東海學報 42 卷(2001):79-90
中華民國九十年七月出版

使用遺傳演算法求解二機開放工場具工作連接性限制問題

賴崇瑋　姚銘忠　曾宗瑤

東海大學工業工程與經營資訊研究所

摘要

本研究中探討運用遺傳演算法求解二機開放工場中工作具有工作連接性現象的排程問題（即 $O_2|Blocking|C_{max}$ 問題）：假定工作在兩機台的處理時間均已給定，本研究的目標爲將 n 個具有工作連接性的工作排入此二機開放工場排程中，求其最短之製距(Makespan)。依據本研究所得的結果，我們建議運用遺傳演算法求解 $O_2|Blocking|C_{max}$ 問題時，可使用 LOX(Linear Order Crossover)爲交配運算子，採用 PBM(Position Based Mutation)作爲突變運算子。同時本研究也在遺傳演算法求解時參數之設定，如：族群數目、交配機率、突變機率及後代數等提供明確之建議。此外，本研究運用 600 個隨機範例的數據結果，觀察工作數目及工作處理時間的變異程度(標準差)等實驗因子，對遺傳演算法求解之品質及其執行時間的影響。本論文的研究成果，可提供決策者欲運用遺傳演算法求解 $O_2|Blocking|C_{max}$ 問題時，得有完整決策環境的參考，並事先得知可能影響決策品質的注意事項。

關鍵詞：排程、工作連接性、開放性工場、遺傳演算法

一、前言

在排程中所謂工作具連接性(Blocking)現象乃是指『從開始到完成的過程當中，工件必須被加工，不可以有任何等待或是閒置的情形發生；換句話說，工件在加工機器或是機器之間不能有任何中斷的情形發生的限制條件底下，使產出速率最大；即製距(makespan)最短』。

這樣的生產型態，除了因爲機器之間沒有存貨緩衝區(Buffer)外，大部分的原因是因爲生產技術本身的自然限制，例如在某些製程中，由於物料的特性(例如：溫度、黏性等等)，使其每一項操作完成之後必須緊跟著下一項。一般來說，這樣的生產型態常見於鋼鐵、塑膠、製藥、電鍍、食品、紡染或甚至積體電路製造業等。(林恆貞, 1998)

本研究考慮在二機的開放工場(Two-machine openshop)工作具有連接性現象的排程問題。一般在開放工場中，對於任一工作，在機器上的加工順序可以任意決定。本研究的目標爲：完成 n 個具有工作連接性的工作於二機開放工場的排程，求取其製距(Makespan)的極小化。根據 Graham et al. (1979) 所提出之排程問題標準化標記方法，我們可將這個問題標記爲 $O_2|Blocking|C_{max}$ 問題。

本文將以遺傳演算法（Genetic Algorithm; GA）爲研究方法，求解 $O_2|Blocking|C_{max}$ 問題之最佳解。

二、文獻探討

在本節我們先針對 $O_2|Blocking|C_{max}$ 問題相關的主題，例如：開放工廠、二機及工作具連接性的排程問題及遺傳演算法在排程問題之應用等相關的文獻進行探討。

2.1 相關的排程問題

二機排程問題的濫觴，可以 Johnson (1954)爲其代表作。Johnson 提出了一個簡單（其演算法複雜度爲多項式時間內可完成）的演算法，求解 $F_2||C_{max}$ 問題的最佳解。當我們將工作連接性之限制現象加入考慮時，則 $F_2|Blocking|C_{max}$ 問題具有旅行銷售員問題(Traveling Salesman Problem;TSP)的特性，故可運用 Gilmore and Gomory (1964)極富盛名、複雜度爲 $O(nlogn)$的演算法 (參見 Hall and Sriskandarajah, 1996、Lawler *et al.*, 1989 及 Pinedo, 1995)。

另外，在二機開放工場不考慮工作具連接性現象的排程問題，即 $O_2||C_{max}$ 問題，Gonzalez and Sahni (1976)提出了一個複雜度爲 $O(n)$的演算法。在 Pinedo (1995)的書中也指出，可運用 LAPT (Longest Alternate Processing Time First；即最長相對處理時間者優先)的派工法則來解決此問題。上述文獻中的研究部分，皆爲目前能夠保證在多項式時間內求得其最佳解的演算法或是派工法則。

如 Yao *et al.* (2000) 所提及：$F_2|Blocking|C_{max}$問題事實上比本研究中所探討之 $O_2|Blocking|C_{max}$限制更多。另有Hall and Sriskandarajah (1996) 明白地指出：$F_2|Blocking,separated|C_{max}$ 及 $O_2|Blocking|C_{max}$ 兩種問題同屬NP-complete in the strong sense 的問題。

在附表 1 中，我們整理出與 $O_2|Blocking|C_{max}$ 相關之排程問題之文獻及其所研究問題之複雜度。

2.2 遺傳演算法應用於排程問題

在本節我們先簡介遺傳演算法的架構，再探討文獻中，學者如何運用遺傳演算法求解排程問題。

遺傳演算最主要利用電腦程式來模擬自然界中演化的過程，來解決最佳化(Optimization)的問題。其常運用一群二元字串(Binary String)表示問題的解答，而字元所組成的字串碼即是決定特徵或問題結果的染色體(Chromosome)。選擇(Selection)機制依據每個染色體在母群體中的適應值(Fitness Value)決定染色體繼續生存與否；而適合函數代表著該染色體對外在環境的適應能力，相當於該染色體的性能指標(Performance Index)。故適應值越好的染色體，就有越高的機率生存與繁衍後代，也就是下一代染色體能佔有更多的比例利用遺傳演算的交配及突變等機制，根據現有的族群基因產生下一代的染色體。這個交替的過程會一直持續到設定的結束條件爲止。遺傳演算主要是由一群染色體所構成的族群(Population)，每個子代都有一組染色體字串，此一字串代表著此物種的特性(Characteristics)。依照問題型態與需求，可以透過不同的編碼(Encoding) 方式，表達出不同的子代特性，再透過適應函數顯現出染色體的好壞程度。

目前已有許多研究提出許多的方式改良遺傳演算法，但其基本精神都是根據 Holland (1975, 1992)中所描述的「簡易遺傳演算法」(Simple Genetic Algorithm；SGA)發展而來。SGA 的演算架構如下所示：

```
SGA ()
{
    隨機設定初始群體
    (Initialize Population)
    評估群體中的個體
    (Evaluate Population)
    還沒符合結束條件時，重複執行(While
    Not Stopping)
    {
      選擇好的個體 (Selection)
      進行交配及突變的運算
      (Crossover and Mutation)
      評估群體產生的新個體
      (Evaluate New Population)
    }
}
```

針對不同的最佳化問題，欲應用 GA 來求解時，則必須針對問題形式，仔細斟酌設計編碼方式、適合度函數及操縱基因交配或突變的運算子等資料結構，方能使 GA 的搜尋具全面

性並能有效收斂 GA。

目前已經有相當多的研究應用遺傳演算法求解進行排程問題，在此僅探討與 $O_2|Blocking|C_{max}$ 問題相關之文獻。

在流程工場(Flowshop)，已有 Cleveland and Smith (1989)、Gupta *et al.* (1993)及鄧浩敦(2000)等學者運用遺傳演算法求解其排程問題的最佳解。

在零工工場(Jobshop)排程，是目前可以看到使用 GA 求解排程問題最爲活躍的領域。例如：Falkenaurer and Bouffouix (1991)研究排程工作受先後次序與時間限制影響的問題。Croce *et al.* (1995)利用 GA 以及前瞻式(Look-Ahead)模擬方法處理最小化總流程時間的靜態零工工場排程問題。利用文獻中極少數已經求得最佳解的問題來比較，上述學者的研究結果顯示 GA 會比文獻中其他啓發式方法較佳，且與最佳解的差距在 1%~1.8%之間。Croce *et al.* (1995)並建議結合其他更具穩健性的啓發法則或簡單的數理方法來得到起始解，使 GA 更有效率。

關於 GA 使用在開放工場(Openshop)的文獻目前還相當少，在 Liaw (2000)研究中，使用了混合遺傳演算法(Hybrid GA；HGA)來求解一般化的開放工場問題。Liaw 混合塔布搜尋法(Tabu Search)爲基礎的區域優化方法，來進行個體的搜尋，使得 GA 中的每一世代個體做鄰近區域的優化。Liaw (2000)的研究結果顯示 HGA 求解的品質優於下列其他方法：插入啓發式解法(Insertion Heuristic)、模擬退火法(Simulated Annealing)及塔布搜尋法。另外，Liaw (2000)也針對求解開放工廠排程問題，運用 GA 求解最佳解，測試許多隨機範例，並對於演算法的使用者建議成效較好的遺傳演算元。

相較於其他解法，遺傳演算法能夠在可以接受的時間內，搜尋到不錯的解答（甚至是全域最佳解，Global Optimal Solution），而且 GA 對問題變數的多寡及起始解的優劣等影響相對的比較小；即其有較強韌（Robust）求解的品質。

針對於 $O_2|Blocking|C_{max}$ 問題，在文獻中作者僅發現 Yao *et al.* (2000)所提出之一隨機搜尋演算法求解其最佳解，但尚未有有學者使用遺傳演算法來求解。考量以往遺傳演算法在其他組合最佳化問題出眾的求解的品質，本研究將針對運用遺傳演算法來求解 $O_2|Blocking|C_{max}$ 問題。在完成本研究之探討後，以期未來再與 Yao *et al.* (2000)的隨機搜尋演算法進行比較。

三、遺傳演算法的基本架構

3.1 編碼(Encoding)

進行 GA 之前必須先進行基因編碼(Gene Encoding)的模式。而本研究中所採用實數編碼的方式，稱爲浮點(實)數 GA(Float GA)，每單一個體(individual)分別代表一組可行之排程。事實上，本研究中考慮的工作連接性時，在編碼選擇方式上，採用單一機器的工作順序爲個體，是最直覺之方法；即機器 B 的順序會因爲機器 A 的順序決定時，同時也跟著決定。舉例來說：<1,2,3,4,5,6>代表的意思爲機器 A 的處理工作順序，則在機器 B 的處理工作順序必爲<2,1,4,3,6,5>。這是因爲本研究之最重要假設條件之一，即工作在兩機器之間不能等待也沒有緩衝區存在(即 Blocking or No-waiting)所致。

本研究中因採用實數編碼，每單一個體皆代表一組可行解，而不需經過解碼之程序，因爲此可行解可以直接視爲一個獨立個體或是一組染色體。

3.2 族群數目(Population Size)與初始族群(Initialize Population)

接著遺傳演算法需要產生一個初始的族群，以作爲一個起始點。遺傳演算法中每一世代的個體數目是固定的，故要決定的是每一世代要用多少個體(族群數目)去進行演化(Evolution)。

針對遺傳演算法的起始解方式，本研究中

應用隨機產生方式，也就是根據族群數目等於其工作個數(假設爲 N 個)，採用隨機的方式產生一組 N × N 大小的矩陣，每列皆不重複的順序當作起始解。

3.3 適應函數(Fitness Function)與選擇(Selection)、複製(Reproduction)

在 GA 中適應函數是用來決定一個染色體對問題所處的環境條件的適合程度，即提供衡量每個解答染色體好壞的標準。

遺傳演算法的適應函數，大都以最大化爲目標；對於求取排程最小化之目標函數(例如：製距、平均流程時間等)，Goldberg (1994)及楊宗銘(1995)皆建議爲：

適應函數值＝*MAX*-目標函數值。

因爲本研究的目標函數最小化製距(Minimal Makespan)。因爲「目前群體中最大的目標函數值」可以代表目前所評估之群體特性，故將 MAX 值設爲該值。

選擇這個機制主要模擬自然界適者生存的現象。表現優良的染色體在評估之後，接下來就是挑出那些我們認爲有助於優化或適應目標的個體，使其享有較大的機率，複製到下一個世代。

遺傳演算法中用來模擬物競天擇的選擇機制有很多種，常用的輪盤選擇法(Roulette Wheel Selection)是設計一個輪盤，上面每一格代表一個個體，而其面積大小(或機率值)與該個體的適合度成正比。其選擇機率大小可以由以下式(1)求得：其中 $fit_i(x)$ 爲第 i 個個體之適應函數值，而 RP_i 則爲第 i 個個體被選中之機率(Reproduction Probability)。而

$$RP_i = \frac{fit_i(x)}{\sum_{i=1}^{n} fit_i(x)} \qquad (1)$$

傳統 GA 中所使用的選擇與複製方法，並無文獻特別說明其不適合使用於排程問題；故本研究中採用輪盤選擇法作爲研究中選擇與複製方法。

3.4 交配(Crossover)

基因交配的目的是爲了讓個體互相交換有用的資訊，以使新個體有可能組合出更高的適合度，以達到不斷演化的目標。

在解決排序問題時，一個染色體由許多“不重覆”的數字基因所構成的字串。在這個限制之下，如果以傳統簡易遺傳演算法中所提的運算子(指二進制所產生的字串)來搜尋，很有可能會產生一個個體中有兩個相同的數字基因。爲了避免發生重覆的情況，必須要設計新的運算子，來產生合理的子代。

針對浮點實數編碼，Goldberg (1989)提出下列五種常見交配方法，PMX、LOX、SX、RX 及 CX，本研究再加上一種交配 PBX，進行比較，作爲 GA 解法運用的基準。以下針對本研究所測試的六種交配方法進行介紹：

1. PMX (Partially Matched Crossover)

此方法原本是針對 TSP 問題所發展的交配法，爲 Goldberg (1989)建議較適合排程問題的基因運算子之交配法。PMX 著重於保存各字元在母代染色體上的位置資訊，其作法如下(Goldberg, 1989)：

(1) 隨機產生兩個切點

個體 A：(9 8 4｜5 7 6｜1 3 2)

個體 B：(8 7 1｜2 3 6｜9 5 4)

(2)將個體 A 與個體 B 在兩個切點中的基因互調。

個體 A：(9 8 4｜2 3 6｜1 3 2)

個體 B：(8 7 1｜5 7 6｜9 5 4)

(3)將個體 A 位於切點之外重覆的基因與個體 B 位於切點之外重覆的基因互調。

個體 A：(9 8 4｜2 3 6｜1 7 5)

個體 B：(8 3 1｜5 7 6｜9 2 4)

2. LOX (Linear Order Crossover)

LOX 爲另一個 Goldberg (1989)建議較適合排程問題之基因運算子。其作法如下：

(1)隨機產生兩個切點。

個體 A：(9 8 4｜5 7 6｜1 3 2)

個體 B：(8 7 1｜2 3 6｜9 5 4)

(2)將個體 A 切點中的所有位元，在個體中以* 代替；B 者亦然。

個體 A：(9 8 4｜5 7 *｜1 * *)

個體 B：(8 * 1｜2 3 *｜9 * 4)

(3)將*往中間移動，使得兩個切點中的位元皆爲*。

個體 A：(9 8 4｜* * *｜5 7 1)

個體 B：(8 1 2｜* * *｜3 9 4)

(4)將原本個體 A、B 切點中的位元互調。

個體 A：(9 8 4｜2 3 6｜5 7 1)

個體 B：(8 1 2｜5 6 7｜3 9 4)

LOX 和 PMX 最主要的差別在於 LOX 著重於保存母代染色體上各字元的順序關係。

3. SX (Simple Crossover)

(1)隨機產生一個切點。

個體 A：(9 8 4｜5 7 6 1 3 2)

個體 B：(8 7 1｜2 3 6 9 5 4)

(2)保留個體 A 切點左邊的基因，切點右邊的位元以其在個體 B 的順序填入。

個體 A：(9 8 4｜7 1 2 3 6 5)

個體 B：(8 7 1｜9 4 5 6 3 2)

4. RX (Random Crossover)

(1)隨機產生兩個切點。

個體 A：(9 8 4｜5 7 6｜1 3 2)

個體 B：(8 7 1｜2 3 6｜9 5 4)

(2)切點兩旁的基因保留不變，兩切點中的基因以隨機產生。

個體 A：(9 8 4｜6 5 7｜1 3 2)

個體 B：(8 7 1｜3 6 2｜9 5 4)

5. CX (Cycle Crossover)

(1)在個體 A 中任選一個基因，假設選到 9，其相對在個體 B 爲 1，將其標示起來。

個體 A：(9 8 2 1 7 4 5 6 3)

個體 B：(1 2 3 4 5 6 7 8 9)

(2)個體 A 中基因爲 1 的位元在個體 B 中是 4，將其標示起來。

個體 A：(9 8 2 1 7 4 5 6 3)

個體 B：(1 2 3 4 5 6 7 8 9)

(3)個體 A 中基因爲 4 的位元在個體 B 中是 6，將其標示起來。

個體 A：(9 8 2 1 7 4 5 6 3)

個體 B：(1 2 3 4 5 6 7 8 9)

(4) 個體 A 中基因爲 6 的位元在個體 B 中是 8，將其標示起來。

個體 A：(9 8 2 1 7 4 5 6 3)

個體 B：(1 2 3 4 5 6 7 8 9)

(5)重覆以上的步驟，直到最後的標示回到 9。

個體 A：(9 8 2 1 7 4 5 6 3)

個體 B：(1 2 3 4 5 6 7 8 9)

(6)將個體 A、B 中沒有被標示的基因互換。

個體 A：(9 8 2 1 5 4 7 6 3)

個體 B：(1 2 3 4 7 6 5 8 9)

Liaw (2000)依其測試的數據指出 LOX 運算子在開放工場問題中會得到較好的結果。本研究在第四節中的測試結果也建議採用 LOX 作爲運算方式。

6. PBX (Position Based Crossover)

(1) 隨機產生的兩個切點

個體 A：(9 8 4 5 7 6 1 3 2)

個體 B：(8 7 1 2 3 6 9 5 4)

(2)個體 A 中的 6 與 1 交換位置；個體 B 中的 5 與 6 交換位置即可。

個體 A：(9 8 4 5 7 1 6 3 2)

個體 B：(8 7 1 2 3 5 9 6 4)

3.5 突變(Mutation)

在自然生物系統中，除了物競天擇外，生物常爲了適應生存環境，而產生突變。常見的排序問題基因突變方式有下列六種：(Goldberg, 1989)

1. SM (Swap Mutation)

任選兩個基因，將其互調即可。

(9 8 4 5 7 6 1 3 2)→(9 8 4 1 7 6 5 3 2)

2. PBM (Position-based Mutation)

任選兩個基因，假設爲 5、1，將 7、6、往前移，再將 5 填入原本 基因 7 的位置即可。

(9 8 4 5 7 6 1 3 2)→(9 8 4 7 6 5 1 3 2)

這種方式類似 PMX 著重於保存各字元在父代染色體上的位置資訊之效果，本研究所採用的突變運算子爲 PBM。

3. IM (Inverse Mutation)

任選兩切點，將切點之間的基因反轉。

(9 8 4 | 5 7 6 | 1 3 2)→(9 8 4 | 6 7 5 | 1 3 2)

4. TSM (Three Swap Mutation)

任選三點，採用隨機的方式交換。

(9 8 4 5 7 6 1 3 2) →(9 7 4 5 3 6 1 8 2)

5. ASM (Adjustment Swap Mutation)

任選兩相鄰的點，將其交換。

(9 8 4 5 7 6 1 3 2) →(9 8 4 7 5 6 1 3 2)

6. SHM (Shift Mutation)

任選兩切點，將切點之間的染色體進行位移。

(9 | 4 5 7 6 1 | 3 2) →(9 | 1 4 5 7 6 | 3 2)

四、遺傳演算法中參數的設定

在探討遺傳演算法中討論如何設定其參數之前，必須先確定評估排程效率的指標；如此，我們才有比較不同參數組合設定時，何種參數組合較優的依據。

文獻中一般學者用來評估排程效率的指標可以分爲兩方面：一爲解答之品質，另一方面爲求解時間。

在解答品質的比較方面，，我們採用的解答品質指標爲「平均百分比誤差」(Mean Percentage Error；MPE)」，如下式：

$$MPE = \frac{C^*_{\max} - \text{LowerBound}}{\text{LowerBound}} \times 100\% \quad (2)$$

其中，式(2)中之 Lower Bound(下限)，原本理應設爲問題之全域最佳解(Global Optimum)。但是由於本研究屬於 NP-hard 問題，全域最佳解找尋困難，故以一容易取得的下限取代之，並將下限定義成：

$$\max\{\sum_{j=1}^{n} a_j, \sum_{j=1}^{n} b_j\} \quad (3)$$

其中 a_j, b_j 分別爲工作在兩機器的處理時間；故式(3)中的下限乃指在開放工場中，不考慮工作連接性(或是不等候性)的限制時，所有工作的最短完成時間。

在第三節中，我們已經討論在本研究之前，如：Goldberg (1989)、Liaw (2000)及 Falkenaurer and Bouffouix (1991)等學者，運用遺傳演算法求解具順序性的排程問題(如:TSP 等）時，建議多種的基因運算子。本研究將以隨機範例測試文獻中所建議的各種基因運算子，再針對 $O_2|Blocking|C_{max}$ 問題，提出本研究認爲最適用的基因運算子。

4.1 隨機範例測試設定

因爲我們將製距(Makespan)視爲最佳化的目標時，基本精神在於縮短兩連續工作之間的機器閒置時間。本研究發現機器的閒置時間跟所有工作在機器上的處理時間的變異數(Variance of Processing Time on Each Machine)，具有相當密切的關係。通常，所有工作在機器上的處理時間的標準差（Standard Deviation；即變異數的開平方）較大時，求解複雜度相對增加，求取更佳解的困難度也會隨著增加。

根據以上原因，本研究將工作數目與所有工作在機器上的處理時間的標準差，視爲產生隨機實例的兩個依據。研究中將工作數目分成五組，分別爲 N = 20、30、40、50 及 60 個工作；將工作在機器上處理時間之平均值設定爲 100，將標準差分成四組，分別爲 σ =10、15、20 及 25。舉例來說，工作數爲 20 的情況下，其又可分成四組範例，第一組服從常態分配 $N(100,10^2)$，第二組服從常態分配 $N(100,15^2)$，第三組服從常態分配 $N(100,20^2)$及第四組服從常態分配 $N(100,25^2)$等，其餘依此類推。本研究在每組「工作數目與所有工作在機器上的處理時間的標準差」設定下，產生 30 個範例，再以 MPE (見式 2)作爲評估指標。

4.2 參數的設定

對於遺傳演算法的起始解方式，本研究中所應用的是一般常見的隨機產生方式，也就是族群數目等於其工作個數(假設爲 N 個)，採用隨機的方式產生一組 N × N 大小的矩陣，每列皆不重複的順序當作起始解。

遺傳演算法中有四組基本參數：族群數目(Population Size; P_s)、交配機率(Probability of Crossover; P_c)、突變機率(Probability of Mutation; P_m)及後代數(Number of Offspring ; Gen)，這些參數的設定會影響求解的時間還有品質，根據問題的特性而定。

本研究中設定的族群數目爲問題中所給定的工作數量(參考林恆貞, 1998 所建議)；因爲當工作數量增加，相對的求解空間也會增

加，故必須找到更多的隨機起始點，以免落入局部最佳解。

根據上述 20 組之參數設定組合，每一組各取 5 個隨機範例，設 LOX 爲其交配運算子，PBM 爲其突變運算子；另外，預設突變機率(P_m)=0.5，後代數(Gen)=20。在不同交配機率下，GA 求解所得之 MPE(%)平均值，如圖 4-1：

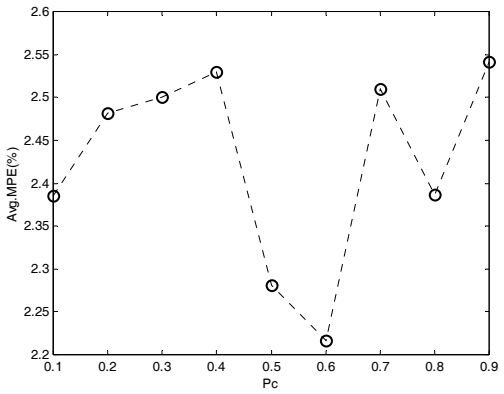

圖 4-1 GA 的交配機率及 MPE(%)的結果比較

由圖 4-1 可知，交配機率可設定爲 0.6。在固定交配機率爲 0.6 時，繼續測試突變機率(P_m)；結果如下圖 4-2：

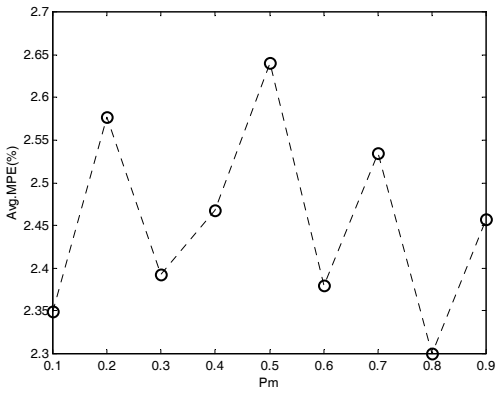

圖 4-2 GA 的突變機率及 MPE(%)的結果比較

由上圖可知，突變機率 P_m 設定爲 0.8 可以有不錯之結果。

由於求解時間也是本研究考慮的重點之一，過多的後代數目將造成求解時間的增加，故本研究將後代數目上限設定爲 100 代，希望於 100 代之中找到符合的代數，利用 P_c=0.6,P_m=0.8，可得到圖 4-3：

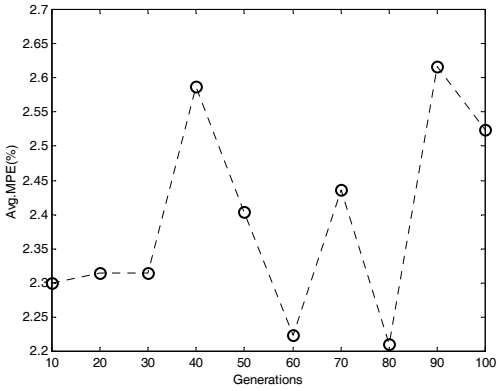

圖 4-3 GA 的後代數及 MPE(%)的結果比較

本研究認爲 60 代所使用的平均執行時間(167.3 sec.)比 80 代所得到的平均執行時間(279.5 sec.)，減少了 40.1%左右，而其 MPE(%)平均值並無明顯改善，所以在參數的選擇上，設定 GA 執行的後代數爲 60 代。

故本研究中遺傳演算法之參數設定將以上列敘述爲基礎；即設(P_s, P_c, P_m, Gen)=(工作數目,0.6,0.8,60)，作爲實驗數據之依據。

4.3 交配及突變運算子設定

在上述的參數設定下，本研究再從 (N, σ)=(40,25)的實驗設定中，選取 30 組隨機範例進行實驗，並求取其 MPE 平均值，所得到結果如表 4-1：

Crossover Operatiors	**LOX**	**PMX**	**RX**	**CX**	**SX**	**PBX**
MPE. (%)	2.05	2.40	2.25	2.30	2.93	2.60

表 4-1 GA 在不同交配方式所得到的 MPE(%)值

將上表結果畫成圖 4-4；圖 4-4 顯示交配運算子 LOX(Linear Order Crossover)其所得到的解會比使用其他運算子來的好。在其他運用遺傳演算法中求解具順序性的排程問題的研究中，另有學者(Falkenaurer and Bouffouix, 1991 及 Liaw, 2000)等建議使用 LOX 爲其交配運算子。

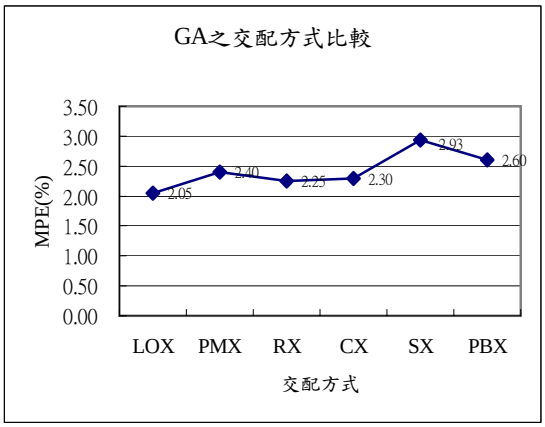

圖 4-4 不同的交配方式之結果

突變運算子部分，由於此運算子主要功能爲在鄰域中找出新的的搜尋點，另需要注意的是防止產生不可行解即可。如同選取交配運算子的做法，我們將不同突變運算子所得的結果列於表 4-2 及圖 4-5：

Mutation Operators	**ASM**	**IM**	**TSM**	**PBM**	**SM**	**SHM**
MPE (%)	2.41	2.40	2.44	2.31	2.37	2.47

表 4-2 GA 在不同突變運算子所得到的 MPE(%)值

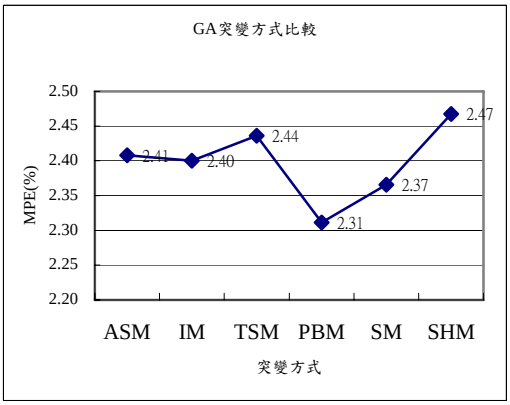

圖 4-5 不同的突變方式之結果

由圖表中觀察可知，不同的突變方式之間差異不大；本研究採用 PBM(Position Based Mutation)作爲突變運算子，因爲 PBM 可以盡量保持在親代中相關位置的資訊。

4.4 遺傳演算法的終止條件

遺傳演算法的終止條件第一種方式爲達到所設定的後代數，則演算法終止，而另一種在文獻中常見者爲目標直達到可接受範圍，即本研究中之 MPE(%)達到某一個定值時，則將演算法終止；本研究採用達到預先設定的後代數作爲終止條件。

在本節中，本研究已經運用隨機實驗爲驗證基礎，完成在遺傳演算法中所需之各項參數與基因運算子之設定。在第五節中，針對遺傳演算法在 $O_2|Blocking|C_{max}$ 問題排程效果的分析，將以本節所得的設定爲實驗之基礎。

五、遺傳演算法的排程結果

本研究乃是利用 Matlab 5.3 (R11, Hanselman and Littlefield, 1996)撰寫遺傳演算法程式；工作環境爲 Pentium MMX Celeron 600MHz 之個人電腦中執行，記憶體大小爲 256MB。運用第四節所得的參數設定，本研究再爲每組(N,σ)產生 30 個隨機實例，以遺傳演算法求解，再取其 MPE(%)平均值。

本研究所得實驗之結果，列於下表 5-1：

MPE%	σ=10	σ=15	σ=20	σ=25
N=20	0.3789	1.0561	1.4392	1.8737
N=30	0.5689	1.0219	1.2868	2.6818
N=40	0.9238	1.8842	2.5248	2.6173
N=50	1.3204	2.3076	2.2920	3.1209
N=60	1.1174	1.6465	2.2618	2.9216

表 5-1 遺傳演算法的平均排程績效(%)

在不同的工作數目及工作處理時間標準差的水準下，我們得到 MPE(%)的趨勢線如圖 5-1 及 5-2：

從圖 5-1 及圖 5-2 觀察得知，本研究以遺傳演算法求解 $O_2|Blocking|C_{max}$ 問題其解答品質之優劣，除了和工作數目的多寡有關之外，事實上還必須考慮工作處理時間標準差之大小。

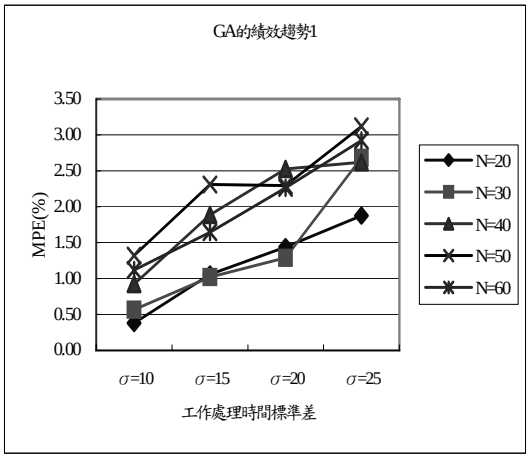

圖 5-1GA 的 MPE 和工作數之趨勢

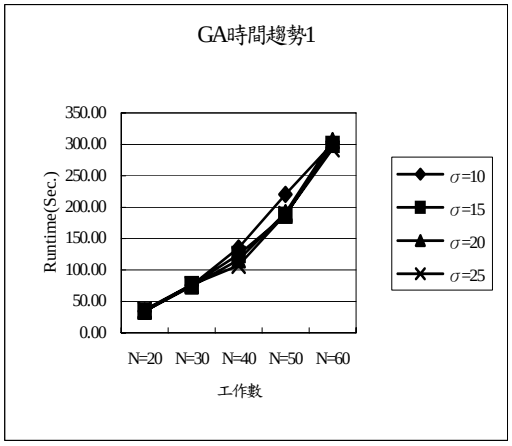

圖 5-3 GA 的 Runtime 和工作數之趨勢

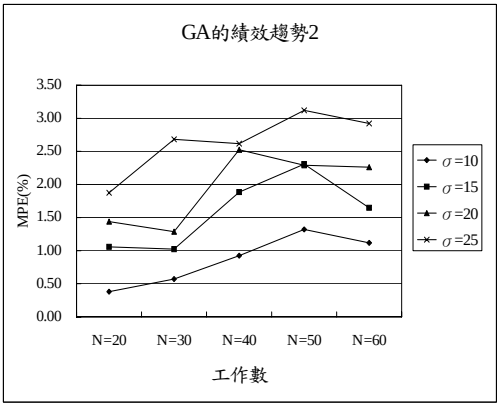

圖 5-2 GA 的 MPE 和工作處理時間標準差趨勢

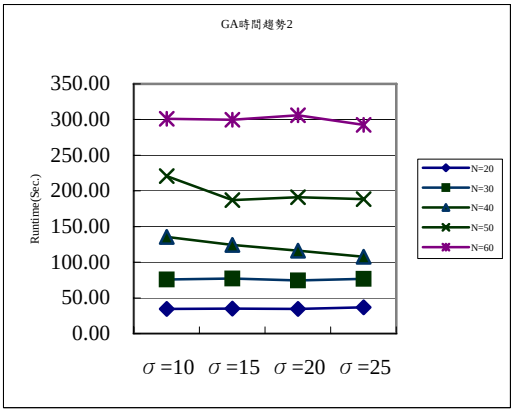

圖 5-4 GA 的 Runtime 和工作處理時間標準差趨勢

在同樣的參數設定下，我們再觀察演算法執行時間與工作數及工作處理時間標準差之間的關係，其結果整理於表 5-2：

Runtime	σ=10	σ=15	σ=20	σ=25
N=20	34.5667	35.0267	34.4700	36.8567
N=30	75.6333	77.0033	74.5167	76.7667
N=40	135.3900	124.5000	116.0967	107.4700
N=50	220.5800	186.9267	191.2500	188.3967
N=60	301.1733	299.9433	305.9000	292.7333

表 5-2　GA 的平均執行時間(Sec.)

在不同的工作數目及工作處理時間標準差的水準下，我們比較遺傳演算法的執行時間，得圖 5-3 及圖 5-4：

比較圖 5-3、圖 5-4，依本研究之觀察，運用 GA 來求解 $O_2|Blocking|C_{max}$ 問題時，求解時間並不會隨著工作處理時間標準差的增加而增加；即對 GA 來說，影響求解時間的變因只有「工作數目」而已。

六、結論及未來研究方向

本研究所探討之二機開放工場具有工作連接性限制的排程問題，即 $O_2|Blocking|C_{max}$ 問題，是屬於 NP-hard 問題，在求取全域最佳解具有相當的困難度。本研究建議運用遺傳演算法，求得所謂的近似最佳解的啓發解；並以平均誤差百分比(MPE; 參考式 2)及演算法執行時間作爲績效衡量指標。

運用遺傳演算法求解 $O_2|Blocking|C_{max}$ 問題，本研究建議使用 LOX(Linear Order Crossover)爲交配運算子，採用 PBM(Position Based Mutation)作爲突變運算子。在參數設定方面，本研究建議族群數目(P_s)、交配機率(P_c)、突變機率(P_m)及後代數(Gen)可設爲(工作數目,0.6,0.8,60)。

文獻中其他學者在探討二機排程如：

$F_2|No\text{-}waiting|C_{max}$ 及 $O_2|Blocking|C_{max}$ 等問題時，常以均勻分配(Uniform distribution)產生隨機範例。本研究發現：工作處理時間的變異程度(標準差)，顯著影響問題的複雜度及遺傳演算法等啓發式解法求解之品質；所以在產生隨機範例時，應該將工作處理時間的變異程度列入考慮。此外，我們也觀察到：本研究的效率評估指標 MPE，即平均誤差百分比，隨著工作數目增加而增加。

本研究在未來可朝下列的方向繼續延伸：

1. 研究中所使用簡單實數(浮點)式 GA(Simple Floating GA)，針對本問題，其區域搜尋的能力不明顯，將來可以考慮和其他區域搜尋的方式結合，成爲所謂之混合式 GA(Hybrid GA)，以提高其效率。
2. 針對 $O_2|Blocking|C_{max}$ 問題，將來可以考慮特別設計因應其特性的基因運算子，增強遺傳演算法搜尋的能力；例如：不以考慮單一工作爲主，而以兩兩一組的的「工作對」之結構來設計運算子。
3. 另外，可以與其他啓發演算法進行比較其求解的效率（求解的時間與解答的品質），如：模擬退火法(Simulated Annealing)、塔布搜尋(Tabu Search)及類神經網路(Neural Network)等，或是新發展的啓發式演算法。
4. 還可以考慮不同之排程目標函數，例如：平均流程時間、總流程時間…等等。

參考文獻

1. Cleveland, G. A. and Smith, S. F. (1989), “Using Genetic Algorithm to Solving Flow Shop Release” *Proc. of the Third Int. Conf. Genetic Algorithm*, pp.160-169.
2. Croce, F. D., Tadei, R. and Volta, G. (1995), “A Genetic Algorithm for the Job Shop problem, ” *Computer and Operation Research*, vol. 22, pp.15-24
3. Falkenaurer E. and Bouffouix, S. (1991), “A Genetic Algorithm for Job Shop Scheduling, ” *Proceedings of the 1991 IEEE International Conference on Robotics and Automation Sacramento*, April, pp. 824-829
4. Gilmore, P. C. and Gomory, R.E. (1964), “Sequencing a one state variable machine: a solvable case of the traveling salesman problem”, *Operations Research*, vol. 12, pp.655 - 679.
5. Goldberg, D. E. (1989), *Genetic Algorithm in Search, Optimization, and Machine Learning*. Addison – Wesley, Mitchell,
6. Goldberg, D. E. (1994), “Genetic and evolutionary algorithm come of age”, *Communications of the ACM*, vol.37, No.3, pp.113-119
7. Gonzalez, T. and Sahni, S. (1976), “Open shop scheduling to minimize finish time,” *Journal of the Association for Computing Machinery*, vol. 23, pp. 665 - 697.
8. Graham, R.L., Lawler, E.L., Lenstra, J.K. and Rinnooy Kan, A.H.G., (1979), “Optimization and approximation in deterministic sequencing and scheduling: a survey”, *Annals of Discrete Mathematics*, vol. 5, pp. 287 - 326.
9. Gupta, M. C., Gupta, Y. P., and Kumar, A. (1993), “Minimizing Flow time Variance in a Single machine system using Genetic Algorithms,” *European Journal of Operational Research*, vol. 70, pp, 289-303
10. Hall, N. G. and Sriskandarajah, C. (1996), “A survey of machine scheduling problems with blocking and no-wait in process,” *Operations Research*, vol. 44, pp. 510 - 525.
11. Hanselman, D. and Littlefield, B. (1996), *Mastering Matlab: A Comprehensive Tutorial and Reference*, Prentice Hall, Upper Saddle River, New Jersey.
12. Holland, J. H. (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan press, Ann. Arbor
13. Holland, J. H. (1992), *Adaptation in Natural and Artificial System: an introductory analysis with applications to biology, control, and artificial intelligence*, MIT press.
14.

15. Johnson, S.M. (1954), “Optimal two and three-stage production schedules with setup times included,” *Naval Research Logistics Quarterly*, vol. 1, pp. 61 - 68.
16. Kammoun, H and Sriskandarajah, C. (1993), “The complexity of scheduling jobs in repetitive manufacturing systems”, *European Journal of Operational Research*, vol. 70, pp. 350 - 364.
17. Lawler, E. L., Lenstra, J.K., Rinnooy Kan, A.H.G. and Shmoys, D.B., (1989), "Sequencing and Scheduling: Algorithms and Complexity," *Report BS-R8909, Centre for Mathematics and Computer Science*, P.O. Box 4079, 1009 AB Amsterdam, the Netherlands.
18. Liaw, C.-F. (2000), “A hybrid genetic algorithm for the open shop scheduling problem,” *European Journal of Operational Research*, vol. 124, pp.28-42.
19. Logendran, R. and Sriskandarajah, C. (1993), “Two-machine group scheduling problem with blocking and anticipatory setups ”, *European Journal of Operational Research*, vol. 69, pp. 467 - 481.
20. Pinedo, M. *Algorithm, and Systems*,” Prentice Hall, Englewood Cliffs, New Jersey.
21. Sriskandarajah, C. and Ladet, P. (1986), “ Some no-wait shops scheduling problems: complexity aspects”, *European Journal of Operational Research*, vol. 24, pp. 424 - 438.
22. Sahni, S. and Cho, Y. (1979), “Complexity of scheduling jobs with no-wait in process”, *Mathematics of Operations Research*, vol. 4, pp. 448 - 457.
23. Yao., M. J., Hanijanto S., and Elmaghraby S. E. (1999) “Simple Heuristics for the two Machine Openshop Problem With Blocking”, *Journal of Chinese Institute of Industrial Engineers*, vol. 17, pp. 537-547
24. 林恆貞 (1998),“重複性製造系統排程效率之比較—以 No-wait Problem 爲例”, 國立成功大學工業管理研究所碩士論文。
25. 鄧浩敦 (2000)，“混合遺傳演算法於流程工場排程之應用”,逢甲大學工業工程研究所碩士論文。
26. 楊宗銘 (1995)，“遺傳演算法在多途程排程問題之探討”，中原大學工業工程研究所碩士論文。

Problems	Complexity	References
$F_2 \| \| C_{max}$	P [註1]	Johnson (1954)
$F_2 \| blocking \| C_{max}$	P	Gilmore & Gomory (1964)
$F_2 \| blocking, m_1=1, m_2=2 \| C_{max}$	CS [註2]	Kammoun & Srikandarajah (1993) Srikandarajah & Ladet (1986)
$F_2 \| blocking, separated \| C_{max}$	CS	Logendran & Srikandarajah (1993)
$F_3 \| blocking \| C_{max}$	CS	Hall & Srikandarajah (1996)
$J_2 \| blocking \| C_{max}$	CS	Kammoun & Srikandarajah (1993) Sahni & Cho (1979)
$O_2 \| \| C_{max}$	P	Pinedo (1995) Gonzalez &Sahni (1976)
$O_2 \| blocking \| Cmax$	CS	Kammoun & Srikandarajah (1993)

附表 1 一些與本研究相關問題的複雜度分析結果

註 1:P 代表在多項式時間內可以求解

註 2:CS 爲 NP-hard in strong sense

A Genetic Algorithm for the Two Machine Openshop Scheduling Problem with Blocking

Chung-Wai Lai, Ming-Jong Yao and Tsueng-Yao Tseng

Department of Chemical Engineering
Tung-Hai University

ABSTRACT

In this paper, we consider a two-machine scheduling problem in an openshop with blocking jobs. We are given the processing times of n blocking jobs on both machines, and the objective is to minimize the makespan. Symbolically, we are dealing with the problem $O_2|Blocking|C_{max}$. The results from our numerical experiments suggest that one should use LOX (Linear Order Crossover) and PBM (Position-Based Mutation) as the genetic operators if one would like use the genetic algorithm (GA) to solve the $O_2|Blocking|C_{max}$ problem. And, we state useful guidelines for setting the parameters in GA, for instance, population size, crossover rate, mutation rate and number of generations, etc. From the 600 random examples, we also observe that the number of jobs and the variance of the processing time for the jobs significantly affect the performance of the GA. Indeed, our study provides valuable decision support information for the decision makers who attempts to use the GA to solve the $O_2|Blocking|C_{max}$ problem.

Key words: Scheduling, Blockiing, No waiting, Openshop Problem, Genetic Algorithms