的使用順序，排程時按照此順序來調整目前排程。

（六）Completion Time-Based 表示法

Completion time-based 表示法的染色體是一串步驟的完成時間表，每個基因代表不同的步驟的完成時間。

這個方法很容易發生同一個機器上，工作步驟執行時間有重疊的情況，所以也需要設計特別的遺傳運算子。

（七）Machine-Based 表示法

Machine-based 表示法[3]將 GA 與傳統的移動瓶頸啟發法相結合。在這個表示法中，染色體是一串機器的順序，每個基因代表一個機器。整個搜尋過程是以移動瓶頸啟發法為基礎，而 GA 只是決定解決機器瓶頸的順序。此法的結果和傳統移動瓶頸啟發法的結果比較起來，在問題較小時的確比較好，然而在問題較大時 (如 15×15 問題)，就略遜一籌了。

（八）Random Key 表示法

在 random key 表示法中，每個基因代表一步驟，其值為一隨機數，整數部分代表執行機器，小數部分代表優先順序[15]。

用這種方式來轉換的排程很容易違反 JSP 的限制，因此 Bean 和 Norman 還提出一些特別的解碼方式來輔助。另外，在產生初始群體及交配、突變時也需要特別處理以滿足限制。

四、系統架構

由於 JSP 是要找出每個機器上工作步驟的執行順序，因此 JSP 本質上是排優先序(priority)的問題。利用 GA 解 JSP 時，使用優先序的編碼方式應該是最自然的。

以單一的機器來看，只需要考慮哪一個工作應該先排，而以工作來看，還需考慮工作中步驟執行順序的限制。因此，如果先遵循工作的步驟順序，再來考慮機器上的工作順序，這樣排出來的排程表比較容易是可行解。

但如果採用 TSP 問題中用的排列法 (permutation)，對 n 個工作和 m 個機器的 JSP 染色體排列組合有$(n\times m)!$個，而機器上工作的排序最多只有$m\times(n!)$個，搜尋範圍顯然過大，因此本研究提出可以縮小解答空間的「比較型編碼」。

另外，為了避免 GA 落入局部最佳解，本研究擷取「GA 結合模擬退火求解」的觀念，提出以「改良型退火參數」的設計方式調整替換條件。

1. 比較型編碼

在 n 個工作和 m 個機器的問題裡，比較型編碼方式的染色體有 n×m 個基因，其中 n 個次染色體分別代表每個工作，每個次染色體裡有 m 個基因，代表該工作步驟之優先權。每個次染色體的內容是一串介於 1 到 n 的隨機整數。比較型編碼方式的染色體排列組合有$(n\times m)^n$個，在大部分的情形下 $m>1$，因此$(n\times m)^n < (n\times m)!$。而且比較型編碼方式可以在普通的 GA 操作，不用另做修正。其解碼步驟如下：

// M[m][j]: 代表機器 k 上，第 j 個排程步驟。
// S_k: 在 k 次修改時，可以被排列的步驟

比較型解碼演算法( )

```
{
  for all m              j_m :=1;
  for all m,j            M[k][j] := {ϕ};
  S_k:={ϕ}+所有沒有前期步驟的步驟;
  for(k=1;k=n*m;k++)
  {
        {TP} :=S_k中數目字最大的步驟群;
        if ({TP}個數>1)    then
        {
        {TP} := {TP}中所屬工作做最慢的步驟群;
              if ({TP}個數>1)    then
              {TP} :={TP}中工作編號最前的步驟;
              P_k := {TP};
        }
        else           P_k := {TP};
        m* := P_k步驟中需要用到機器;
        i := P_k步驟屬於的工作;
        M[m*][j_m] := M[m*][j_m] + {P_k}中;
        j_m++;
        PS_k := {PS_k} + {P_k};
        S_k := {S_k} - {P_k} +前期步驟已做完的步驟;
}
```

本研究所提之比較型編碼方式的染色體可能排列組合範圍，在大部分時候小於優先順序型的染色體排列組合，但是比較型編碼方式一樣可以排出優先順序型所排出的排程。解碼時，每次找出前步驟已排定之步驟中基因值最大者，排入所需機器最早可排入處即可。不難看出，此解碼方式所得之排程為 active 排程，即無法在不改變步驟順序下改善排程。

要證明此比較型編碼方式包含最佳解，只需證明對每個最佳排程，都有一個染色體經解碼後所得排程即為原最佳排程。給一個最佳排程，只要將每個機器上第 i 個步驟之基因設為(n+1-i)，則此染色體經解碼後所得排程之相對順序與原最佳解相同。由於所得排程與最佳解均為 active 排程，因此所得排程即為原最佳排程。

2. 改良型退火參數的設計方式

解決 GA 收斂太快的方法很多，其中由於模擬退火具備避免落入區域最佳解的特性，過去有一些研究顯示，結合 GA 與模擬退火法求解作業排程方面的問題，確實有明顯的成效[16,17,18]。本研究除了採取結合 GA 與模擬退火法的觀念外，更進一步提出結合改良型退火參數的設計方式求解 JSP 問題。

觀看地球的歷史後會發現，生物界會有大變動的情形，通常是因為外在環境突然改變了，使得本來適應力很好的生物，適應力突然變差，而適應力本來不好的生物，可能會變好。地球上突然換了一批生物重新演化，這樣一來，很有可能可以找到比變動前更好的生物。

本研究將上述觀念融入 GA 的演化中，希望藉此找到更好的答案。要一次替換大量的物種有兩種做法：第一種是改變外在環境（也就是重新設計適應函數），讓原本適

表二、Lawrence 10×10 benchmark 的最佳解與相關資料

大小	編號	歷來最佳解	最佳解出處	SB-GA
10 x 10	La16	945*	[21]	961
	La17	784*	[22]	784
	La18	848*	[22]	848
	La19	842*	[22]	848
	La20	902*	[23]	910
15 x 15	La36	1268*	[21]	1317
	La37	1397	[21]	1446
	La38	1196	[6]	1241
	La39	1233*	[23]	1277
	La40	1222*	[23]	1252

表三、各種替換法在 la16 問題上的測試結果

替換法	二十次平均值	前十名平均值	二十次變異數	前十名變異數	二十次最佳解
競賽替換法	988.6	978.7	194.14	17.21	968
群眾替換法	983.2	976.8	111.06	51.76	956
父代替換法	975.6	965.9	172.24	114.69	946

應度比較好的物種變差而被替換；第二種就是改變替換率，讓新產生的物種可以大量的替換原物種。對 JSP 問題而言，重新設計適應函數會較為困難，而改變替換率就比較簡單，所以本研究將採改變替換率的策略，將上述觀念放在替換動作的退火參數上。

退火方式的替換是從模擬退火的觀念中簡化而來的，在大部分的情況下，只有適應度比較好的子代染色體才能替換父代染色體，但是在剛開始演化時，適應度較差的子代染色體還是有機會可以替換父代，越到後面的演化，這個機率會越降越多。普通的退火方式，會用一個退火參數來控制，在開始演化的時候，退火參數設得比較高，每演化一代就將退火參數乘以一個小於一的常數。

很明顯的，如果要替換較多的父代，可以將退火參數調高，如此就可以做到重新演化的效果。如果要替換較多的父代，可以將退火參數調高，如此就能達到重新演化的效果。在後面的實驗結果裡，我們會探討究竟哪一種參數的退火較有效。

五、實驗結果與分析

JSP 問題的標準測試問題(benchmark)很多，本實驗所選的標準測試問題是 Lawrence 所提出的[19]。選擇這套問題主要是為了和 Dorndorf 及 Pesch 的 SB-GA(Shifting Bottleneck based Genetic Algorithm) [18]結果作比較。

本研究所選的 2 組問題，分別為 10×10 的問題（編號為 la16 至 la20），以及 15×15 的問題（編號為 la36 至 la40）。10×10 的問題往往是研究 JSP 的人拿來比較結果的標準問題，而 15×15 的問題則是 Dorndorf 及 Pesch 的研究中解答集合最大的一組問題。

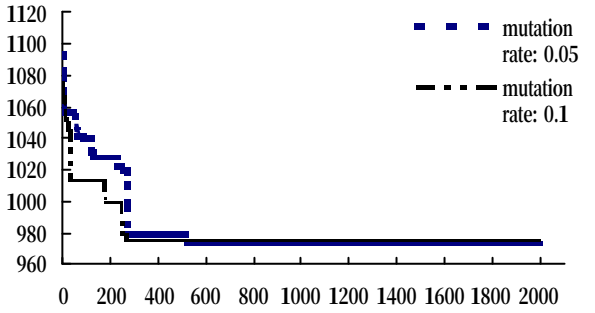

圖 1、la16 在傳統替代法、選擇法下的演化

表二是這 2 組標準問題目前為止所找到最好的解（有*號者為最佳解）、最佳解出處，及得自 Dorndorf 和 Pesch 所提 SB-GA 的最佳解。

本研究使用 Hunter [20] 發展的 SUGAL (Sunderland Genetic Algorithm)作為發展工具。對 10 x 10 的問題我們在 PC Pentium 133, Win 95 的軟硬體平台進行實驗。每個問題平均約需花費數分鐘的時間才能求得解答。由於 15 x 15 的問題所需花費的時間更多，因而改採 SUN SPARC 20, Unix (Solaris 2.4) 進行實驗。

（一） 10 x10 問題

在 10×10 問題的實驗中，我們根據測試問題的複雜度，選擇了 GA 的基本參數值，整數大小為 3，染色體長度為 100，執行世代為 2000 代，其他如編碼方式及交配方式等則嘗試不同設定，以觀察其對搜尋結果的影響。

（二） 選擇、替換方式測試

這個測試的目的是想找出一些較合適推廣到其他問題的參數。在此我們以 la16 為測試問題。目前 la16 問題的最佳解為 945，而 SB-GA 的最佳解是 961 [3]。

本測試的參數設定是：交配方式為兩點切割；替換方式為均勻法(uniform) ；選擇方式為輪盤法。突變率則測試 0.05 及 0.1 兩種。每組參數都重複做了二十次測試，所得的結果如圖 1。

這兩組參數同樣的沒有找到最佳解（還有 3% 的差距），且收斂太快。同時當突變率較大時，演化速度也較快，但因變動率太大，找到的解也比較不好。

為了要減緩收斂速度，我們改變了選擇方式與替換方式。選擇方法用競賽法，群組數目設為 25，突變率用 0.05。表三統計了每組參數做二十次重複測試的結果。歸納上面的分析結果可知，當選擇法使用競賽法，替換法使用父代替換法，替換條件使用退火參數時，解答結果較好。在後面的測試中的選擇法和替換法將以此為基準。

（三） 突變率、交配切割點測試

（四）

表四、切割點與突變率交叉測試的結果

切割點	突變率	平均誤差	前 50%平均誤差	變異數	前 50%變異數	最佳解誤差
1	0.1	2.89%	1.64%	154.41	16.38	1.14%
1	0.15	3.05%	1.55%	233.55	34.94	0.79%
2	0.1	2.70%	1.50%	156.38	23.83	0.72%
2	0.15	2.36%	1.45%	107.69	15.03	0.61%

表五、la36 至 la40 在不同群組數的測試結果

群組數	平均誤差	前 50%平均誤差	變異數	前 50%變異數	最佳解誤差
25	8.57%	7.23%	437.38	120.87	5.80%
10	7.80%	6.54%	395.54	123.63	5.05%
5	7.58%	6.59%	250.91	79.23	5.17%

表六、la37 在群組數為 10，改良退火一所得結果

方法	平均值	前 50%平均值	變異數	前 50%變異數	最佳解
改良退火一	1490.5	1477.3	286.25	165.61	1448
原始退火方法	1483.4	1470.9	284.74	78.89	1458

本測試的目的是找出適合的突變率和交配切割點數，此處我們以 la17 至 la20 做測試。由於本研究的替換條件是使用退火參數，因此子代替換父代的機率和其他研究相比就小很多，如此一來突變率也相對降低，因此本研究傾向將突變率調高，本組測試選擇了突變率 0.1 和 0.15。測試的結果如表四所示。

在 GA 中只看是否找到最佳解是比較不客觀的，因為隨機產生的初始群體，會影響搜尋的方向，如果使用較好的參數值，可以讓搜尋的方向更多元，如此一來，就能找到較為集中、優良的解答。從以上四組數據中的變異數來看，當交配是兩點交配、突變率為 0.15 時，所得的結果較好（與歷來的最佳解誤差最小），解答也較為集中（變異數最小），這樣的結果比較符合 GA 的要求。

2. 15x15 問題

在 15×15 問題（la36 至 la40）的測試上，加入了改良式退火參數測試，也根據前面的測試結果設了幾個基本參數值：整數大小為 4，染色體長度為 225，執行世代為 5000 代，交配為兩點交配，突變率為 0.15，選擇方式為競賽選擇法，替換方式為父代替換法，替換條件使用退火參數。其他則嘗試不同設定，以觀察其對搜尋結果的影響。

（五） 群組數目測試

從表五中可以發現當群組數目越大的時候，所求出解答的分布就越分散。然而從三個群組數目的最佳解解答效果來看，除了群組數為 25 時，解答效果明顯較差外，群組數 10 和 5 的解答效果都差不多。

表七、群組數與改良退火方式的交叉測試結果

方法	平均值	前 50%平均值	變異數	前 50%變異數	最佳解
改良退火二	1484.96	1476.1	121.84	50.29	1464
改良退火三	1484.86	1475.2	181.82	45.36	1464
改良退火四	1485.55	1476.2	178.65	116.76	1450
改良退火五	1485.40	1475.9	152.84	20.89	1470

在競賽選擇法中，如果總群體數是 100，當群組數設為 25 時，相當於隨機挑選 25%的染色體來比較，這樣子的選擇法會讓擁有較佳解的染色體快速繁殖。這種方法在小問題上會有較大幫助，但用在較大的問題上，反而會是一種障礙，因為較大問題的搜尋空間較大，有可能在尚未找到最佳解所在區域時，就已經被其他區域的次佳解佔滿整個群體，而無法有更好的演化。

因此，在 10×10 問題中，群體數目為 25 的效果比較好，但是到了 15×15 的問題中，群體數目為 25 所得的效果反而變差了。所以在後面更進一步的測試中，群體數目將以 5 和 10 交叉測試，以便從中選擇一組較適合 15×15 問題所用的方法。

（六） 退火函數測試

和 Dorndorf 及 Pesch 的研究結果比起來，前面測試結果並不理想。我們發現演化後期群體的進化速度變得很緩慢。從整個族群的分布來看，在演化後期，退火參數的作用已經消減到最低，幾乎毫無作用，子代很難去替換父代，所以各種不同型態的染色體在群體中的分布已經固定不變，如此一來，要繼續演進幾乎不可能。

為了要讓演化後期也能有較好的演進效果，本研究提出改變退火參數的想法。剛開始，改良式退火參數的設計是當演化已經停止進化一段時間後(例如 30 代的族群標準差完全不變），就將退火參數拉昇到原先系統所訂的初始值（改良退火一，在改良式退火參數演算法中的拉昇比率 init=1，遞減比率 decay=1），為了先局部測試改良式退火參數的效果，所以選擇 la37 的問題且群組數為 10 來測試，所得結果與尚未改變時比較，結果如表六。

從表六的資料來看，改良退火一的結果和原先預期差很多。可能是因為將退火溫度拉升到原始溫度，子代替換父代的機率變得很高，所以很容易將原始族群全數替換來重新演化，雖然這樣的替換方式可以很容易跳脫局部最佳解的狀況，但是卻也少了在既有基礎上再進步改善的優點，因此本研究又提出改良退火二到改良退火五的設計。

改良退火二到改良退火五的設計是當演化已經停止進化一段時間後，就將退火參數拉昇，這幾個退火方式的拉昇方法為：(1) 改良退火二：init = 0.1, decay = 0.95，(2) 改良退火三：init = 0.1, decay = 1，(3) 改良退火四：init = 0.5, decay = 0.95，(4) 改良退火五：init = 0.5, decay = 1。測試結果如表七所示。從表七的結果看，每一個改良退火方式的差異都不顯著。在接下來更廣泛的測試中，本研究選擇

表八、la36 至 la40 在改良退火三、群組數為 5 的測試結果

方法	群組	平均誤差	前 50%平均誤差	變異數	前 50%變異數	最佳解誤差
改良退火三	5	7.42%	6.38%	299.66	123.04	4.58%
	10	7.96%	6.84%	350.26	139.50	5.20%
改良退火四	5	7.56%	6.46%	294.51	106.62	4.86%
	10	8.04%	6.94%	313.37	119.15	5.55%

表九、本研究結果與其它研究結果的比較

問題	本研究	誤差	P-GA	誤差	SB-GA	誤差	SB-ABZ	誤差	最佳解
la16	946	0.11%	1008	6.67%	961	1.69%	978	3.49%	945
la17	784	0%	809	3.19%	784	0%	787	0.38%	784
la18	848	0%	916	8.02%	848	0%	859	1.30%	848
la19	851	1.07%	880	4.51%	848	0.71%	860	2.14%	842
la20	907	0.55%	928	2.88%	910	0.89%	914	1.33%	902
10*10 平均		0.35%		5.05%		0.66%		1.73%	
la36	1310	3.31%	1373	8.28%	1317	3.86%	1305	2.92%	1268
la37	1441	3.15%	1498	7.23%	1446	3.51%	1423	1.86%	1397
la38	1241	3.76%	1296	8.36%	1241	3.76%	1255	4.93%	1196
la39	1291	4.70%	1351	9.57%	1277	3.57%	1273	3.24%	1233
la40	1280	4.75%	1321	8.10%	1252	2.46%	1269	3.85%	1222
15*15 平均		3.94%		8.31%		3.43%		3.36%	
總平均		1.98%		6.53%		1.92%		2.47%	

了最佳解平均值較佳的改良退火三，及二十次測試中最佳解最好的改良退火四來做進一步測試。表八是改良退火三、四，與群組數目 10、5 交叉實驗所得的結果。

從表八的結果來看，退火方式三的結果比退火方式四的平均解答結果好，而族群數目 5 的整體解答效果都比族群數目 10 還要好。和原始退火方式做比較會發現在族群數目為 10 的時候，兩種改良式退火方式的解答效果都比原始退火方式還差；但是當族群數目為 5 的時候，改良式退火方式的解答效果卻比原始退火方式還好。

3. 綜合分析

表九是本研究在 10 個標準測試問題所得到的最佳解與分別採用 P-GA、SB-GA 以及 SB-ABZ 所得最佳解的比較。其中，P-GA 是利用 GA 選擇規則，以取代傳統經驗法則中的選擇方式； SB-GA 是利用 GA 選擇調整機器的順序，但搜尋解答的主體並不是 GA 而是移動瓶頸啟發法；而 SB-ABZ 則純屬傳統的移動瓶頸啟發法。選擇和 SB-ABZ 比較是因為在 Lawrence 所提的 40 組標準測試問題中，有 22 組的最佳解記錄仍是由 SB-ABZ 所保持或率先找到。

整體看來，本研究的解答效果和 SB-GA 是相當的，比 P-GA 的解答效果好很多，也比 SB-ABZ 還要好。但是在 15×15 的問題上，就比 SB-GA 以及 SB-ABZ 略差。

本研究所提的方法將排程編入染色體中，不但可以證明其包含最佳解，也符合 GA 的精神，同時，解答效果也比 P-GA 好，而且解答效果和借助區域尋的 SB-GA 相當，因此本研究所提出的方法是 GA 在 JSP 問題應用上較合適的選擇。

六、結論

本研究以 NP-hard ordering問題中的經典問題 JSP 來印證，用 GA 解決 JSP 問題，可達到和區域搜尋法效果相當的解答。實驗結果顯示解答效果可接受，另外和其他 GA 應用在 JSP 的研究相比，本研究所提出的染色體表示法保證包含最佳解。此外，實驗結果顯示本研究提出的改良式退火函數的確可以有效改善解答效果，不但最佳解誤差值可以降低，各個解答的變異數也可以降低。

GA 的收斂速度太快，是本研究中最棘手的問題，我們所提出的改良退火參數設計是初步的嚐試，尚未以嚴格的實驗設計找出合適的拉昇比率、遞減比率，在未來的研究中，應該可以針對這個設計，將這一部份的實驗數據加以補強，藉此找到最有效、合適的改良退火參數。

本研究的編碼方式和解答空間是多對一的對應，增加了 GA 搜尋的困難度，如何利用「折疊法」[22]將一樣排程的染色體折疊成同一個，以提升解答效果，亦是未來值得探討的課題。另外，本研究所提出的編碼方式是為排程問題所設計的，不只可以用在 JSP 以及 flow shop 問題上，也可以擴展應用到 open shop、time-tabling 等問題上，因此，如何將本研究所提出的方法應用在這些問題上，也是未來研究的一個方向。

七、參考文獻

1. Biegel , J. and J. Davern, "Genetic Alogrithms and Job Shop Scheduling", *Computers & Industrial Engineering*, Vol. 19, pp. 81-91 (1990).
2. Cheng, R., M. Hen and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms: I. Representation," *Computers & Industrial Engineering*, Vol. 30, pp. 983-997 (1996).
3. Dorndorf, U. and E. Pesch, "Evolution based learning in a job shop scheduling environment," *Computers & Operations Research*, Vol. 22, pp. 25-49 (1995).
4. Blazewicz, J., W. Domschke and E. Pesch, "The job shop scheduling problems: Conventional and new solution techniques," *European Journal of Operational Research*, Vol. 93, pp. 1-33 (1996).
5. Holland, J., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Michigan (1975).
6. Metropolis, N., A. Rosenbluth, A. Teller, and E. Teller, "Equation of state caculation by fast computing machines," *The Journal of Chemical Physics*, Vol. 21, pp. 1087-1091 (1953).
7. Kirkpatrick, S., C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, Vol. 20, pp. 671-680 (1983).
8. Adams, J., E. Balas and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Management Science*, Vol. 34, pp. 391-401 (1988).
9. Gen, M., Y. Tsujimura and E. Kubota, "Solving job-shop scheduling problem using genetic algorithms," In *Proceedings of the 16th International Conference on Computer and Industrial Engineering*, pp. 576-579 (1994).
10. Holsapple, C., V. Jacob, R. Pakath and J.Zaveri, "A genetic-based hybird scheduler for generating static schedules in flexible manufacturing contexts," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 23, pp. 953-971 (1993).
11. Davis, L., "Jon shop scheduling with genetic algorithm," In *Proceedings of the First International Conference on Genetic Algorithms*, pp. 136-140 (1985).
12. Falkenauer, E. and S. Bouffoix, "A genetic algorrithm for job shop," In *Proceedings of 1991 IEEE International Conference on Robotics and Automation*, pp. 824-829 (1991).
13. Corce, F., R. Tadei and G. Volta, "A genetic algorithm for the job shop problem," *Complex Systems*, Vol. 22, pp. 15-24 (1995).
14. Nakano R. and T. Yamada, "Conventional genetic algorithms for job-shop problems," In *Proceedings of the Fourth International Conference on genetic Algorithms*, pp. 477-479 (1991).
15. Bean, J., "Genetic algorithms and random keys for sequencing and optimization," *ORSA Journal on Computing*, Vol. 6, pp. 154-160 (1994).
16. Kolonko, M., "Some new results on simulated annealing applied to the job shop scheduling problem," *European Journal of Operational Research*, Vol. 113, pp. 123-136 (1999).
17. Murata, T., H. Ishibhchi and H. Tanaka, "Genetic algorithms for flowshop scheduling problems," *Computer ind. Engng*, Vol. 30, pp. 1061-1071 (1996).
18. Yip, P. and Y. H. Pao, "A guided evolutionary simulated annealing approach to the quadratic assignment problem," *IEEE transaction on System, Man, and Cybernetics*, Vol. 24, pp. 1383-1387 (1994).
19. Lawrence, S., *Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques (Supplement)*, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania (1984).
20. Hunter, A., "Sugal programming manual," <http://www.trajan-software.demon.co.uk/> (*1*995).
21. Carlier, J. and E. Pinson, "A practical use of Jackson's preemptive schedule for solving the job-shop problem," *Annals of Operations Research*, Vol. 26, pp. 269-287 (1990).
22. Kobayashi, S., I. Ono and M. Yamamura, "An efficient genetic algorithm for job shop scheduling problems," In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 506-511 (1995).
23. Applegate, D. and W. Cook, "A computational study of the job-shop scheduling problem," *ORSA Journal on Computing*, Vol. 3, pp. 149-156 (1991).