Commerce & Management Quarterly, Vol. 5, No3, pp. 275 ~ 288 (2004)

以基因演算法求解最小化設置時間單機排程問題

SOLVING SDST PROBLEMS BY GENETIC ALGORITHM

蘇純繒　翁瑞聰
雲林科技大學工業工程與管理研究所

Chwen-Tzeng Su　Jui-Tsung Wong
Department of Industrial Engineering and Management
National Yunlin University of Science and Technology

摘要

本研究主要是以啟發式演算法，求解單機的流程式相依設置時間問題(Sequence-Dependent Setup Times，SDST)。此演算法以基因演算法為基礎，加入洞悉法則的長期記憶來解 SDST 問題，主要是使基因演算法在搜解過程中避免下一代族群落入不必要的解空間裡。在經過測試後，本研究之演算法的確可使收斂過程不易落入區域最佳解，使得求解品質優於其他的啟發式演算法，但由於長期記憶的關係，使得在求解時間上較慢。

ABSTRACT

In this study, Sequence-Dependent Setup Times (SDST) problems can be solved by heuristic algorithm based on Genetic Algorithm. Adding long memory of Fathom Rule to solve SDST problems is to avoid next generation population falling into inessential solution space. After testing, the algorithm of this study is certain to make converged progress not to fall down local optima solution, and the quality of solution is better than other heuristic algorithm. Because of long memory, CPU time becomes slower.

壹、緒論

根據 Panwalkar（1973）報告調查指出，企業在考慮排程問題時，有 70%會

考慮到它的設置時間，由此可知設置時間在排程中其實是不可忽略的，流程式生產的管理工作是決定各工作站之負荷（workload）與生產之排程（schedule），而主要的目的有下列幾點：(1)滿足交期。(2)使生產前置時間最小化。(3)使設置時間或成本最小化。(4)使在製品（WIP）庫存最小化。而 SDST 問題主要的目的是使總設置時間最小化。SDST 問題會如此受到重視，是因大多數的實際問題，皆可轉成類似的模型。在排程方面，例如：射出成形加工、電腦工作平台、IC 測試機台、類似加工工件分群等；除應用於排程上，其他方面還有：揀貨上的應用、派工、無人搬運車（AGV）行走路徑規劃、貨運車之行走路線…等。由此可知 SDST 問題可運用的範圍很廣。SDST 問題是屬於 NP-hard 的問題 Du and Leung（1990），以窮舉法的方式來找最佳解，當有 n 個工作時就會有 n！種排序，目前的解法大致分為兩種，一是以較有效率的啟發式解法來求，另一種就是利用動態規劃或分支界限法，啟發式解法所得到的解有可能落於區域最佳解，而動態規劃在求解排列組合方面其效率並不是很好，分支界限法是目前能效率求到最佳解的方法，雖然可利用「叢集電腦」的技術（cluster of computers）來形成平行計算環境，來做平行運算，除了需增加多台機器或處理器外，且其求解效率仍以啟發式演算法較好。因此本研究決定以各取其優點（利用洞悉條件快速的縮小解空間，加上基因演算法平行搜解），發展本研究的基因演算法（Genetic Algorithm and Fathom Rule，GAF），來求解 SDST 問題。

貳、文獻探討

本研究主要是探討 GAF 演算法對 SDST 問題的求解品質，且 SDST 問題與傳統推銷員旅行問題（Traveling Salesman Problem，TSP）很類似。因此，在文獻探討方面主要分為兩個部份，第一部分為 SDST 問題相關文獻探討，另一部分為 TSP 問題。

一、SDST 問題相關文獻

SDST 問題最主要就是在探討設置時間最小化問題。Backer（1974）提到單機的 SDST 問題就類似 TSP 問題，且為 NP-hard 問題。關於這個問題的求解方式 Glassey（1968）以動態規劃（dynamic programming）的方法來得到 SDST 問題最佳解，但是動態規劃在求解排程、派工問題，其效果並不理想。因此，大部分的學者開始尋找有效率且能得到最佳解的方法，來求解 SDST 問題；在後期，開始有一些學者利用分支界限法（B&B）來進行求解，如：Brucker, Hilbig and Hurink（1999）、Ragatz（1989），但他們的洞澈（fathomed）條件，不是設計的過於煩雜就是求解效率不佳，這些解法主要是先以匈牙利法（Hungarian method）來求解指派問題，再以分支界限法作修正的動作，在求解的過程中，除了須畫解題樹（solution tree）外，還需畫許多的修正表格，因此這無疑是降低了求解效率。

二、TSP 問題相關文獻

旅行推銷員問題（Traveling Salesman Problem, TSP）為網路迴路問題中最典型的一種, 典型的TSP問題可定義為從某一城市出發，在所有城市皆拜訪一次又回到起始城市的限制下，找尋一最短路徑。求解TSP問題的方法可分為確切解法（Exact Algorithm）及啟發式解法（Heuristic Algorithm）兩種，確切解法如：分枝界限法（Branch-and-Bound）Volgenant and Jonker（1982）、Fishetti, Salazar, and Toth（1993）、動態規劃（Dynamic Programming）Jellouli and Chatelet（2000）、Hansen and Karp（1962）、整數規劃（Integer Programming）Gomory（1963）等，雖然這些方法都可以找到最佳解，但隨著問題的增加，其求解時間呈指數增加，無法在合理的時間內完成求解；因此，當問題規模很大時，大多以啟發式演算法來求解。TSP傳統啟發式解法大概可分為三種：(1)途徑建構, 如：節省法Clarke and Wright（1964）。(2)途徑改善，如：模擬退火法（Simulated Annealing）Kirkpatrick（1984）、羅中育（2000）、禁忌搜尋（Tabu Search）Knox（1994）、門檻接受法（Threshold accepting）韓復華與楊智凱（1996）。(3)綜合法吳泰熙與張欽智（1997）。上述方法為目前較常見的啟發式演算法的方式。

參、基因演算法之建立

在這節裡面，主要是要構建本研究的基因演算法，此演算法主要以基因演算法為主體，加入了洞悉法則的觀念，也就是在搜尋到區域的最佳解後，就不再於此區域做搜尋。因此，首先設計適用於此問題的洞悉法則，之後再將此觀念加入基因演算法。

一、問題定義

表一 SDST 例題顯示出工作站上的工作(1,2,……,n)，其設置時間是基於工作處理順序。例：若工作 2 後緊跟著後 1，則工作 2 的設置時間長達 *h12* 單位元，其最終目的是要探討工作如何排序會使總設置時間最短。

符號定義：

h_{ij}：表由工作 i 換到工作 j 的設置時間；

$i \neq j$　i=1,2,…,n　j=1,2,…,n

（$h\ i0$ 表一開始工作 i 的設置時間）

$$X_{ij}=\begin{cases}1 & \text{工作 } i \text{ 緊接於工作 } j \text{ 之後}\\ 0 & \text{其他}\end{cases}$$

將上述問題表示為數學式(3-1)至(3-4)，如下：

Minimize

$$\sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij} X_{ij} + h_{i0} \qquad (3\text{-}1)$$

Subject To

$$\sum_{i\in s}^{n}\sum_{j\notin s}^{n} X_{ij} \geq 1 \qquad (3\text{-}2)$$

$$\sum_{j=1}^{n} X_{ij} = 1 \quad , i=1,2,\ldots,n. \qquad (3\text{-}3)$$

$$\sum_{i=1}^{n} X_{ij} = 1 \quad , i=1,2,\ldots,n. \qquad (3\text{-}4)$$

S = {1,2,…,n}；為非空集合

二、下限值與洞悉法則設定

在這個部分主要是介紹本研究使用

的洞悉條件及下限值的計算。此設計主要是當基因演算法搜尋到符合洞悉條件的解時，此區域的解將不必再搜解，因此區域已不可能有更佳的解，本演算法即是藉此設計，使基因演算法的搜解不必浪費在無最佳解的空間上。

(一) 下限值的計算

因本研究主要是求算最小化的問題，所以界限值為 LB（lower bound），因此在這個地方主要是探討 LB 的求算方法。其實這種排序問題，可以把它看成類似指派問題（assignment problem），只是指派問題，須考慮指派物及被指派物兩者。而此排序問題，只須考慮被指派物，因其順序已被確定（即 1,2,3,…，只須考慮那一工作分派到位置 1, 那一工作到位置 2）。因此，以這種觀念來求算 LB 值，其公式(3-5)如下：

$$LB=hi0+\sum_{i\in Q}^{n}\sum_{j\in Q}^{n} h_{ij}+\sum_{j\notin Q}^{n}\operatorname{Min}_{i}\{h_{ij}\} \quad (3\text{-}5)$$

Q：為已排定工作的集合

公式說明：因為除局部解外，其餘未被排入的工作，皆選擇那一個前置工作到此工作其設置時間是最小的，以此為考慮對象作為界限值，因此下限值為局部解之值＋各未被排入之行最小值的總和。在不得產生循環迴路方面，則是因為已被排入的工作不會再出現，所以不用再被考慮。

(二) 洞悉條件(fathomed)

如果有結點符合洞澈條件，則表示此結點已被洞澈，即結點不用再分支出去。而一般的洞澈條件有三個（在前面已經敘述過），但解此種排程問題，因其類似指派問題以分支界限法求解，因此洞澈條件只有兩個，即不會有結點不可能包含可行解的情況發生，所以洞澈條件只有兩個，如下：

洞悉條件 1（F1）：全部工作件皆能被排到，亦即：已找到在此結點中具最小 LB 之可行解。

洞悉條件 2（F2）：當此結點之 LB 值，大於等於目前的總設置時間值（目前結點 all 之值）。

說明：因為下限值的演算法為「局部解值加每未被指派行之最小值」，所以有可能產生一列裡面皆沒有後續作業可選，此時即有作業會未被排完，因此如工作皆能被排到，此時即符合洞澈條件 2，則此結點可被洞澈。

三、基因演算法建立

本研究之基因演算法，有別於傳統的基因演算法主要是在搜解的地方，其運用下限值及洞悉條件，即在確定結點被洞悉後，就不再此區域進行搜尋，因洞悉後的解空間已找到此區域的最佳解，或是已無較佳解的可能，將這區域的解列入基因演算法繁殖後代的禁走名單之中，其示意圖如圖 1。下面為本研究所設計的啟發式演算法的求解步驟：

第一階段：起始階段

step1. 輸入各工作間變換時的設置時間、突變率（cr）、交配率（mr）、群體大小（K），並將迭代計數（t）設為 1。

step2. 以隨機的方式產生 K 組起始解

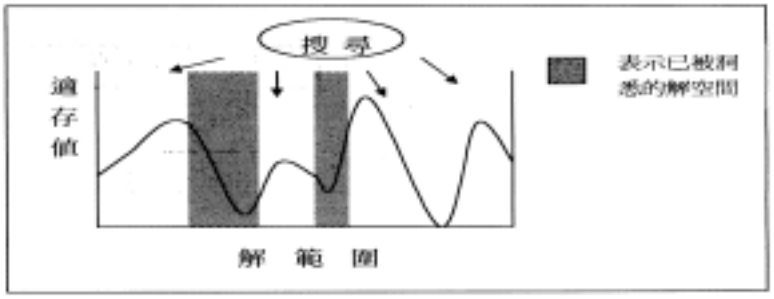

圖 1　本研究基因演算法搜解意示圖

表 1　SDST 例題

		以下工作設置時間(分鐘)					
	設置時間(分鐘)	A	B	C	D	E	F
	A 9	-	8	6	11	8	3
進	B 3	8	-	12	6	10	5
行	C 6	6	12	-	5	9	7
工	D 5	11	9	5	-	13	10
作	E 1	10	13	10	13	-	9
	F 7	12	10	9	8	7	-

（*POP(k)*），*k=1,2,...,K*。並計算各組解的適存值 $fit(k)$。設 $best\_fit(i)=Min\{fit(k)\}$、*best_POP=POP(i)*。

第二階段：檢查洞悉條件

step1. 檢查洞悉條件 1（F1）：先決定此染色體為解題中的層次, 其決定方式以隨機的方式產生, 其位置之後的工作排序按每行最小值, 如全部工作件皆能被排到, 亦即：已找到在此結點中具最小 LB 之可行解, 並將此解存於禁走名單中。

*例：*表 1 有六個工作（A,B,C,D,E,F），其各工作間的設置時間會依工作先後順序有所不同。當以 GAF 產生一組新解 B-D-A-C-F-E, 接著檢查洞悉條件 1, 先以隨機的方式決定此染色體層次, 在此例題如為 2, 即 B-D 以下的工作按照每行最小值重排, 如果剩下的工作皆能被排入, 即符合洞悉條件 1, 如此例題 B-D-C-A-F-E, 即符合洞悉條件 1, B-D 的解空間皆可被洞悉起來。

step2. 檢查洞悉條件 2：先決定此染色體

為解題中的層次, 其決定方式以隨機的方式產生, 檢查是否等合洞悉條件 2, 即此解區域之 LB>best_fit(i), 亦此區域不會有最佳解。因此, 將此解此存於禁走名單（Tabu list）中。

第三階段：搜解階段

step1. 以輪盤選取（Roulette wheel selection）選擇複製名單之染色體, 此法則是適存值較佳的染色體被選到的機率愈大, 其執行的步驟如下：

1. 加總各染色體於輪盤上的域區, $TG=\sum_{k=1}^{K} G(k)$, $G(k)=exp[-fit(k)]$。

2. 計算每一染色體被選取的機率值, $Pk=G(k)/TG$。

3. 計算其累加機率值, $qk=\sum_{j=1}^{K} p_j$。

4. 蒐集複製染色體 *Ca*, *Ca*=
   - 選擇第一個染色體 if $r<qk$
   - 選擇第 k 個染色體 if $qk-1 \leqq r<qk$

   *r*為0~1的隨機值。

step2. 基因交配：確定是否進行交配, 即 r<cr（r 為小於 1 的亂數）; 否, 跳至 *step3.*。是, 進行交配。交配方式為從複制名單中隨機選取兩染色體為交配母代, 採用雙點式交配, 以隨機的方式產生兩亂數 *r1*、*r2*, 作為交配位置。將交配出的基因檢查是否於禁走名單中, 如果是就重新交配, 否則重新確定是否進行交配, 並檢查兩染色體是否符合 F1 或 F2, 其步驟同第二階段。

step3. 基因突變：確定是否進行突變, 即 r<*mr*（r 為小於 1 的亂數）; 否, 跳至 *step4.*。是, 進行突變。突變方式為從複制名單中隨機選取一染色體來進行突變, 採用雙點式突變, 以隨機的方式產生兩亂數 *r1*、*r2*, 作為突位置, 將突變出的基因檢查是否於禁走名單中, 如果是就重新突變, 否則重新確定是否進行突變, 並檢查此染色體是否符合 F1 或 F2, 其步驟同第二階段。

step4. 以輪盤選取方式選出新一代的族群（方法同 *setp1*）。

step5. 更新最佳染色體, $best\_fit(i)=\mathrm{Min}\{fit(k)\}$、*best_POP=POP(i)*。

step6. 收斂與否：檢查是否已達到所定的世代族群 *t=Max_t*; 是, 結束。否, 回到 *step1*, *t=t+1*, 且整理禁走表單：每一百個世代時, 如有較後面的禁名單包含於較前面的禁走名單中, 則予以消除, 以提高搜尋速度。

肆、基因演算法測試

這部份主要是測試本研究之演算法：GFA 演算法, 在總設置時間最小化的單機排程問題中, 其求解品質是否良好。除了將求解結果與修改前的基因演算法 David（1989）做比較外, 並且與其他兩個啟發式解法作一比較：模擬退火法（Simulated Annealing Algorithm, SA）Cerny（1985）、塔布搜尋法（Tabu Search, TS）Armentano and Ronconi（1999）。本研究以套裝軟體 Matlab R12

表 2　收斂世代比較　　(世代)

	演算法	
n-job	基因演算法	改良基因演算法
25	76	237
80	445	612
150	1672	2413
300	3124	4533

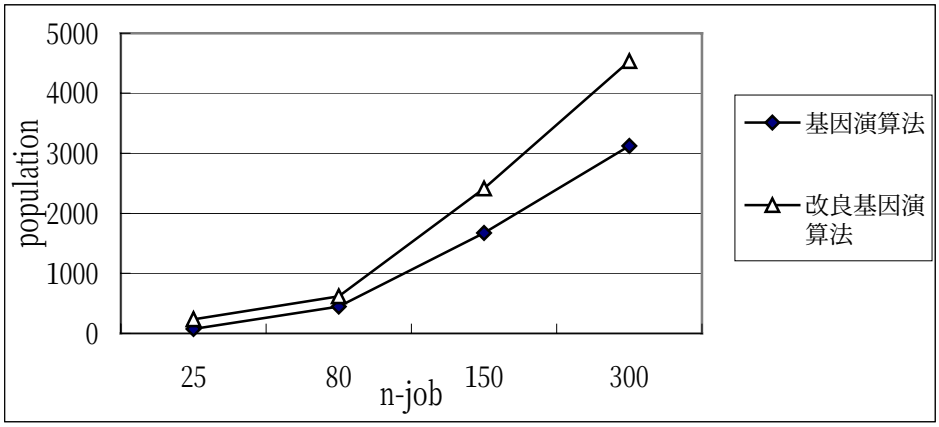

圖 2　收斂世代比較

來撰寫程式語言，因為需要用到許多的矩陣運用，所以在快速記憶體方面，選則較高的 640MB 的 RAM 來進行測試，其 CPU 為 AMD1.0G。測試題目以隨機的方式分別產生 25、80、150、300 個工作，以測試各演算法在不同問題大小下的求解品質。

一、基因演算法

在這個部分，主要是在比較傳統的基因演算法及本研究之基因演算法，在下表四為兩演算法在各問題大小下，分別收斂的世代平均值（每次實驗 20 次），超過 500 個世代其解無進步則判定為收斂。由表 2 及圖 2 中可發現，本研究基因演算法須要較多的世代才能收斂，這顯示它不易落入區域的最佳解。

圖 3、圖 4 分別為基因演算法與本研究基因演算法的收斂過程，在相較之下，本研究基因演算法的確是較晚收斂，且求解品質也較好（下一節有具體的探討）；圖 5 及圖 6 分別為兩者前 20 代的收斂過程，*為各世代族群的染色體，由圖 5 可發現，基因演算演算法在前 20 代中大部分的染色體皆已收斂，而圖 6 本研究基因演算法的染色體大部分

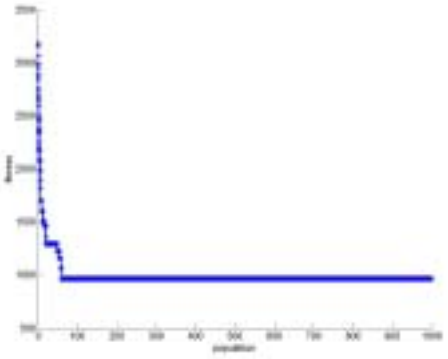

圖 3 基因演算法收斂圖

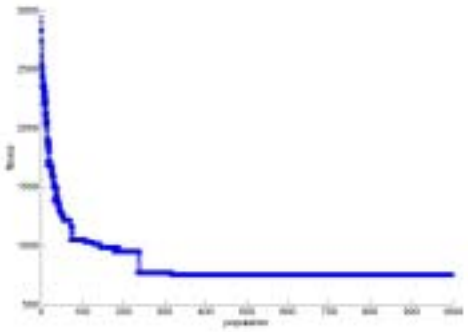

圖 4 改良基因演算法收斂圖

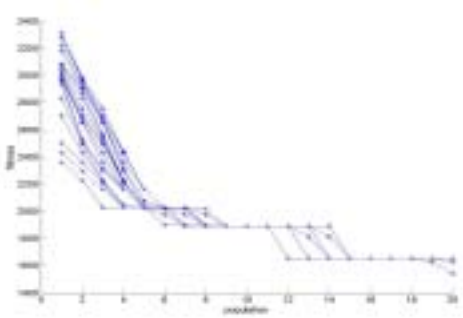

圖 5 基因演算法收斂圖（前 20 代）

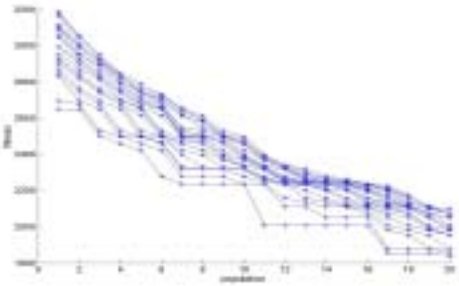

圖 6 改良基因演算法收斂圖（前 20 代）

皆未收斂。因此，在上述測試中瞭解到，本研究基因演算法較基因演算法較不易收斂，亦不易落入區域的最佳解。

二、基因演算法和各演算法比較

在這個部分中，主要是要比較本研究基因演算法與其他啟發式演算法，分別在求解時間及品質上做一探討。

(一) 求解效率探討

表 3 為各演算法分別在不同的問題大小下，執行 20 次的平均值，其中以模擬退火法（SA）的平均求解速度最快，而以本研究之演算法的求解速度較慢，且由圖 7 可發現，其求解時間隨著世代的增加而呈指數的速度增加。

表 4 為本研究基因演算法之禁走表單數量，由表中可發現在問題變大時，禁走表單也隨著大量的增加，即部分的解空間已被洞結，可藉由此方法找到更好的解，而不浪費時間在這些區域搜尋，但也因禁走數量龐大，造成整個演算過程較慢。

(二) 求解品質探討

在這個部分，主要是比較本研究之演算法與其他啟發式演算法的求解品

表 3　求解時間　(秒)

n-job	演算法			
	GA	GFA	SA	TS
25	6.2751	15.475	8.743	1.116
80	8.875	20.52	10.254	4.275
150	54.928	167.572	27.145	62.036
300	172.7325	754.456	41.403	484.225

表 4　改良基因演算法禁走數量

	洞悉層次			
	0%~3%	3%~25%	25%~50%	50%~100%
n-job				
25	103	1748	4525	$152*10^{2}$
80	$453*10^{2}$	$751*10^{2}$	$419*10^{4}$	$675*10^{6}$
150	$258*10^{3}$	$754*10^{5}$	$657*10^{7}$	$475*10^{11}$
300	$825*10^{6}$	$458*10^{10}$	$145*10^{14}$	$345*10^{17}$

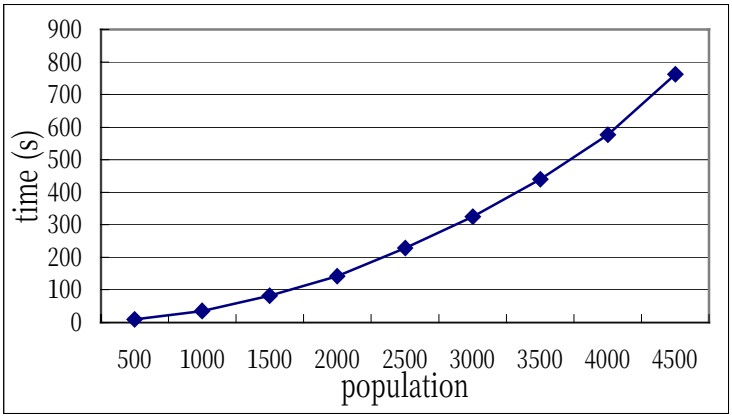

圖 7　改良基因演算法求解時間

表 5 求解品質檢定 α=0.05

n-job	H_0 H_1	GFA=GA GFA<GA	GFA=SA GFA<SA	GFA=TS GFA<TS
25	t-value	-4.831	0.072	-1.187
	p-value	0.000116 *	0.943	0.25
80	t-value	-15.123	-1.1886	-14.507
	p-value	0.0 *	0.074738	0.0 *
150	t-value	-13.702	-6.597	-38.941
	p-value	0.0 *	0.0 *	0.0 *
300	t-value	-13.702	-6.597	-38.941
	p-value	0.0 *	0.0 *	0.0*

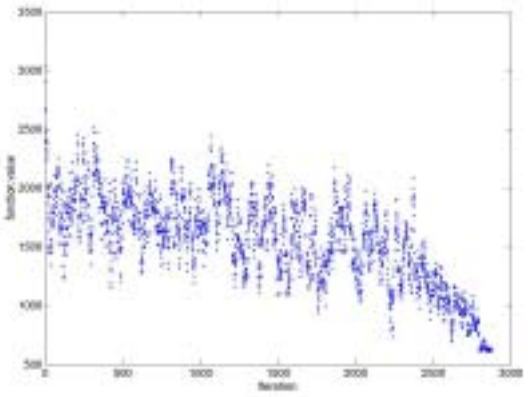

圖 8 模擬退火法收斂過程

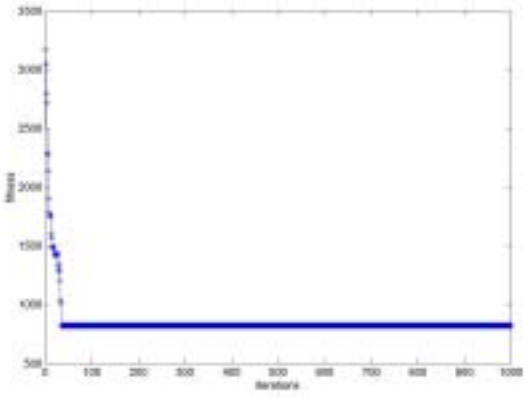

圖 9 塔布搜尋法收斂過程

質，其目標是總設置時間最小化，其中 GA、GFA、TS 的演算世代會因問題大小與收斂與否而有不同，超過 500 代解無進步則判定收斂，其演算時間如表 4。表 5 為本研究之演算法與各演算法分別進行 T 檢定（*號代表有顯著），由表中可發現本研究之基因演算法求解品質除了小問題外，其他皆優於各演算法。圖 8、圖 9 分別為模擬退火法與塔布搜尋法在問題大小為 25 時的收斂圖，從中也發現模擬退火法，在收斂過程是以上下跳動，其收斂時間也較塔布搜尋法慢，而塔布搜尋法則是快速的收斂於近似解。

圖 10 至圖 14 分別是各演算法，在問題大小為 25、80、150、300 時的敘述

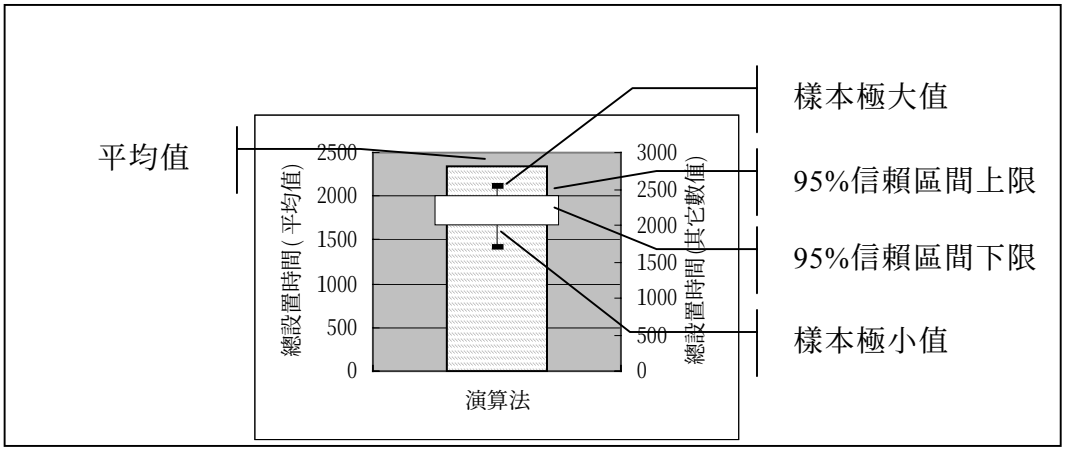

圖10　說明圖

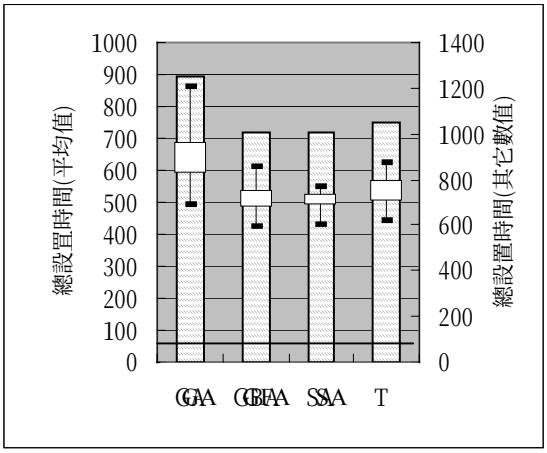

圖11　各演算法敘述統計（工件25）

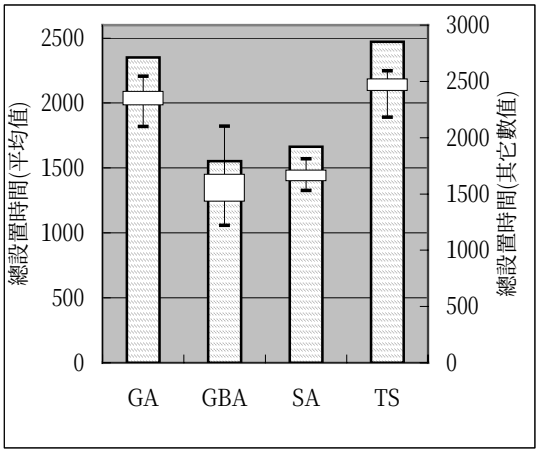

圖12　各演算法敘述統計（工件80）

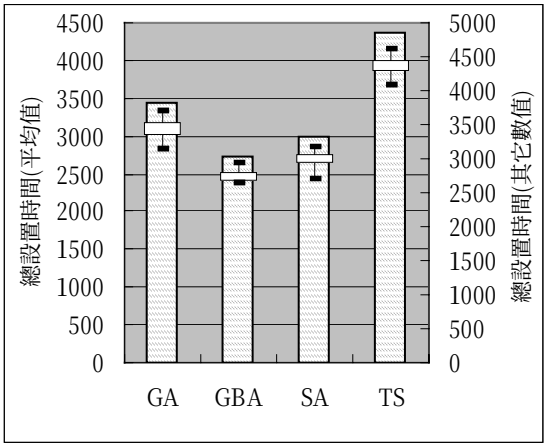

圖13　各演算法敘述統計（工件150）

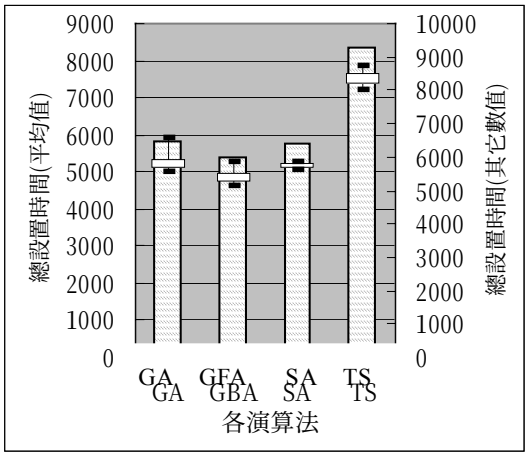

圖14　各演算法敘述統計（工件300）

統計及 95%之信賴區間，圖 10 為圖示說明，從這些圖中可發現，除了小問題外，本研究基因演算法之平均值皆比其他演算法要好，而在信賴區間方法則以模擬退火法表現的較好，亦信賴區間範圍較小具有一定的求解品質。在樣本最小值方面，以基因洞悉演算法較好。因此，在不考慮求解時間因素下，整個而言以本研究演算法的求解品質較好。

伍、結論及建議

在工商社會的時代中，無論企業規模的大小，如果能使作業時間縮短，無疑的能替公司創造許多的工作機會。本研究發展的基因演算法，就是要找到流程式產生設置時間最小化的單機排程問題的較佳近似解。本研究演算法主要以傳統的基因演算法為主體，從中導入了分枝界限法的觀念，即結點在被洞悉後，下一代的染色體就不在此區域做搜尋，以避免落入區域最佳解中，在經過許多的測試實驗後，也證實本研究之基因演算法的確較不易落入區域最佳解，使整體的求解品質有較好的表現，但也因加入了記憶的功能，使得在求得速度上較慢，且隨世代的增加呈指數變化。因此，如果不考慮時間因素，本研究之演算法，其求解品質較優於傳統的一些啟發式演算法，包括：基因演算法、模擬退火法、塔布搜尋法。

雖然本研究基因演算法其求解品質較佳，但由於在演算過程中，需經過許多記憶及搜尋的動作，使得其求解速度要比一般的啟發式解法慢上許多，因此未來研究建議如下：

1. 以其他啟發式演算法，如：模擬退火法等，取代基因演算法，看是否在求解品質及時間上有進步。
2. 在禁走名單中，只記憶前 3%層次之洞悉結點，亦即記憶較大的區域解，而不記憶小區域的解，如此則可大大的減少記憶空間及搜索的時間。
3. 基因洞悉演算法，在換解時採用有智慧搜尋解，以取代隨機式的換解。
4. 將基因洞悉演算法，應用於其他領域，但首先必須找到洞悉法則。
5. 以不同的方法將分枝界限法導入基因演算法，如：基因演算法無法在較佳的解中，作仔細的搜尋動作，可藉由分枝界限法作區域性的搜索。

以上就是提供未來研究方向的一些建議。總而言之，如何以快速的方法，去找到較佳的近似解，仍是在面對 NP-hard 的排程問題所努力的方向。

參考文獻

一、中文部分

1. 吳泰熙、張欽智(1997)，以禁忌搜尋法則求解推銷員行問題，大葉學報，(6)1。
2. 韓復華、楊智凱(1996)，門檻接受法在 TSP 問題上之應用，運輸計劃季刊，25(2)，163-188。
3. 羅中育(2000)，田口品質工程應用於模

擬退火法參數組合—以旅行推銷員問題（TSP）為例，雲林科技大學工業工程與管理研究所碩士論文。

二、英文部分

1. Armentano, V. A., & Ronconi, D. P. (1999). Tabu search for total tardiness minimization in flowshop scheduling problems. Computers & Operations Research, 26, 219-235.

2. Backer, K. R. (1974). Introduction to Sequencing and Scheduling. John Wiley, NY.

3. Clarke, G., & Wright, W. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points, Operation Research, 12, 568-581.

4. Cerny, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimization Theory and Applications, 45, 45-51.

5. David, E. (1989). Genetic Algorithms in Search. Optimization & Machine Learning.

6. Du, J., & Leung, J.Y.-T. (1990). Minimizing total tardiness on one machine is NP-hard. Math. Opns Res, 15, 483-495.

7. Fishetti, M., Salazar, J. J., & Toth, P. (1993). A Branch and Cut Algorithm for the symmetric Generalised Travelling Salesman. Problem, Working paper, University of Bologna.

8. Glassey, C. R. (1968). Minimum changeover scheduling of several products on one machine. Operations Research, 16, 342-352.

9. Gomory, R. E. (1963). Solving linear programming problems in integers. Proceedings of Symposia in Applied Mathematics, 10, 211-215.

10. Hansen, M., & Karp, R. (1962). A dynamic programming approach to sequencing problems. SLAM Review, 10, 196-210.

11. Jellouli, O., & Chatelet, E. (2000). Dynamic programming approach for the generalized traveling salesman problem, International Workshop, Minsk, Belarus.

12. Kirkpatrick, A. (1984). Optimization by simulated annealing: quantitative studies. Journal of statistical Physics, 34, 978-986.

13. Knox, J. (1994). Tabu search performance on the symmetric TSP. Computers & Operations Research, 21(8), 786-802.

14. Brucker, P., Hilbig, T., & Hurink, J. (1999). A branch and bound algorithm for a single-machine scheduling problem with positive and negative time-lags. Discrete Applied Mathematics, 94, 77-99.

15. Panwalkar, S. S. (1973). Sequencing research and the industrial scheduling problem. In Symposium on the Theory of Scheduling and Its Applications

(Edited by S. E. Elmaghraby), 28-38.

16. Ragatz, G. L. (1989). Scheduling to minimize tardiness on a single machine with sequence dependent setup times. Opns Res, 23, 118-136.

17. Volgenant, T., & Jonker, R. (1982). A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation. European Journal of Operational research, 9, 83-89.

2003 年 01 月 15 日收稿

2003 年 02 月 24 日初審

2003 年 07 月 17 日複審

2003 年 10 月 30 日接受