考慮有中斷點的學習效應下等速率平行機之排程問題

Uniform Parallel Machines Scheduling with a Truncation Learning Effect

莊惟翔 Wei-Hsiang Chuang[1]
林暘桂 Yang-Kuei Lin[2*]

摘要

近年來，排程問題考慮學習效應的研究越來越受到關注，本研究考慮以總加工時間為基礎之學習效應，且考量到實際加工現場的學習效應並不會永無止盡的持續下去，因此本研究亦在學習效應式子中加入中斷點，成為一個新的學習效應公式，探討在等速率平行機的環境下求解總加權延遲最小化的排程問題，並提出螞蟻演算法加入起始解以及區域搜尋法測試不同規模大小的題目以驗證本研究之螞蟻演算法的有效性及效率。

關鍵字：等速率平行機、學習效應、螞蟻演算法

Abstract

This research considers the problem of scheduling jobs on uniform parallel machines to minimize total weighted tardiness under a truncation learning effect. An ant colony optimization (ACO) algorithm incorporating heuristic initial solution and local search has been proposed to solve the studied scheduling problem. Some experiment runs has been tested to verify the effectiveness and efficiency of proposed ACO algorithm.

Keywords: Uniform Parallel Machine, Learning Effect, Ant Colony Optimization.

壹、前言

生產排程(scheduling)是一種資源分配的問題，也就是將系統中所提供的有限資源加上各種條件的限制之下，決定出最佳的資源分配。而一般學者針對不同的機器加工環境、工件生產的限制以及不同的目標函數，發展不同的求解方法，而求解方法有可能是可求得最佳解的線性規劃模式(linear programming model, LP)或是可求得近似解的萬用啟發式演算法(meta-heuristic)等等。從眾多的文獻探討中可發現平行機的排程問題在生產排程的研究中十分常見也極具重要性，從學術的觀點來看，針對平行機所發展的演算法，也可應用於求解單機的排程問題；而從實務的觀點來看，現實生活中的製造系統，通常一個工作站有超過一台以上的機器同時在運作，因此探討平行機的排程問題有相當的重要性存在。根據

[1]逢甲大學工業工程與系統管理學系研究生。
[*2]逢甲大學工業工程與系統管理學系助理教授(聯絡地址：407 台中市西屯區文華路 100 號，聯絡電話：04-24517250 轉 3638，E-mail: yklin@ fcu.edu.tw)。

Pinedo(2008)的定義，所謂的平行機即為系統內擁有多部機台，所有機台加工功能皆一樣，但機台間可能存在不同的加工速度，而所有工件只要經過任一機台加工後即可離開此系統。傳統上平行機可分為等效平行機(identical parallel machines)、等速率平行機(uniform parallel machines)以及不相關平行機(unrelated parallel machines)。等效平行機是指同一工件在不同的機器上，處理時間皆相同；等速率平行機是指機器和機器間存有一等速率的關係，本研究即為探討此機器生產環境，此類的生產環境在製造業中十分常見，例如在製造系統的加工中心，普遍存在著舊機器與陸續新購入機器同時存在的情形，新機器一般比舊機器有較快的處理速度，故形成加工中心內擁有數個平行機器同時進行加工，但機器與機器間存有一等速率的關係；不相關平行機則是指工件和機器間不存在任何關係。而平行機的排程問題即是考慮在各種條件的環境限制下，決定如何分配工件至機台加工以及分配至同一機台上的工件之處理順序，並找出目標函數的最佳解或近似解。

近年來，學者們漸漸注意到學習效應存在於現實生活的製造系統中，例如機器需要人工操作的部分就容易產生學習效應，操作員可能因為對機器及工件的熟悉度而影響加工的時間，亦即操作員經由重複性的加工動作中學習並累積經驗，故工件所需要的處理時間會隨著前面已完成的工件加工時間總和的增加，而愈來愈短，這便是所謂的學習效應，為了使研究內容更接近實際上的加工狀況，因此在求解時考量學習效應的影響。其中較為大家所了解的學習效應分為兩大類：一種是以位置為基礎的學習效應(position-based learning effect)，另一種則是以總加工時間為基礎的學習效應(sum-of-processing-time based learning effect)。在第一種類型中，每個工件的實際加工時間會隨著機器上所排的工件數量越多而遞減最後趨近於 0；而第二種類型裡，每個工件實際加工時間會隨著前面已完成工件的總加工時間越大而遞減最後趨近於 0。本研究考慮第二種學習效應，但是因為實際加工現場的學習效應並不會永無止盡的持續下去，也就是學習效應到某個程度就會停止，所以本研究使用之學習效應參考 Cheng 等人(2009)所提出之學習效應式子(1)，並加入 Wu 等人(2011)之學習效應中斷法則式子(2)，兩者綜合如式子(3)所示，

$$p_{j[r]} = p_j (1 + \sum_{l=1}^{r-1} \ln p_{[l]})^a \ , \tag{1}$$

式子(1)中，每個工件 j 有原始處理時間($p_j \geq 1$)，a 為學習效應參數($a = \log_2 LR$, $a \leq 0$)，LR 為學習效率(learning rate)，例如在 80%的學習效率下，$a = \log_2 80\% = -0.322$；在 70%的學習效率下，$a = \log_2 70\% = -0.515$，而學習效率愈小，代表學習能力愈佳。l 為機器上排在第 l 個位置的工件，r 為工件 j 目前要排上機器的位子，所以當要計算工件 j 排在機器上的第 r 個位置的處理時間則會考慮到第 r 個位置之前的所有工件原始處理時間，但為了避免因為前面已完成的工件原始加工時間太大而造成學習效應很快的就降到趨近於 0，因此加入了對數(logarithm)的機制使學習效應不會有急遽下降的情形。例如現在要將工件 2(原始加工時間為 50)排上機台的第二個位置，而前面已經有排上工件 1 其原始加工時間為 500，假設 a 為-0.322，在沒有對數機制的情況下，工件 2 在機台上的第二個位置加工

時間為 $p_{2[2]} = p_2(1+\sum_{l=1}^{2-1} p_{[1]})^{-0.322} = 50(1+500)^{-0.322} = 6.75$；在有對數機制的情況下，$p_{2[2]} = p_2(1+\sum_{l=1}^{2-1} \ln p_{[1]})^{-0.322} = 50(1+\ln 500)^{-0.322} = 50(1+6.21)^{-0.322} = 26.46$，而這就是以總對數加工時間為基礎(sum-of-logarithm-processing-times-based)的學習效應。

$$p_{j[r]} = p_j \max\{(1-\frac{\sum_{l=1}^{r-1} p_{[l]}}{\sum_{l=1}^{n} p_l})^a, \gamma\}\ ， \tag{2}$$

式子(2)中，每個工件 j 有原始處理時間($p_j \geq 1$)，$a \geq 1$為學習效應指標，l 為機器上排在第 l 個位置的工件，r 為工件 j 目前要排上機器的位子，γ 為中斷參數($0 < \gamma < 1$)，當工件 j 要排上第 r 個位置時，會考慮學習效應公式以及中斷參數，當學習效應已經小於中斷參數γ時，則會用γ取代之。而學習效應公式$(1-\frac{\sum_{l=1}^{r-1} p_{[l]}}{\sum_{l=1}^{n} p_l})^a$中，$\sum_{l=1}^{r-1} p_{[l]}$ 代表前面已完成的工件原始加工時間總和、$\sum_{l=1}^{n} p_l$ 代表所有工件在機台上的總加工時間，其學習效應公式含意為考慮尚未加工的工件總處理時間的比例，若比例越高，代表前面已完工的工件原始處理時間愈少；若比例越低，代表前面已完工的工件原始加工時間愈多，所以學習效應會使後面加工的工件實際處理時間較短。

$$p_{ij[r]} = p_{ij} \max\{(1+\sum_{l=1}^{r-1} \ln p_{i[l]})^a, \gamma\}\ ， \tag{3}$$

參考式子(1)與(2)後，本研究使用式子(1)之學習效應公式，其考量是因為本研究認為學習效應應該以前面完成加工的工件總原始加工時間為依據較合理，而不是依據尚未加工之工件還有多少比例的總加工時間；並加入式子(2)之中斷參數發展成為式子(3)，因為本研究亦認為學習效應並不會永無止境的持續下去。其中 p_{ij} 為工件 j 在機器 i 上所需要的處理時間($p_{ij} \geq 1$)，r 為現在工件 j 所要排上第 i 台機器的位置，l 為機器上排在第 l 個位置的工件，γ 為學習效應的中斷點($0 < \gamma < 1$)，在式子(3)中，因為學習效應參數 a 的存在，因此$(1+\sum_{l=1}^{r-1} \ln p_{i[l]})^a$的值會介於 0 跟 1 之間，而且會隨著前面已完成工件之總加工時間增加而從 1 漸漸降為 0，為了避免學習效應使實際加工時間一直遞減最後趨近於 0，因此我們加入中斷參數γ使學習效應只會降到γ即停止，以下針對本研究所使用之學習效應利用範例做一個說明。表 1 為 2 台機器 4 個工件(2*m*4*n*)在等速率平行機上的處理時間。而在此例題中，我們將γ設為 0.5，學習效率設為 70%，則 $a = \log_2 70\% = -0.515$。

表 1 2*m*4*n* 之工件處理時間

	j_1	j_2	j_3	j_4
m_1	40	30	60	10
m_2	60	45	90	15

假設現在要建構一排程(如圖 1 所示)，一開始我們選到了工件 3 要排在第 1 台機器上，因此代入公式可以得到 $p_{13[1]}=60\max\{(1)^{-0.515},0.5\}=60$ ，所以工件 3 在機台 1 的第一個位置實際加工時間為 60；接著假設選到了工件 2 要排在機台 1 的第二個位置上，因此我們由公式得到 $p_{12[2]}=30\max\{(1+\ln 60)^{-0.515},0.5\}=15$，由於學習效應的關係，工件 2 的實際加工時間比原始的加工時間 30 還短，後面的工件以此類推(如表 2 所示)，因此我們得知，工件會因為前面工件加工的總時間越長而使實際加工時間越來越短，但因為有學習效應中斷限制，所以當前面工件的總加工時間到達某一個時間點時，中斷點將會取代學習效應，因為在真正的實務上，學習效應並不會永無止盡的越來越快，而是會有一個極限存在。

表 2 學習效應計算步驟

	r=1	r=2
m_1	$p_{13[1]}=p_{13}\max\{(1)^{a},\gamma\}$ $=60\max\{(1)^{-0.515},0.5\}=60$	$p_{12[2]}=p_{12}\max\{(1+\ln p_{1[1]})^{a},\gamma\}$ $=30\max\{(1+\ln 60)^{-0.515},0.5\}=15$
m_2	$p_{24[1]}=p_{24}\max\{(1)^{a},\gamma\}$ $=15\max\{(1)^{-0.515},0.5\}=15$	$p_{21[2]}=p_{21}\max\{(1+\ln p_{2[1]})^{a},\gamma\}$ $=60\max\{(1+\ln 15)^{-0.515},0.5\}=30.55$

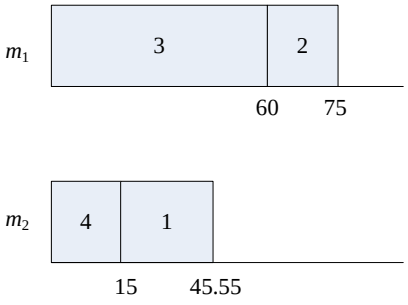

圖 1 學習效應應用於平行機排程問題上

本研究為考慮有中斷點的學習效應下，等速率平行機求解總加權延遲最小化之排程問題。而在目標函數方面，本研究考慮在現實的公司生產系統中，都不希望工件有延遲完工的情況發生，因為會造成額外的成本產生，而且不同的工件其重要性也不同，最重要的工件為最不想要發生延遲情況的工件，因此權重越大的工件若延遲($C_j-d_j>0$)，產生的懲罰就愈大，所以求解目標希望能讓總加權延遲時間最小化。

至於在求解方法裡，則有較多種類的求解方法可用來求解，其中包含可求得最佳解的線性規畫模式以及分枝界限法(branch and bound, b&b)等皆可求得排程問題的最佳解，但因為本研究所探討之問題為 NP-hard 的問題，因此無法使用以上的方法求解；此外也有可求得近似解的基因演算法(genetic algorithm, GA)、模擬退火法(simulated annealing, SA)、粒子群演算法(particle swarm optimization, PSO)以及類電磁演算法(electromagnetism-like

mechanism, EM)等等，而本研究是使用螞蟻演算法來求解，這些方法雖然只能求得近似解，但求解速度較 LP 或 b&b 快，能符合實務的需求。

在排程領域中，有許多學者致力於各種不同的排程相關研究，其中 Graham 等人(1979)將排程問題利用 $\alpha|\beta|\gamma$ 來表示，α 為在機器加工的環境；β 為工件生產的限制；γ 為目標函數。而本研究探討的機器環境是等速率平行機，以 Q 做表示；工件加工的限制有以總加工時間為基礎的學習效應以及中斷點，以 LE_t, *truncation* 表示；求解的目標函數為總加權延遲時間最小化，以 $\sum w_j T_j$ 表示，w_j 代表各工件的權重、$T_j = \max(0, C_j - d_j), j = 1,...,n$，$C_j$ 與 d_j 分別為工件 j 的完工時間與到期日；綜合上述的介紹，本研究的所要探討的排程問題為等速率平行機上，考慮有中斷點的學習效應，求解最小化總加權延遲時間，則可以使用 $Q|LE_t, truncation|\sum w_j T_j$ 來表示。

貳、文獻探討

一、學習效應應用於排程問題

(一)、以位置為基礎的學習效應(position-dependent)

Biskup(2008)為首位將學習效應觀念導入排程領域的學者，並驗證 SPT(shortest processing time)法則可求得單機於學習效應下最小化總完工時間($1|LE|\sum C_j$)之最佳解；Mosheiov(2001)分別探討單機與平行機於學習效應下的排程問題，更針對平行機於學習效應下最小化總完工時間的排程問題($P|LE|\sum C_j$)提出線性規劃模式求解；Zhao 等人(2004)考慮相同工件於學習效應下的等速率平行機排程問題，個別針對最小化總加權完工時間($Q|LE, p_j = 1|\sum w_j C_j$)與最小化最大延遲時間($Q|LE, p_j = 1|L_{\max}$)發展啟發式演算法(heuristic)，且皆可求得問題之最佳解；Eren 和 Güner(2009)不僅發展 0-1 整數規劃模式(0-1 integer programming model)求解單機考慮學習效應之最小化總延遲時間排程問題($1|LE|\sum T_j$)，也提出禁忌搜尋法(tabu search)與模擬退火法求解相同排程問題以相互比較其效果優劣；Wu 等人(2007)針對單機考慮學習效應之最大延遲時間最小化排程問題($1|LE|L_{\max}$)，發展分支界限演算法、啟發式演算法，以及模擬退火法解之並探討其優劣。

(二)、以時間為基礎的學習效應(time-dependent)

Kuo 和 Yang(2006[1], 2006[2])探討學習效應於單機的排程問題，並證明總時程最小化問題($1|LE_t|C_{\max}$)與總完工時間最小化問題($1|LE_t|\sum C_j$)皆可由 SPT 法則解得最佳解；Wang(2008)和 Wang 等人(2008)也探討了許多單機於學習效應下的一些排程問題($1|LE_t|\sum w_j C_j$、$1|LE_t|L_{\max}$、$1|LE_t|\sum U_j$)，在特殊的條件下可使用一些簡易的派工法則(dispatching rule)解得最佳解；由於 Cheng 等人(2009)發現現有的學習效應會使處理時間(processing time)遞減過快，並針對此現象提出改善，也證明 SPT 法則可解得單機考慮學習效應之總時程最小化問題($1|LE_t|C_{\max}$)以及總完工時間最小化問題($1|LE_t|\sum C_j$)的最佳解；Wu 等人(2011)認為學習效應的效果不是永無止盡的，而是學習到某個程度就會停止，

故針對此現象提出學習效應有中斷點的觀念，並驗證 SPT 法則亦可求解單機考慮有中斷點學習效應下之最小化總時程問題($1|LE_t,truncation|C_{\max}$)與最小化總完工時間問題($1|LE_t,truncation|\sum C_j$)的最佳解。

二、螞蟻演算法應用於排程問題

螞蟻演算法是由 Dorigo 等人(2003)觀察蟻群覓食的行為所衍生出來的萬用啟發式演算法，其原理為螞蟻透過費洛蒙(pheromone)的氣味成為彼此溝通的管道，在覓食的過程中，走過的路徑螞蟻會遺留費洛蒙，同時費洛蒙也會慢慢蒸發，漸漸的較多螞蟻在覓食過程中走過的路徑(通常也是離食物最近的路徑)會累積較多的費洛蒙，因此也較容易吸引後來的螞蟻走相同的路徑覓食，久而久之，蟻群會找到從螞蟻窩與食物之間的最短路徑。螞蟻演算法成功的被用來求解許多 NP-hard 的問題，如路徑問題，指派問題，排程問題，子集合問題，及機器學習問題等。Besten 等人(2000)提出了在單機環境下使用螞蟻演算法求解總加權延遲最小化的排程問題($1\|\sum w_jT_j$)，並有效結合幾個簡單的區域搜尋法使找出來的解與最佳解的距離非常接近；Merkle(2003)提出了在單機環境下使用螞蟻演算法求解總延遲時間最小化的排程問題($1\|\sum T_j$)，並在螞蟻演算法的費洛蒙路徑(pheromone trail)和演算法資訊(heuristic information)做改良，使所找到的目標函數能最佳化；Ying 和 Liao(2003)使用螞蟻演算法求解單機總加權延遲最小化的排程問題($1\|\sum w_jT_j$)；Holthaus 和 Rajendran(2005)提出了改良的螞蟻演算法，簡稱 FACO(fast ant colony optimization)，求解單機總加權延遲最小化的排程問題($1\|\sum w_jT_j$)，並跟現存的啟發式演算法做比較，其結果顯示 FACO 能求得較佳的解；Liao 和 Juan(2007)提出了在單機環境下考慮順序相依整備時間(sequence dependent setup times)的限制使用螞蟻演算法求解總延遲時間最小化的排程問題($1|s_{jk}|\sum T_j$)，並在螞蟻演算法的費洛蒙路徑加入區域搜尋法做改良，其結果與現存的遺傳基因演算法和禁忌搜尋法做比較，均有較佳的求解效果；Zhou 等人(2007)使用螞蟻演算法求解不相關平行機下總加權延遲最小化的排程問題($R\|\sum w_jT_j$)，在結合有效的起始解及區域搜尋法後，得到的解比原始的螞蟻演算法佳；Mönch(2008)使用螞蟻演算法求解不相關平行機總加權延遲最小化的排程問題($R\|\sum w_jT_j$)；Behnamian 等人(2009)結合螞蟻演算法、模擬退火法及變數鄰近搜尋法(variable neighborhood search)求解等效平行機下考慮相依整備時間的限制下總時程最小化的排程問題($P|s_{ij}|C_{\max}$)；Srinivasa Raghavan 和 Venkataramana(2009)則提出使用螞蟻演算法求解等效平行機總加權延遲最小化的排程問題($P\|\sum w_jT_j$)；Mönch 和 Almeder(2009)則使用螞蟻演算法求解等效平行機考慮批量(batch)和不相容(incompatible)的限制，求解總加權延遲最小化的排程問題($P|p-batch,incompatible|\sum w_jT_j$)。

根據文獻探討的結果，目前尚未有文獻發表與本研究完全相同的排程問題，因為本研究之學習效應為參考兩位學者的學習效應所延伸，利用延伸之學習效應加入本研究所探討的問題，並使用螞蟻演算法加入起始解以及區域搜尋法求解$Q|LE_t,truncation|\sum w_jT_j$ 的排程問題。

參、研究方法

本章節將介紹如何利用螞蟻演算法求解考慮有中斷點的學習效應下，等速率平行機之總加權延遲時間最小化排程問題，其中還包含了本研究所提的派工法則求得起始解以及使用區域搜尋法來增加解的多樣性。

一、參數定義

本節將所有使用到之參數統一表列出來(表 3)，其中包含學習效應、螞蟻演算法等參數。

表 3 參數定義

參數	定義
m	機器數。
n	工件數。
i	代表機器 i 的指標。
j	代表工件 j 的指標。
r	代表位置 r 的指標。
n_i , $i=1,2,...,m$	機器 i 上目前分配的工件數量。
a	學習效應參數。
p_{ij} , $\mathrm{i}=1,2,...,\mathrm{m}, j=1,2,...,n$	工件 j 在機器 i 上的的原始處理時間(processing time)。
C_j , $j=1,2,...,n$	工件 j 的完工時間(completion time)。
w_j , $j=1,2,...,n$	工件 j 的權重(weight)。
d_j , $j=1,2,...,n$	工件 j 的到期日(due dates)。
t_i , $i=1,2,...,m$	機器 i 目前可開始加工的時間。
T_j , $j=1,2,...,n$	工件 j 的延遲時間 $T_j=\max(0,C_j-d_j)$, $j=1,...,n$ 。
J	用於演算法中儲存工件 j 的集合。
J^*	另一個用於演算法中儲存工件 j 的集合。
I	用於演算法中儲存機器 i 的集合。
α	影響費洛蒙的權重參數。
β	影響演算法資訊的權重參數。
q_{i0}	使用者自訂之機率門檻 $0\le q_{i0}\le 1$ 。
q_{j0}	使用者自訂之機率門檻 $0\le q_{j0}\le 1$ 。
ρ_{local}	區域費洛蒙(local pheromone)的揮發率(evaporation rate)， $0<\rho_{local}<1$ 。
ρ_{global}	全域費洛蒙(global pheromone)的揮發率， $0<\rho_{global}<1$ 。
$\eta_i, i=1,...,m$	機器 i 的演算法資訊(heuristic information)。
$\tau_{ij}, i=1,...,m, j=1,...,n$	螞蟻在工件 j 行經機器 i 路徑上留下的費洛蒙值。
$\eta_{ij}, i=1,...,m, j=1,...,n$	工件 j 在機器 i 的演算法資訊。
Q	控制殘餘費洛蒙的固定參數。
$ITER$	螞蟻演算法的迭代(iteration)次數。

ANTS	螞蟻演算法的人工螞蟻(artificial ants)數量。

本研究主要探討考慮有中斷點的學習效應下，求解等速率平行機總加權延遲時間之排程問題。其求解流程為先使用 ATC-I 與 NEH_WEDD 產生排程之起始解，之後利用螞蟻演算法求解，並加入了區域搜尋法，以下將做一系列的詳細介紹。

根據 Srinivasa Raghavan 和 Venkataramana(2009)表示，一般求解目標函數為總加權延遲時間時可利用以下多種派工法則:EDD(early due date)、MDD(modified due date)、WSPT(weighted shorted processing time)、ATC(apparent tardiness cost)，而本研究所使用的為 Lin 等人(2011)以 ATC 為基礎進行延伸的派工法則，稱為 ATC-I。當使用 ATC-I 產生一個初始排程後，參考 Naderi 等人(2010)中提到之 NEH_EWDD 稍做修改後為 NEH_WEDD 用於強化 ATC-I 所求得之起始解，經過題目測試後發現兩者合併使用會較單獨使用任一種的起始解較好，因此使用 ATC-I+NEH_WEDD 為本研究之起始解，以下用 ATC-NEH 代表本研究之起始解並詳細介紹其運作步驟。

(一) ATC-NEH

ATC-NEH 執行步驟如下。

STEP1: 設定工件集合 J:={1,…,n}，以及初始化機器 i 上一開始可以開始的加工時間皆為 0，t_i=0,i=1,2,…,m。

STEP2: 計算所有機台 i 上現在可開始的加工時間 t_i,i=1,2,…,m，接著選最快可開始加工之機器 $i^* \in \min\{t_i\}, i = 1,...,m$，若同時有兩個以上最早可開始加工時間相同的機器，則隨機選取其中一台機器。

STEP3: 參考式子(4)，計算各工件的 $I_{i^*j}, j \in J$ 值，選擇工件 $j^* \in \max I_{i^*j}$。

$$I_{ij} = \frac{w_j}{p_{ij[r]}} \exp\left(\frac{-\max(d_j - t_i - p_{ij[r]}, 0)}{k\overline{p}}\right) \tag{4}$$

由於學習效應的關係，原本的處理時間 p_j 改為 $p_{ij[r]}$，其中 $p_{ij[r]}$根據式子(3)計算，r 為工件 j 目前可以排入的位置，$\overline{p}$ 代表尚未排入機器的工件平均處理時間，k 為比例參數(scaling parameter)。

STEP4: 重新選擇工件 j^*所要排入的機器 i。計算工件 j^*排在所有機器 i 上的延遲時間，以工件排入機器不會延遲的為優先選擇，如果工件 j^*可排在 1 台以上的機器不會延遲，工件選擇在處理時間最小的機器 i^{**} 排入。如果工件 j^*排在所有機器上均為延遲，選擇延遲時間最小的機器 i^{**} 排入。

STEP5: 更新工件集合 $J = J \setminus j^*$ 移除所選取的工件以及更新機器 i^{**} 上目前可以開始的加工時間 $t_{i^{**}} = t_{i^{**}} + p_{i^{**}j^*[r]}$。

STEP6: 重複 STEP2-5，直到 $J=\phi$，所產生的完整排程存入 π。

STEP7: 設定工件集合 J^*:={1,…,n}。

STEP8: 由 J^* 中找出 w_j/d_j 最大的工件 j^{**} 並從 π 中拿出工件 j^{**}，計算每台機器上的工件數 n_i, i=1,2,…,m，插入每台機器上的 n_i+1 個位置上，其中 i=1,2,…,m，針對插入每個位置後所產生的新排程計算 *TWT*，最後將工件排至 *TWT* 最小的位置上，以下利用一個例子說明之。

假設現在由 w_j/d_j, j=1,2,…n，計算後得到 j^{**} 為工件 1，則工件 1 從排程拿出來，並計算機台 1 目前的工件數量 n_1=3，機台 2 目前的工件數量 n_2=1；因此工件 1 在機台 1 上有 n_1+1=4 個位置可以排，在機台 2 有 n_2+1=2 個位置可以排，工件 1 排在每個位子上皆會計算一次 *TWT*，最後將工件 1 放在 *TWT* 值最小的位置上。

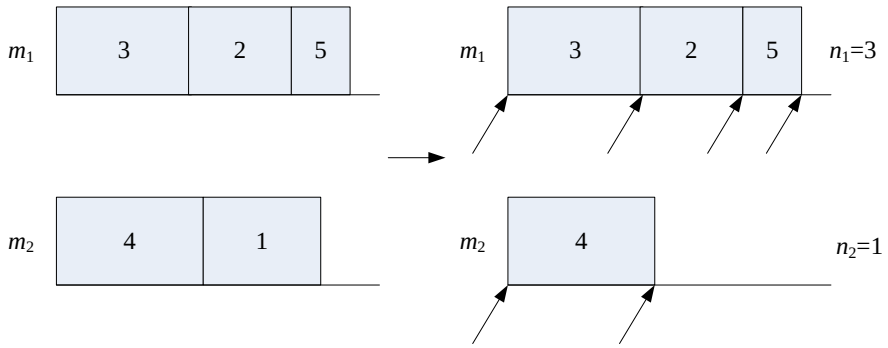

圖 2 STEP8 範例說明

STEP9: 排完一個工件便更新一次排程 π，並將排過的工件 j^{**} 從未排過的集合中移除 $J^*=J^*\setminus j^{**}$。

STEP10: 重複 STEP8-9，直到 $J^*=\phi$。

(二)螞蟻演算法

本節參考 Srinivasa Raghavan 和 Venkataramana(2009)和 Mönch(2008)之螞蟻演算法，針對本研究修改後之螞蟻演算法在機器的選擇、工件的選擇、費洛蒙的更新、區域搜尋方法做詳細的介紹，執行步驟如下。

STEP1: 設定初始參數:*ITER*、*ANTS*、α、β、ρ_{local}、ρ_{global}、Q、q_{i0}、q_{j0}、$\tau_0=1/(\mathrm{ANTS}*\mathrm{ATC}-\mathrm{NEH}_{twt})$、$TWT_{best}$(ATC-NEH 產生的起始解)、*iter*=1(目前迭代次數)、*ants*=1(目前人工螞蟻數量)、工件集合 J:={1,...,n}、機器集合 I:={1,...,m}，以及機器 i 目前可以開始的加工時間 $t_i=0$, i=1,2,...,m。

STEP2: 機器 i 選擇，方式如下，$i^* = \begin{cases} \arg max\{\eta_i\}, i \in I & if\ q_i \le q_{i0} \\ P_i = \dfrac{\eta_i}{\sum_{\forall p \in I} \eta_p}, i \in I & otherwise \end{cases}$，其中機器的演算法資訊 $\eta_i = 1/t_i$。當 $q_i > q_{i0}$，依照機率分配 p_i 隨機選擇機器 i^*。

STEP3: 工件 j 選擇，方式如下，$j^* = \begin{cases} \arg\max\{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta\}, j \in J & if\ q_j \le q_{j0} \\ P_{ij}, & otherwise \end{cases}$，其中工件的演算法資訊 $\eta_{ij} = \left\{ \dfrac{w_j}{p_{ij[r]}} \exp\left(\dfrac{-\max(d_j - t_i - p_{ij[r]}, 0)}{k\bar{p}} \right) \right.$，當 $q_j > q_{j0}$，依照機率分配 $P_{ij} = \begin{cases} \dfrac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{\forall j \in J} [\tau_{ij}]^\alpha [\eta_{ij}]^\beta}, & if\ j \in J \\ 0, & otherwise \end{cases}$ 隨機選擇工件 j^*。

STEP4: 重新選擇工件 j^*所要排入的機器 i。計算工件 j^*排在所有機器 i 上的延遲時間，以工件排入機器不會延遲的為優先選擇，如果工件 j^*可排在 1 台以上的機器不會延遲，工件選擇在處理時間最小的機器i^{**}排入。如果工件 j^*排在所有機器上均為延遲，選擇延遲時間最小的機器i^{**}排入。

STEP5: 將工件 j^* 排入機器 i^{**} 的路徑執行區域費洛蒙的更新，$\tau_{i^{**}j^*} = (1-\rho_{local})\tau_{i^{**}j^*} + \rho_{local}\tau_0$。

STEP6: 更新工件集合 $J = J \setminus j$，移除所選取的工件以及更新機器 i^{**}上目前可以開始的加工時間 $t_{i^{**}} = t_{i^{**}} + p_{i^{**}j^*[r]}$。

STEP7: 重複 STEP2-6，直到 $J = \phi$ 產生 ACO 之排程。

STEP8: 將找到的排程使用區域搜尋方法增加解的多樣性，其執行方式是依據 ACO 產生之排程，並找出該排程中加權延遲($w_j T_j$)最大的工件，將工件移至該機台上的第一個位置，若是能求得較小的 *TWT* 值則更新排程，否則跳至 STEP9。

STEP9: 將完成的排程對相鄰兩個工件進行交換的動作(swap)，以改善總加權延遲時間，方法如下:當找到完整排程時會對此排程同機台的工件做兩兩交換的動作，若工件交換後總加權延遲時間較佳，繼續下面工件兩兩交換。

STEP10: 計算總加權延遲時間(*TWT*)，如果 $TWT < TWT_{best}$，則 $TWT_{best} = TWT$，令 *ants*=*ants*+1，如果 $ants \le ANTS$，更新工件集合 $J := \{1,...,n\}$，以及機器 i 上目前可以開始的加工時間 $t_i = 0$, $i = 1,2,...,m$，回到 STEP2；如果 *ants*>*ANTS*，跳至 STEP11。

STEP11: 將目前所找到最好的解 TWT_{best} 排程做全域費洛蒙更新，$\tau_{ij}^{best} = (1-\rho_{global})\tau_{ij}^{best} + \rho_{gloaal}\Delta\tau_{ij}$，其中 $\Delta\tau_{ij} = \dfrac{Q}{TWT_{best}}$。令 *iter*=*iter*+1，如果

iter<=*ITER*，則 *ants*=1，回到 STEP2；如果 *iter* >*ITER*，跳至 STEP12。

STEP12: 當 *iter* >*ITER* 即符合停止條件(termination criterion)，此時 TWT_{best} 為最終解。

肆、數據測試分析

本研究所產生之測試題目大小包含兩種類型：4*m*20*n* 與 10*m*100*n* 分別代表 4 台機器處理 20 個工件以及 10 台機器處理 100 個工件的情況。處理時間(p_{ij})為 p_j 乘以每台機器之速率 v_i 產生 1 到 100 之間的處理時間，其中 p_j 與 v_i 皆是由 Uniform[1,10]隨機產生，權重(w_j)為 Uniform[1,10]隨機產生，根據 Potts 和 Van Wassenhove(1982)，到期日(d_j)由 Uniform[P(1-T-R/2), P(1-T+R/2)]隨機產生，其中 $P=(\sum_{i=1}^{m}\sum_{j=1}^{n}p_{ij})/m^2$，$T$ 為延遲因子，R 為相對範圍，本研究考慮較難的題目設定，測試 T=0.8, R=0.4 時的總加權延遲時間，因為此類題型的到期日較緊，所產生的題目較有挑戰性，ATC-NEH 中的比例參數 k_1 使用 22 種不同的參數(0.2, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.4, 2.8, 3.2, 3.6, 4.0, 4.4, 4.8, 5.2, 5.6, 6.0, 6.4, 6.8, 7.2)，並使用總加權延遲時間最小的比例參數當作起始解，ACO 參數設定值 *ITER*=250、*ANTS*=20、Q=1、α=1、β=3、$\rho_{local}=0.01$、$\rho_{global}=0.01$、$q_{i0}=0.9$、$q_{j0}=0.9$。使用的電腦 CPU 為 Intel(R) Core(TM)2 Duo CPU E8400 @ 3.00GB, 3.24GB RAM，演算法以 Dev-C++ 4.9.9.2 軟體撰寫。

一、起始解、螞蟻演算法、b&b 在 4*m*20*n* 之求解比較。

本節將針對本研究所使用之螞蟻演算法做一系列測試分析，其中包含與 Liaw 等人(2003)利用分支界限演算法(b&b)求得 $R\|\sum w_jT_j$ 問題之最佳解做比較(4*m*20*n*)，此時將本研究之學習效應停止準則 γ 改為 1 即為無學習效應之加工時間，藉此比較本研究所提的螞蟻演算法及起始解的求解品質與求解效率；在大題目時(10m100n)，我們將學習效應之參數 a 設為-0.5(學習效率約 71%)，停止準則 γ 設為 0.5，測試螞蟻演算法(ACO)、加上起始解之螞蟻演算法(ATC-NEH+ACO)以及有加入區域搜尋法之螞蟻演算法(ATC-NEH+ACO+LS)做比較，而測試結果分別顯示於表 4 與表 5。

表 4 4*m*20*n* 之 *TWT* 比較

題目編號	b&b(opt)	ATC-NEH	ATC-NEH+ACO+LS
1	1336	1417	1336
2	1398	2766	1398
3	925	1025	925
4	690	862	690
5	2801	2891	2801
6	258	258	258
7	855	954	862
8	357	357	357
9	373	398	373
10	593	645	596
11	1428	1428	1428
12	788	867	788
13	1437	1437	1437

14	1679	1679	1679
15	1023	1087	1023
16	539	559	539
17	391	391	391
18	479	610	484
19	1840	1939	1842
20	457	457	457
Average *TWT*	982.5	1101.4	983.2
$\frac{(\text{algorithm - opt})}{\text{opt}} \times 100\%$	—	12.10%	0.09%
Average computation time(s)	5122.94	0.00	2.56

由 4*m*20*n* 的比較結果來看，本研究所使用之起始解距離最佳解約 12.1%，但經過螞蟻演算法之後可以將 *TWT* 值降至與最佳解距離 0.09%，在求解時間上最佳解平均一題需花費 5122 秒與本研究之 ATC-NEH+ACO+LS 的 2.56 秒相差懸殊，因此可以發現本研究之螞蟻演算法在求解時可以發揮很不錯的效果。

表 5 10*m*100*n* 之 *TWT* 比較

題目編號	ACO	ATC-NEH+ACO	ATC-NEH +ACO+LS
1	1573.80	1056.95	1056.95
2	2254.77	2096	2096
3	2116.69	1712.39	1712.39
4	1297.40	790.01	790.01
5	971.46	673.96	673.96
6	1668.18	1477.85	1469.58
7	2992.27	2595.05	2595.05
8	2713.81	2491.3	2491.3
9	4239.14	3205.82	3205.82
10	2525.30	2025.47	2025.47
11	291.39	267.34	202.28
12	634.02	566.75	559.74
13	493.13	405.65	405.65
14	2295.04	2002.38	2002.38
15	2211.37	1634.45	1634.45
16	1972.49	1405.47	1405.47
17	4856.29	3965.41	3965.41
18	2020.01	1600.08	1600.08
19	7993.47	6200.02	6200.02
20	2737.79	1636.3	1636.3
average	2392.89	1890.433	1886.42
$\frac{\text{Algorithm} - (\text{ATC} - \text{NEH} + \text{ACO} + \text{LS})}{\text{ATC} - \text{NEH} + \text{ACO} + \text{LS}}$	26.85%	0.21%	—
Average computation time(s)	11.52	11.55	57.70

由表 5 得知，只有使用螞蟻演算法(ACO)與加了起始解之螞蟻演算法(ATC-NEH+ACO)的求解效果約有 27%的差距，而 ATC-NEH+ACO 與 ATC-NEH+ACO+LS 只有差距不到 1%並無明顯差異，因此得知此區域搜尋法使用在本研究的求解上沒有很明顯的效果。

伍、結論與未來方向

本研究為考慮有中斷點的學習效應下，利用螞蟻演算法求解等速率平行機最小化總加權延遲時間之排程問題。其中提出了 ATC-NEH 當作本研究之起始解，經過螞蟻演算法之運算後，之後亦加入了區域搜尋法設法使目標函數 *TWT* 更小，其中 4*m*20*n* 與沒有學習效應之最佳解做比較，發現本研究之螞蟻演算法與最佳解比較有不錯的效果，因為平均求得的解只有 0.09%的差距，但是在求解時間上卻有很大的差別；而在 10*m*100*n* 下，本研究測試 ACO、ATC-NEH+ACO 以及 ATC-NEH+ACO+LS 比較其結果，測試結果顯示 ATC-NEH+ACO+LS 能求到較好的近似解。

在未來的研究上，可以考慮學習效應下加入工件釋放時間($Q|LE,r_j|\sum w_jT_j$)或是設置時間($Q|LE,s_{jk}|\sum w_jT_j$)等不同的限制。

參考文獻

Behnamian, J., Zandieh, M., & Fatemi Ghomi, S. M. T. (2009). Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm. *Expert Systems with Applications*, 36, 9637-9644.

Besten, M. D., Stutzle, T., & Dorigo, M. (2000). Ant colony optimization for the total weighted tardiness problem. *Lecture Notes in Computer Science*, 611-620.

Biskup, D. (2008). A state-of-the-art review on scheduling with learning effects. *European Journal of Operational Research*, 188, 315-329.

Cheng, T. C. E., Lai, P. J., Wu, C. C., & Lee, W. C. (2009). Single-machine scheduling with sum-of-logarithm-processing-times-based learning considerations. *Information Sciences*, 179, 3127-3135.

Eren, T. & Guner, E. (2009). A bicriteria parallel machine scheduling with a learning effect. *International Journal of Advanced Manufacturing Technology*, 40, 1202-1205.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy, Kan A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5, 287-326.

Holthaus, O., & Rajendran, C. (2005). A fast ant-colony algorithm for single-machine scheduling to minimize the sum of weighted tardiness of jobs. *Journal of the Operational Research Society*, 56, 947-953.

Kuo, W. H., & Yang, D. L. (2006)[1]. Minimizing the makespan in a single machine scheduling problem with a time-based learning effect. *Information Processing Letters*, 97, 64-67.

Kuo, W. H., & Yang, D. L. (2006)[2]. Minimizing the total completion time in a single-machine scheduling problem with a time-dependent learning effect. *European Journal of Operation Research*, 174, 1184-1190.

Liaw, C. F., Lin, Y. K., Cheng, C. Y., & Chen M. C. (2003). Scheduling unrelated parallel machines to minimize total weighted tardiness. *Computers & Operations Research*, 30, 1777-1789.

Liao, C. J., & Juan, H. C. (2007). An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups. *Computers and Operations Research*, 34, 1899-1909.

Lin, Y. K., Pfund, M. E. & Fowler, J. W. (2011). Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems. *Computers & Operations Research*, 38, 901-916.

Merkle, D. (2003). Ant colony optimization with global pheromone evaluation for scheduling a single machine. *Applied Intelligence*, 18, 105-111.

Mönch L. (2008). Heuristics to Minimize Total Weighted Tardiness of Jobs on Unrelated Parallel Machines. *IEEE Conference on Automation Science and Engineering*, 572-577.

Mönch, L., & Almeder, C. (2009). Ant colony optimization approaches for scheduling jobs with incompatible families on parallel batch machines. *Multidisciplinary International Conference on Scheduling : Theory and Applications*, 105-114.

Mosheiov, G. (2001). Parallel machine scheduling with a learning effect. *Journal of the Operational Research Society*, 52, 1165-1169.

Naderi, B., Tavakkoli-Moghaddam, R., & Khalili, M. (2010). Electromagnetism-like mechanism and simulated annealing algorithms for flowshop scheduling problems minimizing the total weighted tardiness and makespan. *Knowledge-Based Systems*, 23, 77-85.

Srinivasa Raghavan, N. R., & Venkataramana, M. (2009). Parallel processor scheduling for minimizing total weighted tardiness using ant colony optimization. *International Journal of Advanced Manufacturing Technology*, 41, 986-996.

Pinedo, M. (2008). Scheduling Theory, Algorithms, and Systems. Prentice Hall, 3rd ed.

Potts, C. N., & Van Wassenhove, L. N. (1982). A decomposition algorithm for the single machine total tardiness problem. *Operations Research Letters*, 1, 177-181.

Wang, J. B. (2008). Single-machine scheduling with general learning functions. *Computer and Mathematics with Applications*, 56, 1941-1947.

Wang, J. B., Ng, C. T., Cheng, T. C. E., & Liu, L. L. (2008). Single-machine scheduling with a time-dependent learning effect. *International Journal of Production Economics*, 111, 802-811.

Wu, C. C., Lee, W. C., & Chen, T. (2007). Heuristic algorithms for solving the maximum lateness scheduling problem with learning considerations. *Computers & Industrial Engineering*, 52, 124-132.

Wu, C. C., Yin, Y., & Cheng, S. R. (2011). Some single-machine scheduling problems with a truncation learning effect. *Computers and Industrial Engineering*, 60, 790-795.

Ying, K. C., & Liao, C. J. (2003). An ant colony system approach for scheduling problems. *Production Planning & Control: The Management of Operations*, 14, 68-75.

Zhao, C., Zhang, Q., & Tang, H. (2004). Machine scheduling problems with a Learning effect. *Dynamics of Continuous, Discrete and Impulsive Systems Series A: Mathematical Analysis*, 11, 741-750.

Zhou, H., Li, Z., & Wu, X. (2007). Scheduling unrelated parallel machine to minimize total weighted tardiness using ant colony optimization. *International Conference on Automation and Logistics*, 132-136.