國立交通大學
工業工程與管理學系

碩士論文

共用機台之跳躍式不允許等待流程式生產排程

Scheduling for Jumping No-Wait Flowshop with a Starting Common Machine

研究生：蔡宗翰
指導教授：許錫美 博士

中華民國九十七年六月

共用機台之跳躍式不允許等待流程式生產排程

Scheduling for Jumping No-Wait Flowshop with a Starting Common Machine

研究生：蔡宗翰　　Student：Tsung-Han Tsai

指導教授：許錫美 博士　　Advisor：Dr. Hsi-Mei Hsu

國立交通大學

工業工程與管理學系

碩士論文

A Thesis

Submitted To Department Engineering and Management College of Management

National Chiao Tung University

In Partial Fulfillment of the Requirements

For the Degree of Master

in

Industrial Engineering

June 2008

Hsin-Chu, Taiwan, Republic of China

中華民國九十七年六月

共用機台之跳躍式不允許等待流程式生產排程

研究生：蔡宗翰　　指導教授：許錫美　博士

國立交通大學工業工程與管理學系

摘要

排程問題在產業界中扮演著重要的課題，本研究提出 JNWF-SCM 排程問題的演算法。所謂 JNWF-SCM 生產線，表示在流程式生產線中，工件一旦進入生產線加工，就不能有待機的情況發生，且所有工件的前段加工是在同一機台作業，後段作業則會依據產品不同而傳送至某些機台來加工。因此加工過程中，工件除了有不可待機的條件之外，工件途程有特殊的共用機台(starting common machine)特性以及跳機台(jumping)特性。本研究根據 JNWF-SCM 排程問題的特性，以禁忌搜尋法(tabu search)為架構，發展一套演算法(ITS)來求解，以最小化總排程時間為目標，決定工件的投料時間。本研究設計了一系列的案例來執行，並以解的總排程時間及其穩定度、與求解時間作為演算法的評判績效，實驗結果發現本研究所提出的方法有較佳的表現，大型問題上有更明顯的優勢。

關鍵字：no-wait、流程式生產、總排程時間、禁忌搜尋法

Scheduling for Jumping No-Wait Flowshop with a Starting Common Machine

Student : Tsung-Han Tsai Advisor : Dr. Hsi-Mei Hsu

Department of Industrial Engineering and Management

National Chiao Tung University

Abstract

This study proposed an algorithm to solve the jumping no-wait flowshop with a starting common machine (JNWF-SCM) scheduling problem. JNWF-SCM indicates that if a job is released to a flowshop production line, then its operations must be carried out with no delay in the production line. All jobs must be processed in the starting machine and at least one machine in the rest of the others. JNWF-SCM has two special properties, starting common machine and the jumping several machines from starting common machine to the end of the production line. A tabu search approach is proposed for the JNWF-SCM scheduling problem with the makespan minimization criterion. The results of the simulated experiment show that our approach performs better than other algorithms, especially in the large scale problems in term of both solution quality and computational time.

Keywords：no-wait, flowshop, makespan, tabu search

誌　　謝

本篇論文得以順利完成，首先要感謝我的指導教授許錫美博士不厭其煩的指導與督促。每當研究遇到瓶頸時，老師總能有效的點破思考盲點、啟發研究構想。更重要的是，老師傳授給我研究精神必須執著的道理。還要感謝巫木誠教授、彭德保教授和洪一薰教授，在論文口試期間給予我寶貴的意見。向以上各位老師致上最高的敬意。

感謝學長英森、泰盛，研究夥伴廷勳、馨吟、智偉、景閎、幼容、俊霖、冠賢，學弟妹雨欣、成權、以及室友茂鈞。有了你們的陪伴與支持，才能在撰寫論文的這段期間有充實且溫馨的時光。因為有你們，這兩年將成為我永久的回憶。

最後感謝我的爸媽以及妹妹，全力的支持我、勉勵我，因為他們的陪伴與容忍，讓我在研究的道路上，沒有任何的後顧之憂。特別是感謝爸媽在我求學的這十幾年來對我的栽培。謹以此論文獻給我最敬愛的家人。

蔡宗翰 2008/6/18

于 風城交大

目錄

表目錄

圖目錄

第一章　緒論

1.1　研究背景與動機

排程(scheduling)問題是生產與製造管理上重要的課題，良好的排程是善用生產環境中的人力、機器、工具等製造資源來安排工件的生產優序，提高資源的使用率與減少閒置時間，以達成所設定的管理目標。

排程的問題可依工件在生產環境加工的特性分為數種不同的情境型態，其中流程式生產(flowshop)是生產環境中最常見的加工型態，所謂的流程式生產是指各工件所經過的加工機台順序皆相同，在現實產業中，此流程式生產型態存在於以大量製造為主的產業中。

流程式生產型態中，有些生產線因品質要求，工件加工過程中不得有任何待機(no-wait)的特性，工件一旦投入生產線中，從第一部機台開始加工，直到最後一部機台加工完畢前，都不得有任何待機或滯留的情況發生。此特殊生產製程型態稱之為「不允許等待流程式生產」(No-wait flowshop，NWF)。

NWF 的情境廣泛的存在於鋼鐵業、化工業、食品製造業、製藥業、玻璃工業及半導體業；以半導體產業為例，晶圓前端製程中的蝕刻製程，是將一批晶圓(工件)投入酸槽做加工，而這些酸槽是以流線式排列，進行晶圓上需蝕刻的化學反應，作業完畢後再以機器手臂迅速的將晶圓移動至下一個化學槽進行作業，由於酸槽內化學反應的作業時間需要非常精準，否則會造成不良品的出現，因此一旦工件進入酸槽，作業完畢後不得有任何待機發生，為了確保不會有任何待機事件，在投料進入生產線之前可能會以比較長的間隔時間，才投料下一批晶圓，造成機台產能的損失。

由於產業結構的變遷，業者傾向客製化產品來滿足顧客需求，提高服務水準，因此客戶導向的生產已經是普遍的現象，為了產品的差異化，工件在生產流程上可能會跳過某些機台的加工，進而達到不同需求的產品。以製藥業為例，有些藥物因為市場需求，所以須製作成液體藥物與固體藥片兩種，生產過程中，藥

品在生產線中段是以液態狀呈現，但是部份藥品必須經過脫水乾燥機台或固態成型機台來做成固體藥片，而以液態呈現的液體藥物則會跳過那些機台，直接到後段的包裝機台，造成後段生產線上有跳躍式作業(jumping operations)的現象。食品麵粉製造業中，因為市場的需求，餐飲業用的麵粉或大賣場販售的麵粉都是採購大包裝麵粉，但是超級市場或零售消費者的需求則是小包裝。因此麵粉製造商在生產線後段，會有大包裝的包裝機台還有小包裝的包裝機台，需要大包裝的麵粉在生產流程上就會傳送至大包裝機台，而需要小包裝的麵粉在生產流程上就會被傳送至小包裝機台。再以玻璃製造業為例，因為滿足顧客的需求，對於玻璃的顏色、規格、樣式等有不同的需求，因此在生產線加工上會加入不同化學物質來做不同顏色的玻璃，也會因為規格不同而用不同的壓制機器來加工，導致玻璃原料在生產線上就會依照產品特性而傳送至某些機台到後面機台來加工。總而言之，兼具有前段共用機台與後段跳躍式特性的生產線，工件會在生產線前段的機台做相同的加工作業，在後段的機台則會依據產品需求不同而跳至某些機台來加工。

故許多製程的生產線不僅結合了 No-Wait Flowshop 特性，還含有前段共用機台(starting common machine)與後段跳躍式(jumping)加工的現象，此種生產情境，稱之為「共用機台之跳躍式不允許等待流程式生產」(Jumping No-Wait Flowshop with a Starting Common Machine，JNWF-SCM)。

以下分別針對 NWF 與 JNWF-SCM 生產線做介紹：

1.2 No-Wait Flowshop(NWF)生產線

不允許等待流程式生產(NWF)與一般流程式生產(Flowshop)的最大相異處，在於 NWF 排程問題中，工件在加工過程不能有所等待的情況發生，因此工件可能會延遲投料，以避免等待的情況發生。

以下舉例說明 NWF 排程問題的加工特性，假設有四個工件依序在四部機台上加工，各工件在各機台的加工時間如下表 1.1：

表 1.1　NWF 的工件加工時間表

加工時間(分) 機台 / 工件	M_1	M_2	M_3	M_4
J_1	3	6	2	4
J_2	3	4	4	2
J_3	4	3	3	3
J_4	3	3	2	4

假設工件的加工優序為 $J_1 \rightarrow J_2 \rightarrow J_3 \rightarrow J_4$ 時，工件加工的甘特圖如圖 1.1 所示：

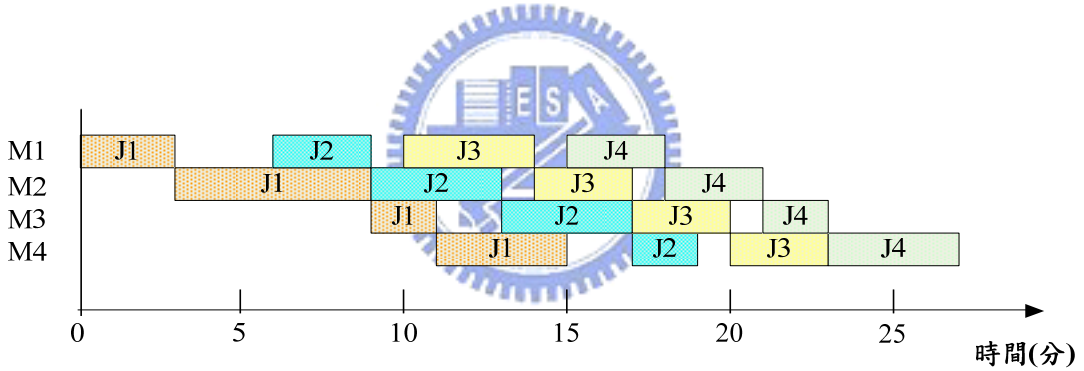

圖 1.1　NWF 之甘特圖

為了確保 J_2 在 M_1 加工後能立刻進入 M_2 加工，J_2 需延遲 3 分鐘後才投料於 M_1，方可使 J_2 在第 9 分鐘進入 M_2 加工，避免造成 J_2 待機的情況發生；同理 J_3 也必須延遲 2 分鐘才投料於 M_1，使 J_3 在第 17 分鐘進入 M_3 加工。以上兩段延遲投料的時間是為了能滿足 NWF 特性，才造成了在機台 M_1 的閒置。根據 NWF 特性，工件必須經過所有機台加工，因此各機台所加工的工件順序會等於工件投料的順序，此現象與一般性 flowshop 排程問題相同。

1.3 Jumping No-Wait Flowshop with a Starting Common Machine (JNWF-SCM)生產線

由於共用機台之跳躍式特性的生產線，工件會在生產線前段的機台做相同的加工作業，在後段機台則會依據產品需求不同而傳送至某些機台來加工。所以設計一個符合 JNWF-SCM 特性的生產線，在多部機台中，各個工件皆必須經過第一部機台來加工，其他機台則依照產品特性，可允許工件傳送至任何機台來加工。

以下舉例說明 JNWF-SCM 排程問題的加工特性，假設有四個工件依序在五部機台加工，且必須經過第一部機台加工，各工件在各機台的加工時間如表 1.2：

表 1.2　JNWF-SCM 的工件加工時間表(1)

加工時間(分) 機台 / 工件	M_1	M_2	M_3	M_4	M_5
J_1	2	-	6	-	3
J_2	4	10	-	2	-
J_3	1	-	3	4	4
J_4	2	-	-	-	2

假設已知工件的加工優序(佔用機台的優序)為: $J_1 \rightarrow J_2 \rightarrow J_3 \rightarrow J_4$ 時，工件加工的甘特圖如圖 1.2 所示：

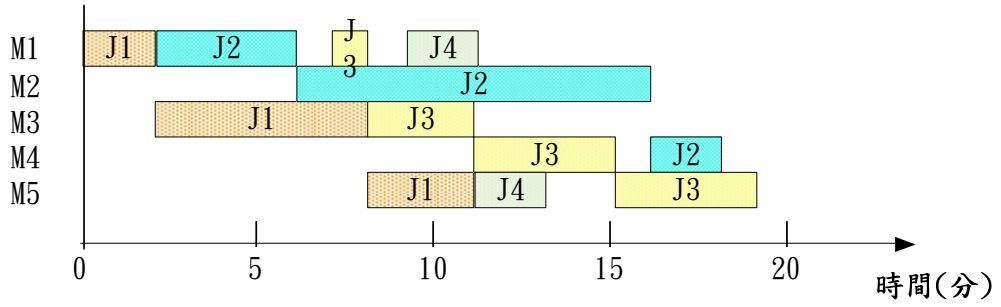

圖 1.2　JNWF-SCM 之甘特圖

工件 J_1 需在 M_1、M_2、M_5 機台加工；工件 J_2 需在 M_1、M_2、M_4 機台加工；工件 J_3 需在 M_1、M_3、M_4、M_5 機台加工；工件 J_4 需在 M_1、M_5 機台加工。所有工件的加工途程先後順序仍然是 $M_1 \rightarrow M_2 \rightarrow M_3 \rightarrow M_4 \rightarrow M_5$，但會跳過某些機台加工。

工件加工優序首先是 J_1，所以最先佔用 M_1、M_2、M_5 等機台；工件 J_2 佔用 M_1、M_2、M_4 機台接其後；工件 J_3 亦隨之佔據 M_1、M_3、M_4、M_5 機台；最後，工件 J_4 佔據機台 M_1、M_5 機台，但由於機台 M_5 中，工件 J_1 與 J_3 之間的閒置時間夠長，導致 J_4 可以插入其中來加工。根據上述 JNWF-SCM 排程問題的結果，可以看出機台 M_1 的工件加工序為 $J_1 \rightarrow J_2 \rightarrow J_3 \rightarrow J_4$，機台 M_2 的工件加工序為 J_2，機台 M_3 的工件加工序為 $J_1 \rightarrow J_3$，然而機台 M_4 與機台 M_5 的工件加工序卻分別變成為 $J_3 \rightarrow J_2$ 與 $J_1 \rightarrow J_4 \rightarrow J_3$。此現象說明，在 JNWF-SCM 生產環境下，後投料的工件，在某些機台可能比先投料的工件先加工。

各工件在各機台加工序的變化現象，增加計算各工件在各機台加工的起始時間與結束時間的困難度，進而影響總排程時間計算的複雜度，顛覆以往 NWF 排程問題的特性，也因為這種特性大幅增加解題的難度及複雜度，從中可說明 NWF 問題是 JNWF-SCM 問題的特例，也就是 JNWF-SCM 問題比 NWF 問題更具一般性。

因此若能有一套良好的方法來解決 JNWF-SCM 排程問題，可以在各大產業上運用，以期減少產品製造週期時間，提高資源效率，增加產出，並且減少在製品存貨量，降低成本，故激發本研究以 JNWF-SCM 排程問題為研究課題。

1.4 研究基本假設

本研究主要探討 JNWF-SCM 排程問題，基本假設與傳統 NWF 排程問題的假設相同，相異的只有下方列舉的最後一點：

(1) 已知工件總數。

(2) 已知機台總數。

(3) 每一站機台都只有一台。

(4) 所有工件之間彼此獨立，不具有任何優先順序的限制。

(5) 所有工件在一開始即可進行加工。

(6) 不考慮機器故障的問題。

(7) 不考慮機器的整備時間。

(8) 同一時間每部機台只能處理一項工件，工件在同一時間也只能在一部機台上加工，各機台加工中的工件加工完後才可下機。

(9) 工件於各機器的加工時間皆已知，且為已知常數(稱為確定性排程，Deterministic Scheduling)。

(10) 各工件必須經過第一部機台加工，其餘機台則至少經過一部來加工。

1.5 研究目的

Rock[16]已研究證明二部機台以上之 NWF 排程問題之複雜度為 NP-hard，也就是無法在有限時間內求出最佳解；然而，JNWF-SCM 問題比 NWF 問題更具一般性，所以本研究問題亦屬於 NP-hard 問題。

本研究目的是以 JNWF-SCM 生產線下，最小化總排程時間(makespan)為目標，求得各工件的投料時間點，而總排程時間等於第一個工件開始加工的時間至結束所有工件加工的時間總長度。為了能夠在短時間內求出一個近似最佳工件排程，本論文提出一套適合 JNWF-SCM 排程問題的禁忌搜尋法(Tabu Search，簡稱 TS)，藉由禁忌搜尋法擁有較短運算時間的優勢，更能縮短排程的決策時間。

1.6 論文架構

本論文計畫書架構共分六章。第一章為緒論，介紹研究背景、研究動機、研究目的與架構；至於本論文所涉及的相關文獻，包含了 No-Wait Flowshop 問題與 No-Wait Jobshop 問題，在第二章加以討論與回顧；第三章則詳細說明問題定義及特性，並設計兩套禁忌搜尋法來求解；第四章同樣是應用禁忌搜尋法來求

解，針對 JNWF-SCM 排程問題的特性，在起始解的產生與搜尋鄰近解這兩部分作改良；第五章與第六章分別為模擬與實驗分析，以及結論與未來研究方向。

第二章　文獻回顧

No-wait 排程問題已經被廣泛的討論與研究，過去研究提出很多的演算法來解決此問題，目標希望以更短時間內求出近似最佳解，故「解的品質」與「求解時間」將決定該演算法好壞的重要指標。

本研究目的是發展一套演算法來解決 JNWF-SCM 排程問題。一方面 NWF 問題是屬於 JNWF-SCM 問題的特例，同樣含有 No-wait 限制；另一方面 JNWF-SCM 問題具有跳躍式(jumping)特性，類似 Jobshop 問題中不固定加工流程特性。因此本章針對 No-Wait Flowshop、No-Wait Jobshop 二大方向來做回顧。

2.1　探討 No-Wait Flowshop 相關文獻

根據 No-Wait Flowshop 特性，可再細分為下列三種情境：

- 傳統 NWF 性質的 Pure No-Wait Flowshop
- 跳躍式 NWF 性質的 Jumping No-Wait Flowshop
- 分流式 NWF 性質的 No-Wait Flowshop with Missing Operations

2.1.1　Pure No-Wait Flowshop 相關文獻

Gilmore and Gormory[9]將二部機台的NWF排程問題轉換成旅行者銷售網路問題(Traveling Salesman Problem, TSP)，將工件視為旅行者必須經過的城市，兩兩工件之間的投料間隔時間視為兩兩城市間的距離，目標值最小化總排程時間則對應為最小化旅行者走完所有城市的時間。Rock[16]的研究證明 NWF 排程問題在二部機台以上之問題其複雜度為 NP-hard。

許多文獻[1,2,7,11,12,13,14,15]同以最短總排程時間(makespan)為目標，提出一些啟發式演算法，並求得各工件的投料優序。

Palmer[13]的研究方法概念，是希望前段機台作業時間較短的工件可先投料。所以給予各機台作業一個固定權值，前面機台的作業時間加權值較小，後面

機台的作業時間加權值較大，依此計算出各工件總加權作業時間，其總加權值由大到小排列，由此得出工件的加工優序。

Campbell 等人[3]運用 Johnson's Rule 將多機台的排程問題簡化成二階段機台的問題。作法是將考慮的機台群分隔成兩階段，第一次只考慮第一部機台與最後一部機台，將此二機台視為兩階段，用 Johnson's Rule 做工件加工優序的排序；第二次則以第一部機台和第二部機台為前階段，最後一部機台和倒數第二部機台為後階段，利用 Johnson's Rule 做工件加工優序的排序；依此類推得到諸多組排序及其對應目標值後，再選出目標值最優的一組作為此方法的解。

Reddi 等人[15]與 Goyal[11]是將三部機台以上之 NWF 排程問題轉換成 TSP 網路問題。只是 Reddi[15]從第一部機台來考慮總排程時間，因此以任兩工件在第一部機台上的機台閒置時間做為 TSP 問題中任兩城市的距離；而 Goyal[11]是從最後一部機台來考慮總排程時間，以兩兩工件在最後一部機台上的機台閒置時間做為 TSP 問題中兩兩城市的距離。同樣是藉由減少 TSP 問題的總途程來達到最小化總排程時間的目的。

Bonney and Gundry[2]提出的方法則是希望相鄰工件有較少的機台閒置時間，運用工件在甘特圖上的表現，從中特性找出適合的兩相鄰工件。首先計算得出各工件的在各機台的開始加工時間回歸線斜率及結束加工時間回歸線斜率，如圖 2.1 所示，為了各機台不要閒置太久，所以將開始加工時間回歸線斜率值最小(最陡峭)的工件排第一個，為了縮短下個工件加工前的機台閒置時間，故從未排序的所有工件中，找出結束加工時間回歸線斜率最接近第一工件的開始加工時間回歸線斜率，作為排序中的第二個工件，依此類推找出各工件的加工優序。

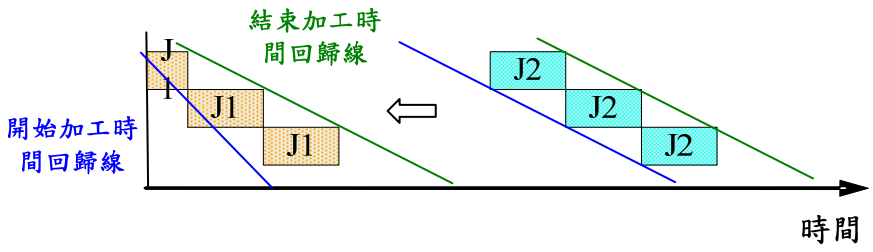

圖 2.1　工件開始加工時間與結束加工時間回歸線斜率之示意圖

Nawaz 等人[12]則是發展一套插入法則演算法來解問題。藉由工件總作業時間大者有排序的優先權，首先取前兩個優先權的工件並交互排列成 $J_1 \rightarrow J_2$ 與 $J_2 \rightarrow J_1$，比較此兩種排序的總排程時間，並選取總排程時間較小的工件排序，保持前兩工件的相對排序下，再將優先權第三的工件 J_3 插入前述的工件排序。也就是將 J_3 插入「□ J_2 □ J_1 □」中的位置「□」，若插入點有較小的總排程時間，則為 J_3 最適的插入點，依此類推，將所有工件依優先權先後插入排序，找出此方法的工件排序。

Gangadharan 等人[7]考慮到總排程時間等於所有工件在最後一部機台上的加工時間與機台閒置，因此將第一部到最後一部機台的加工時間為遞增的工件排在序列的前半部，而遞增特性愈明顯的工件排愈前面，加工時間為遞減的工件則排在序列的後半部，遞減特性愈明顯的工件排愈後面，如圖 2.2。

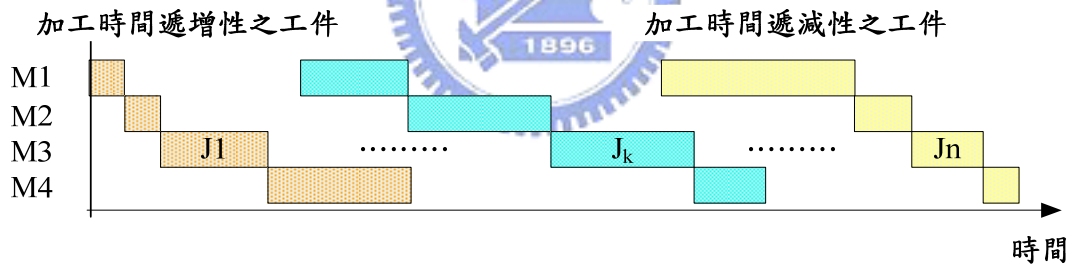

圖 2.2　加工時間遞增與遞減性質之工件

Rajendran [14]則是將工件以機台加工時間特性分為快速遞增、漸慢遞增、漸慢遞減、快速遞減等四個群組特性，如圖 2.3，工件藉由此四個特性依序排列形成一個起始序列，再依序從第一個工件開始做序列中各位置的插入，若插入點有較小的總排程時間，則為最適的插入點。此方法可以讓第一個工件的後段機台閒置時間較短，最後一個工件的前段機台閒置時間也能較短，以達到總排程時間短的目的。

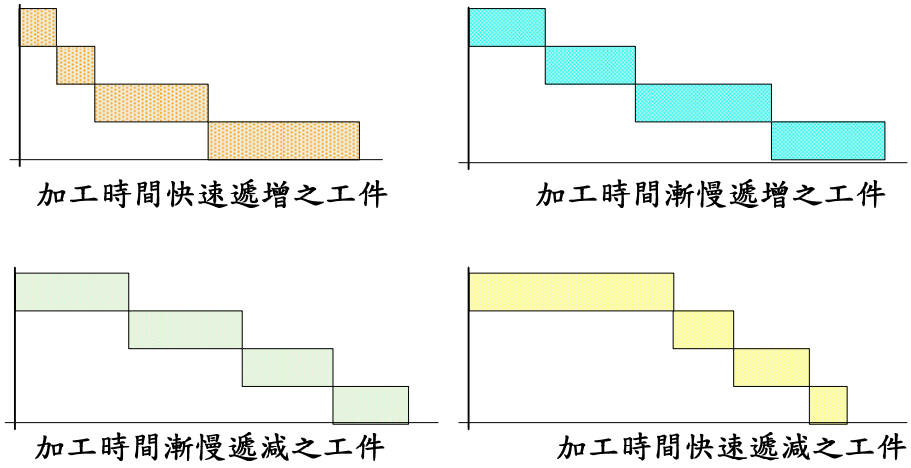

圖 2.3　工件之加工時間特性示意圖

Bertolissi[1]研究目標是最小化總流程時間總和(sum of the total flow time)，提出一個新的啟發式演算法來解決 NWF 的排程的問題。演算方法是先取兩兩工件並交互排列成 $J_i \rightarrow J_j$ 與 $J_j \rightarrow J_i$，比較此兩種排序的總排程時間，選取值較小的工件排序，若 $J_i \rightarrow J_j$ 排序較佳，則 J_i 計點贏一次。贏的次數較高的工件有排序的優先權，首先取次數最高的第一名與第二名工件並排列成 $J_1 \rightarrow J_2$，再將次數第三名的工件 J_3 插入前述的工件排序。也就是將 J_3 插入「□ J_1 □ J_2 □」中的位置「□」，若插入點有較小的總排程時間，則為 J_3 最適的插入點，依此類推，將所有工件依優先權先後插入排序，找出此方法的工件排序。

大多數啟發式演算法是先在短時間內獲得一個近似解，再利用交換法或插入法等進行近似解之改善，然而，此類方法的運算邏輯通常為確定性，並沒有跳出局部解的機制，容易在尋優過程中收斂在局部最佳解(local optimum)而停止，因此無法找到最佳解。之後，許多學者結合人工智慧(artificial intelligence, AI)設計出更有彈性的巨集式演算法(metaheuristic algorithm)來解 NWF 排程問題。Chen 等人[4]應用基因演算法(Genetic Algorithm，GA)來求 NWF 排程問題，Shyu 等人[19]以最小化總完工時間為目標，套入螞蟻演算法(Ant Colony Optimization，ACO)來處理二部機台的 NWF 排程問題，Fink 等人[5]則是應用禁忌搜尋法(Tabu Search,

TS)來處裡多部機台的 NWF 排程問題。

2.1.2 Jumping No-Wait Flowshop 相關文獻

NWF-J 排程問題是一個比較新的研究課題，其主要特性除了遵守 NWF 限制之外，還添加了 Jumping 的特性，也就是說工件在生產線上會有跳機台的現象發生。NWF-J 情境的簡單定義：NWF-J 生產線中，工件經過機台的先後順序一樣，但會因產品的種類不同，加工途程也會不同，工件可能會跳過(jumping)某些機台來加工，導致工件不需要在所有機台加工。

對於 NWF-J 生產線排程問題過去的文獻中只有李東森[20]做出相關研究，其探討問題是在 NWF-J 生產線下，以最小化總排程時間為目標，並設計出一套有效率的禁忌搜尋法來求解。作者考慮到排程序列中，相鄰兩工件的完工延遲時間(delay time)差愈小時，所造成的機台延遲時間也愈小，如圖 2.4 所示，也就能藉此來減少總排程時間。

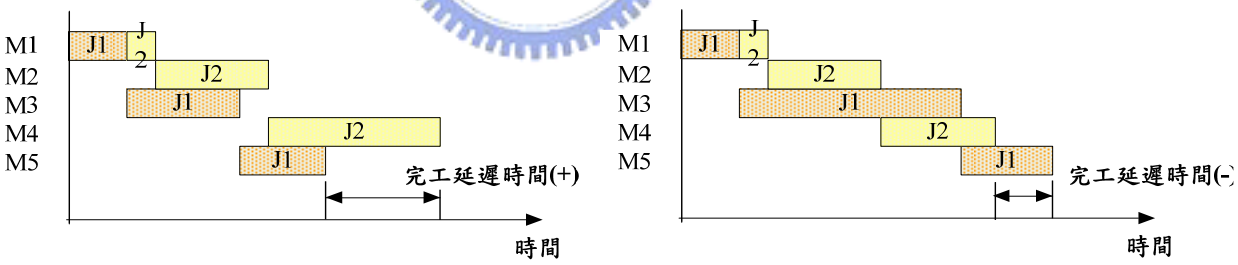

圖 2.4　相鄰工件的完工延遲時間

首先計算得出兩兩工件排序的完工相鄰時間，並以最小總加工時間最小的工件作為第一排序工件，隨之，找與第一工件完工相鄰時間最接近的工件最為第二排序工件，依此類推排完所有工件得到一個起始解。改善方向考慮到起始解序列中，兩相鄰工件的投料時間差可能甚大，藉由拆開此相鄰工件來改善，另一方面，由於 NWF-J 特性，相鄰兩工件的共用作業機台數愈少，則可以提高機台利用率，

縮小總排程時間，故藉由相鄰工件共用作業機台數愈少愈好之觀念來改善。以此上述從兩觀點來對起始解找不同的鄰近解，並使用禁忌搜尋法求解。

2.1.3 No-Wait Flowshop with Missing Operations 相關文獻

藉由 Sahni and Cho[17]的證明，已知 NWF-MO 問題是 NP-hard 問題。Glass 等人[10]針對二部機台的 NWF-MO 排程問題，以最小化排程時間為目標，提出一套演算法。其研究情境假設工件一共分為兩種，第一種是經過兩部機台加工，第二種是只經過第一部機台加工，並跳過第二部機台。演算法主要概念分為兩階段：第一階段是先將第一種工件以 Gilmore and Gormory[9]的 TSP 方法求得出一組序列，並且計算出此序列上相鄰工件在第一部機台的閒置時間，如圖 2.5；第二階段則是類似裝箱問題(bin packing problems)，希望能將第二種工件把第一部機台閒置時間給填滿。作法是先以第二種工件的加工時間，由大到小決定排序的優先權，從優先權順位第一的工件開始插入至第一階段的結果。

例如，第一階段結果序列為 $J_1 \rightarrow J_2 \rightarrow J_3 \rightarrow J_4$，將優先權順位第一的工件 J_{first} 插入至「$J_1 \square J_2 \square J_3 \square J_4 \square$」中的位置「□」，若插入點位置的第一部機台閒置時間最接近工件 J_{first} 的加工時間，則為工件 J_{first} 最適的插入點，依此類推，將所有工件依優先權先後插入排序，找出此方法的工件排序。

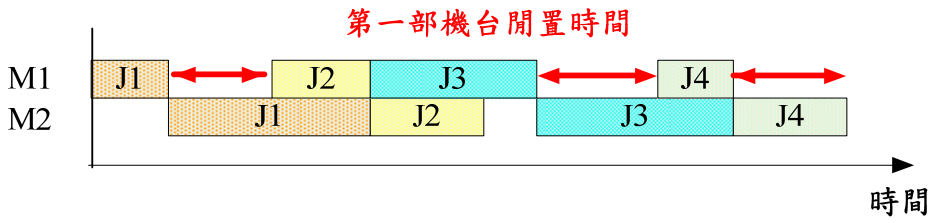

圖 2.5　第一部機台閒置時間示意圖

2.2 探討 No-Wait Jobshop 相關文獻

Sahni and Cho[17]已證明二機台以上的 Jobshop 排程問題是屬 NP-hard。No-Wait Jobshop 排程問題中，有些學者使用巨集式演算法來解決，其中包含了禁忌搜尋法、模擬退火法。後來，Schustera and Framinan[18]運用鄰近搜尋法(Variable Neighborhood Search algorithm ,VNS)以及 GASA 來求 No-Wait Jobshop 排程最小化總排程時間問題。

Framinan and Schustera [6]研究 m 部機台的 No-Wait Jobshop 排程問題，以最小化總排程時間為目標。為了縮小機台的閒置時間，提出 Inverse Timetabling 方式來計算總排程時間，並且運用 Complete Local Search with Memory(Ghosh and Sierksma[8])演算法來求解，此方法與禁忌搜尋法類似，但相異處是此方法[6]以長期記憶來搜尋較好的解。

上述 No-Wait Flowshop 與 No-Wait Jobshop 相關文獻彙總如表 2.1。

表 2.1 相關文獻彙總表

作者	情境	目標	方法	究結果與貢獻
Gilmore and Gormory[9]	二部機台 NWF	最小化總排程時間	換成 TSP 網路問題	最先把問題架構成 TSP 模型
Palmer[13]	多部機台 NWF	最小化總排程時間	將前端作業時間短的工件排序較前面	以工件作業時間大小的特性來做排序考量
Campbell 等人[3]	多部機台 NWF	最小化總排程時間	Johnson's Rule	解品質表現比 Palmer[13]佳
Reddi 等人[15]和 Goyal [11]	多部機台 NWF	最小化總排程時間	TSP 網路問題	第一個探討多部機台 NWF 排程問題，且以 TSP 模型架構求解
Bonney and Gundry[2]	多部機台 NWF	最小化總排程時間	用工件時間回歸線斜率的特性來排序	將 m 部機台的 NWF 問題簡化為二部機台

Nawaz 等人[12]	多部機台 NWF	最小化總排程時間	提出一套插入法來改善原先的排序	解品質表現比 Campbell 等人[3]佳
Gangadharan 等人[7]	多部機台 NWF	最小化總排程時間	以工件加工時間有遞增遞減特性來排序	處理較大型問題時，解的品質較差
Rajendran [14]	多部機台 NWF	最小化總排程時間	綜合工件加工時間遞增遞減特性以及插入法來排序	演算法的求解時間稍長，穩定性也不足
Bertolissi[1]	多部機台 NWF	最小化總流程時間總和	兩兩工件為一組排序，排在前方的工件次數最多的優先排列	解品質表現比 Bonney and Gundry[2]佳
李東森[20]	多部機台 NWF-J	最小化總排程時間	從問題特性找較好的相鄰解並用禁忌搜尋法求解	第一個探討 NWF-J 排程問題
Glass 等人[10]	二部機台 NWF-MO	最小化總排程時間	運用 Gilmore and Gormory[9]的方法與裝箱問題解法來尋優	第一個探討 NWF-MO 排程問題，證明此方法的 worst case performance ratio 為 4/3
Schustera and Framinan[18]	多部機台 No-Wait Jobshop	最小化總排程時間	鄰近搜尋法以及 GASA	小型問題結果以 GASA 比較佳，大型問題則以 VNS 較佳
Framinan and Schuster [6]	多部機台 No-Wait Jobshop	最小化總排程時間	運用 Complete Local Search with Memory 與 Inverse Timetabling	解品質表現比 Schustera and Framinan[18]佳

2.3 本研究與過去不同處

本研究針對多機台 JNWF-SCM 排程問題，設計一套有效率的演算法來求解，多機台的 JNWF-SCM 生產線中，各機台對工件的加工順序並不盡相同，大幅提高問題的複雜度，因此，致力於發展一套有效率的演算法來求解。過去文獻

中有 Glass 等人[10]研究此類型相關問題，然而問題的規模只限制於兩部機台，故不適用於多機台問題。而相較於李東森[20]探討的多機台的 NWF-J 排程問題，其情境是比本研究的 JNWF-SCM 更具為一般性，因此本研究提出之演算法將會與李東森[20]的演算法作為比較。

第三章　問題描述與禁忌搜尋方法

3.1　問題定義

本研究探討的 JNWF-SCM 排程問題定義如下：一個具有 m 部機台的 flowshop 生產線，另有 n 個獨立且符合 no-wait 加工特性的工件，每一個工件的加工途程皆相同，加工途程的第一站亦即第一部機台，當工件經過第一站加工之後，從第二部機台開始，則會依照工件種類，可能跳過部份機台的加工，但至少經過某一站機台來加工，以總排程時間的最小化為目標，決定 n 項工件的投料時間。故本問題不僅是一般的 No-Wait Flowshop 排程問題，還結合了 Jumping 與 Starting Common Machine 的特性。

本研究問題的其他假設條件在 1.4 節中已詳細說明，解的表達方式將以一組工件序列來表示，此序列代表工件佔用第一部機台的順序，以下稱之為「工件優先序列」，並由演算法執行得出，藉由此序列以及各工件加工時間資訊，依序求得各工件的投料時間，進而求得總排程時間。詳細的方法與步驟在 3.3 節與 3.4 節中做說明。

3.2　符號定義

指標符號：

k　：　演算過程中，迭代次數的指標變數

i, j　：　工件的指標變數

h　：　機台的指標變數

變數定義：

n　：　工件總數

m　：　機台總數

$J_{\sigma_k(i)}$　：　演算過程第 k 次迭代的序列中，第 i 個工件

$J\sigma_k$ ： $J\sigma_k = \{J_{\sigma_k(1)}, J_{\sigma_k(2)}, ..., J_{\sigma_k(i)}, ..., J_{\sigma_k(n-1)}, J_{\sigma_k(n)}\}$：演算法中第 k 次迭代得到的解，表示工件佔用第一部機台的優先序列，依序為 $J_{\sigma_k(1)}, J_{\sigma_k(2)}, ..., J_{\sigma_k(i)}, ..., J_{\sigma_k(n-1)}, J_{\sigma_k(n)}$，簡稱為工件優先序列。

P_j^h ： 工件 j 在機台 h 的加工時間。$(j = 1,2,...,...,n \ ; \ h = 1,2,...,...,m)$

T_j ： 工件 j 的總加工時間，$T_j = \sum_{h=1}^{m} P_j^h$ 。$(j = 1,2,...,...,n)$

D_{ij} ： 工件優先序列 $i \rightarrow j$，在第一部機台的閒置時間。$(i, j \in J \ ; \ i \neq j)$

S_{ij} ： 工件優先序列 $i \rightarrow j$ 的完工延遲時間。$(i, j \in J \ ; \ i \neq j)$

E_{ij} ： 工件優先序列 $i \rightarrow j$，在第一部機台的閒置時間加上完工延遲時間，其中 $E_{ij} = D_{ij} + S_{ij}$ 。$(i, j \in J \ ; \ i \neq j)$

$ST_{\sigma_k}^{J_{\sigma_k(i)}}$ ： 第 k 次迭代的解中，第 i 個工件的開始加工時間。$(j = 1,2,...,...,n)$

$FT_{\sigma_k}^{J_{\sigma_k(i)}}$ ： 第 k 次迭代的解中，第 i 個工件的完成加工時間。$(j = 1,2,...,...,n)$

3.3 問題特性

傳統的 NWF 排程方法，是以工件的投料順序作為求解的準則，依照投料序將工件一個個往後投料，所以從排程結果可看出，各機台的加工順序與投料序是相同的，如圖 3.1 所示。然而，工件在 JNWF-SCM 情境下，並非在所有機台上加工，此情境的排程結果，投料順序與各機台加工順序可能會發生不一至的狀況，後投料的工件可能在某機台上，比先投料的工件還早加工，如圖 3.2 所示，因此，JNWF-SCM 排程問題並不適合傳統的排程方法，故 JNWF-SCM 排程問題比傳統 NWF 排程問題有較高的複雜度。

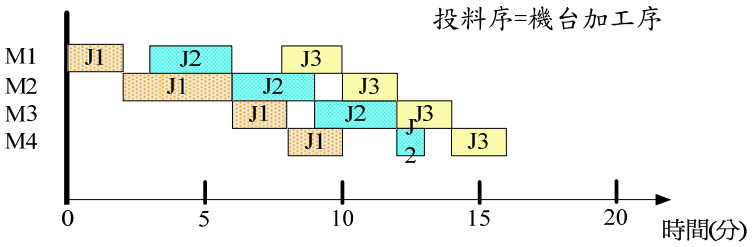

圖 3.1　投料序與機台加工序示意圖(1)

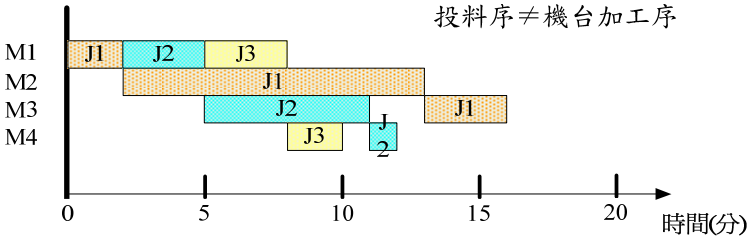

圖 3.2　投料序與機台加工序示意圖(2)

表 3.1 為 JNWF-SCM 排程問題例子，若以傳統方法給定投料順序 $J\sigma=\{J_1,J_2,J_3,J_4\}$，從排程結果圖 3.3 顯示出，各機台上有太多閒置時間，機台使用率偏低，總排程時間達到 19 分鐘，傳統方法的結果顯然效率不高。

表 3.1　JNWF-SCM 的工件加工時間表(2)

加工 機台 / 時間(分) / 工件	M_1	M_2	M_3	M_4	M_5
J_1	1	-	5	-	3
J_2	1	-	2	4	2
J_3	3	2	-	1	0
J_4	2	4	2	-	1

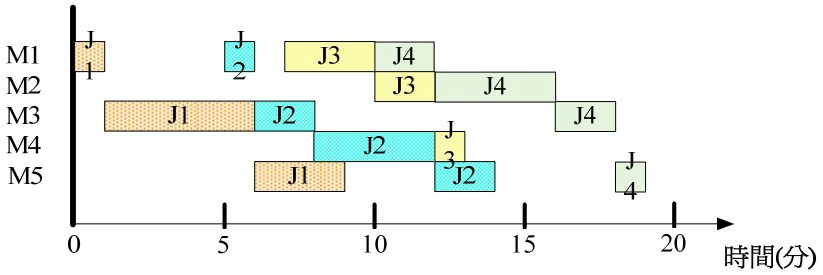

圖 3.3　依投料序排程結果示意圖

故本研究考慮 JNWF-SCM 工件跳機台的特性，使用另一種排程方法。作法為：每一個欲排入的工件，事先檢視各機台的閒置時間，是否能夠將閒置時間充分利用，並從中找尋工件能最早的投料時間。因此這樣的方法下，工件的排入順序如同工件佔用機台的優先順序，故本研究將此順序定義為「佔用第一部機台的順序」，此序列又稱「工件優先序列」。同樣以表 3.1 為例，給定工件優先序列為 $J\sigma=\{J_1,J_2,J_3,J_4\}$，各工件排入的說明如下：

工件 J_1 為序列中第一個佔用第一部機台的工件，所以 $ST_\sigma^{J_1}=0$，如圖 3.4 所示，圖中的線段代表各機台目前的閒置時間。

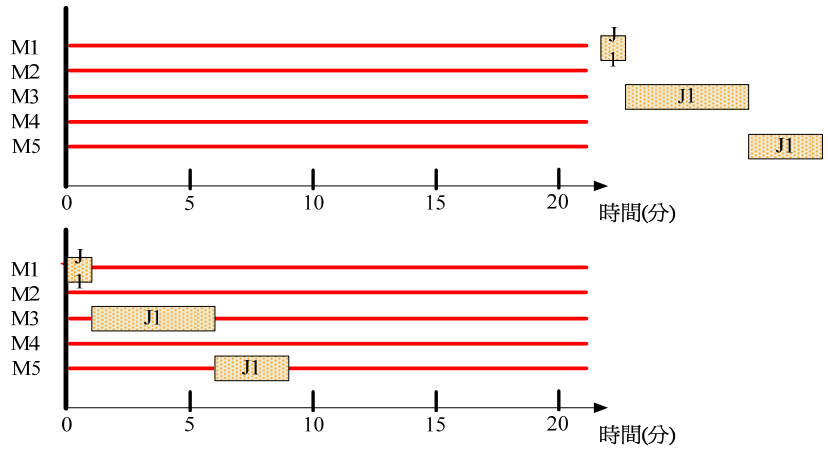

圖 3.4　$J\sigma$ 之 JNWF-SCM 加工甘特圖(1)

接著排序工件 J_2，由於工件 J_1 已排定的情況下，工件 J_2 的最早開始加工時

間為第 5 分鐘，即 $ST_{\sigma}^{J_2}=5$，如圖 3.5 所示。

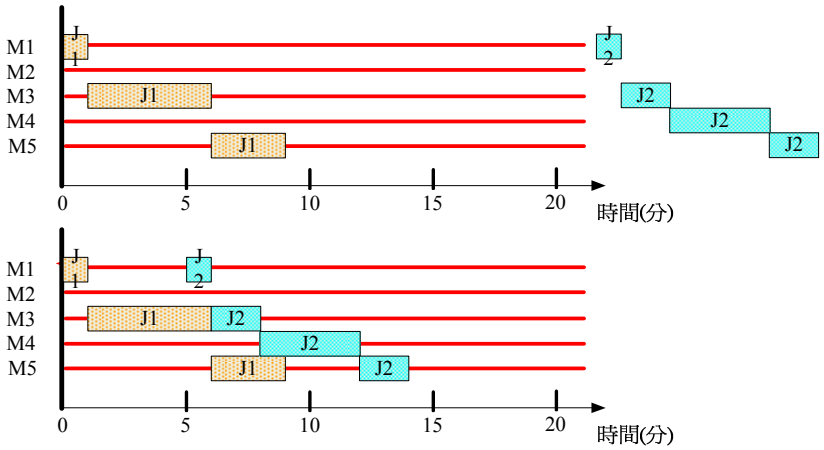

圖 3.5　$J\sigma$ 之 JNWF-SCM 加工甘特圖(2)

再者，排序工件 J_3，由於工件 J_1 與 J_2 已排定的情況下，工件 J_3 的最早開始加工時間為第 7 分鐘，即 $ST_{\sigma}^{J_3}=7$，如圖 3.6 所示。

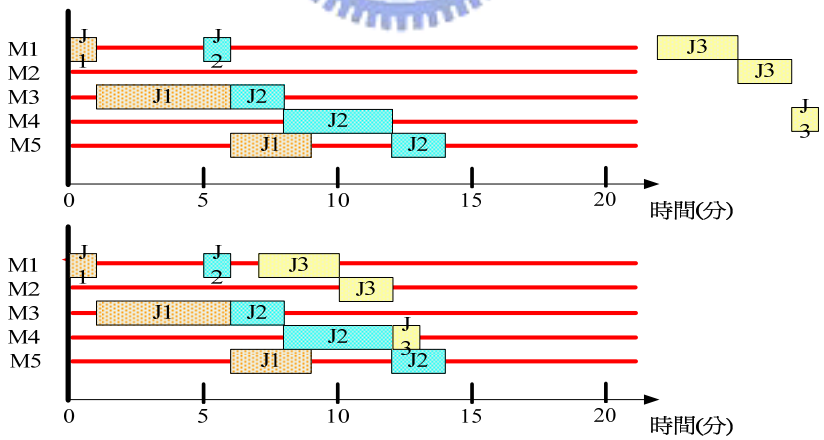

圖 3.6　$J\sigma$ 之 JNWF-SCM 加工甘特圖(3)

最後，排序工件 J_4，由於工件 J_1、J_2 與 J_3 已排定的情況下，檢視機台 M_1 在時間區間分鐘閒置，機台 M_2 在時間區間分鐘閒置，機台 M_3 在時間區間分鐘閒置，因此工件 J_4 可以插入此時段內來加工，故工件 J_4 的最早開始加工時間為第 2

分鐘，即 $ST_{\sigma}^{J_4}=2$ ，如圖 3.7 所示。

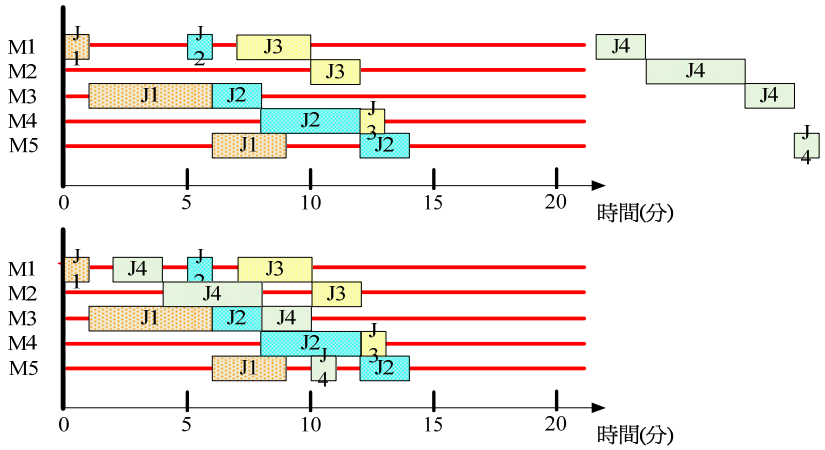

圖 3.7　$J\sigma$ 之 JNWF-SCM 加工甘特圖(4)

四個工件皆排入之後，新方法排程結果如圖 3.7，倘若以投料序的觀點來看，此排程結果的投料序是 $J\sigma=\{J_1,J_4,J_2,J_3\}$，然而若以此投料序做排程，結果如圖 3.8 所示。圖 3.7 與圖 3.8 之投料序雖然相同，但排程結果卻不一樣，這是因為 JNWF-SCM 的特性導致，故 JNWF-SCM 排程問題以新排程方法來執行。

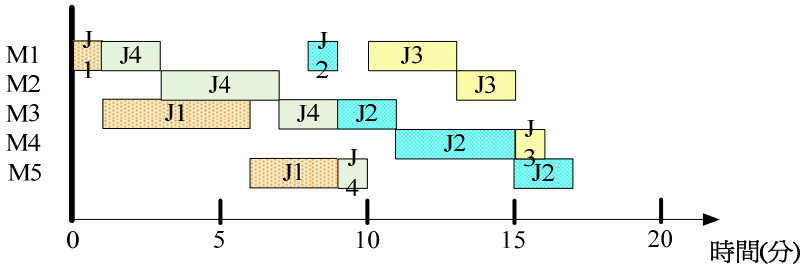

圖 3.8　$J\sigma$ 之 JNWF-SCM 加工甘特圖(5)

此新方法與傳統方法最大不同處，在於工件能夠找尋機台可利用的閒置時間，並作插入投料以善用閒置時間。本研究藉由此方法來排程，使得機台閒置時間少，使用率更高，的確適合 JNWF-SCM 排程問題的加工特性。

3.4 計算總排程時間的方法

傳統排程方法的總排程時間計算，依投料序中最後投料工件的完工時間，為該排程結果的總排程時間。以圖 3.7 為例，排程結果中最後投料的工件是 J_3，因此傳統方法上，則以 J_3 的完工時間作為總排程時間，也就是 13 分鐘，但從圖明顯看出此例的結果並非 13 分鐘。因此，新排程方法的總排程時間計算，是將排程結果中，取所有工件完工時間最大值，作為總排程時間，圖 3.7 例子中最大的完工時間為 $FT_{\sigma}^{J_3}=14$，總排程時間亦為 14 分鐘。

藉由上一節 JNWF-SCM 排程問題的特性，並以工件優先序列為解的表達方式，JNWF-SCM 排程問題的總排程時間計算說明與流程圖如下。

Step1：已知各工件加工時間資訊以及工件優先序列 $J\sigma=\{J_{\sigma(1)},J_{\sigma(2)},...,J_{\sigma(i)},J_{\sigma(n)}\}$

Step2：依序將工件排定佔用機台，找出工件 $J_{\sigma(i)}$ 能最早開始加工的時間。欲排入之工件，必須檢視目前各機台上的閒置時間，以期盡早投料。

Step3：得出各工件的加工開始時間 $ST_{\sigma}^{J_{\sigma(i)}}\ (i=1,2...,n)$，以及各工件的完工時間 $FT_{\sigma}^{J_{\sigma(i)}}\ (i=1,2...,n)$，

Step4：所有工件完工時間值最大的，即為總排程時間。

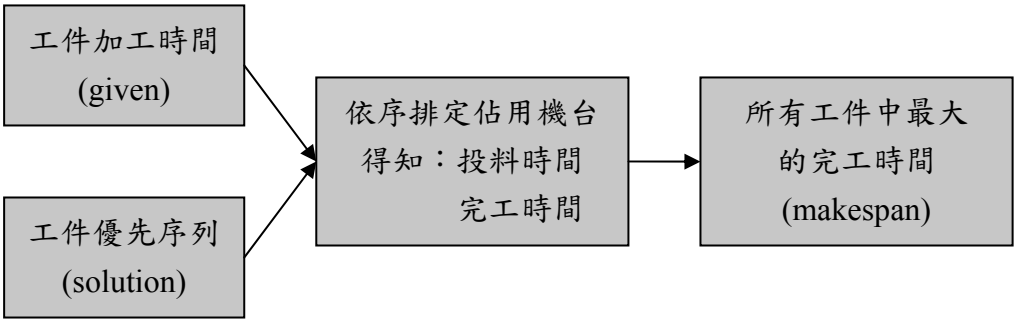

圖 3.9　總排程時間計算流程圖

3.5 禁忌搜尋法

本研究首先設計兩套禁忌搜尋法(EXTS1、EXTS2)，來求解 JNWF-SCM 排程問題，以下五個步驟說明求解過程：

(1) 起始解之產生

(2) 鄰近解之產生方法

(3) 禁忌列表

(4) 終止條件

3.5.1 起始解之產生

從文獻中探討 NWF 排程問題的兩個演算法 Bertolissi[1]與 Rajendran[14]，分別做為本研究 EXTS1 與 EXTS2 演算法的起始解。然而 Bertolissi[1]所提出的方法，原本是以最小化總流程時間總和(sum of the total flow times)為目標，為了符合本研究的研究目標，故將此演算法修改為以總排程時間為目標，作為 EXTS1 的起始解。此兩者的原演算方法於附錄一說明。

3.5.2 鄰近解之產生方法

禁忌搜尋法在每次迭代的過程中，都會搜尋數個現行解的鄰近解，作為移步的候選名單。過去 NWF 排程問題研究方法中，較常使用的三種鄰近解方法為交換法(exchange)、插入法(insertion)與倒反法(invertion)。故本研究的 EXTS1 與 EXTS2 演算法的鄰近解則以上述方法為基礎，延伸出一共五種鄰近解搜尋方法，分別為：(a)任意兩點交換法，(b)任意三點交換法，(c)任意四點交換法，(d)插入式變動法，(e)兩點間倒反法。此五種鄰近解搜尋方法說明如下：

(a) 任意兩點交換法

任意兩點交換法與之後的任意三點交換法和任意四點交換法，又稱做交換式變動法(Swap move)，其方法為任意選取現行解的兩個工件，並將此兩工件交

換其優先順序來產生新的鄰近解。

舉例說明：假設原優先序列為$\{J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8\}$，隨機選取兩個工件$J_4$與$J_7$，則$J_4$與$J_7$交換，產生新的優先序列為$\{J_1, J_2, J_3, J_7, J_5, J_6, J_7, J_4\}$，示意圖 3.10 如下：

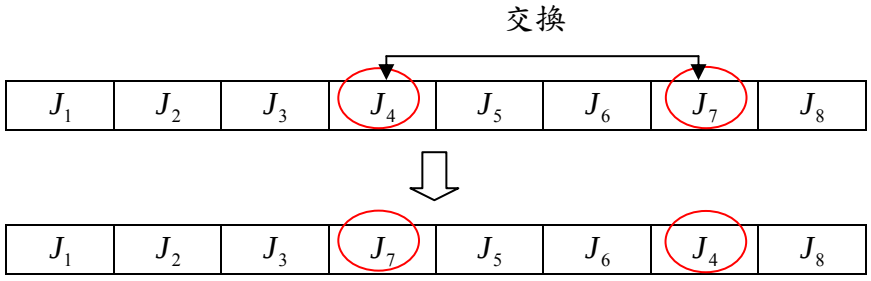

圖 3.10　任意兩點交換法移步後示意圖

(b) 任意三點交換法

任意三點交換法為隨機選取三個工件，並互相交換其優先順序來產生新的鄰近解。

舉例說明：假設原優先序列為$\{J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8\}$，隨機選取三個工件$J_1$、$J_3$與$J_7$，則$J_1$移至$J_3$的排序位置，再將$J_3$移至$J_7$的排序位置，最後將$J_7$移至$J_1$原本的排序位置，產生新的優先序列為$\{J_7, J_2, J_1, J_4, J_5, J_6, J_3, J_8\}$，示意圖 3.11 如下：

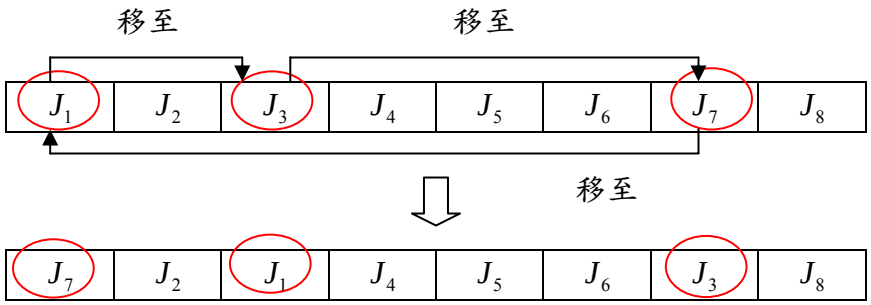

圖 3.11　任意三點交換法移步後示意圖

(c) 任意四點交換法

任意四點交換法為隨機選取四個工件，並互相交換其優先順序來產生新的鄰近解。

假設原優先序列為$\{J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8\}$，隨機選取四個工件$J_1$、$J_3$、$J_6$與$J_8$，則$J_1$移至$J_3$的排序位置，將$J_3$移至$J_6$的排序位置，再將$J_6$移至$J_8$的排序位置，最後將$J_8$移至$J_1$原本的排序位置，產生新的優先序列為$\{J_8, J_2, J_1, J_4, J_5, J_3, J_7, J_6\}$，示意圖 3.12 如下：

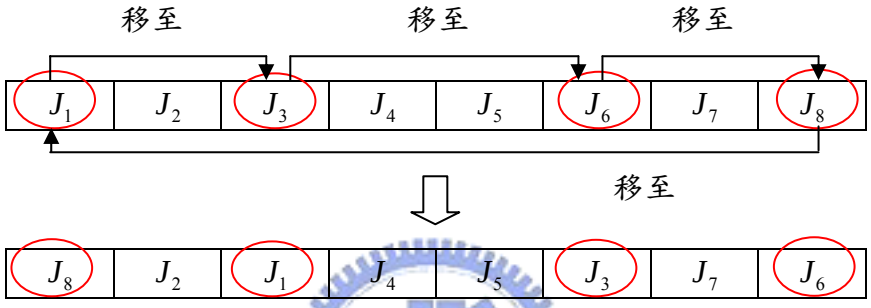

圖 3.12　任意四點交換法移步後示意圖

(d) 插入式變動法

插入式變動是從現行解中隨機選取二個不同的工件，並將此兩工件中將排序較後面的工件移至排序較前的工件之前，得到一個新的鄰近解。

舉例說明：假設原優先序列為$\{J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8\}$，隨機選取兩個工件$J_2$與$J_8$，接著將排序後面的$J_8$移至$J_2$前面，產生新的優先序列為$\{J_1, J_8, J_2, J_3, J_4, J_5, J_6, J_7\}$，示意圖 3.13 如下：

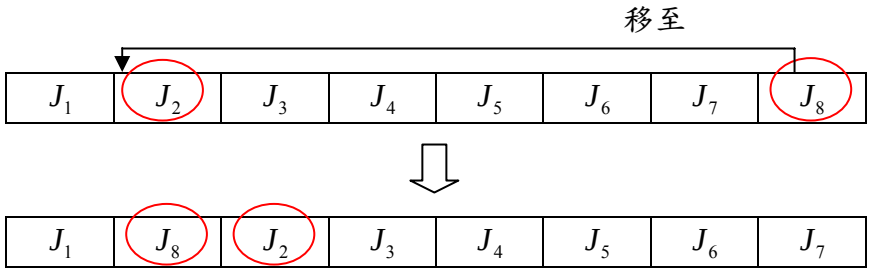

圖 3.13 插入式變動法移步後示意圖

(e) 兩點間倒反法

選取任意兩個工件，將這兩工件之間的所有工件，優序顛倒產生新的鄰近解。

舉例說明：假設原優先序列為$\{J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8\}$，隨機選取兩個工件$J_2$與$J_8$，則$J_2$與$J_8$兩工件之間的所有工件優序顛倒，產生新的優先序列為$\{J_1, J_8, J_7, J_6, J_5, J_4, J_3, J_2\}$，示意圖 3.14 如下：

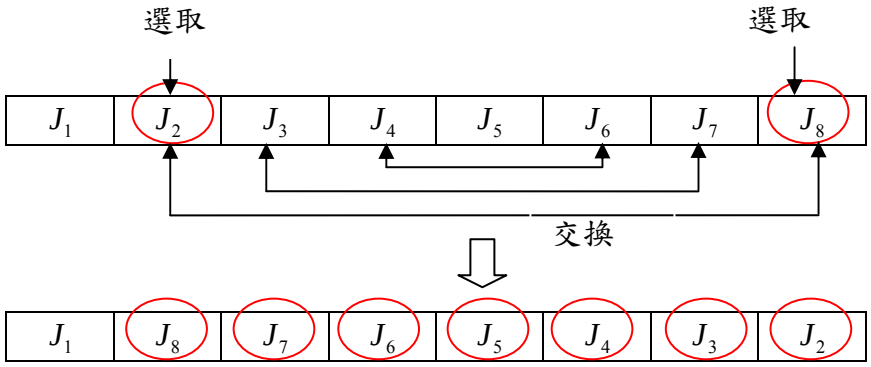

圖 3.14 兩點間工作互換法示意圖

3.5.3 禁忌列表

本研究設定禁忌列表的大小為三倍總工件數，且更新的原則是採取先進先出法則(FIFO)，將每次迭代中將取代現行解的最佳鄰紀錄在禁忌列表，紀錄的資訊是整個工件優先序列，當禁忌列表內的禁忌名單數到達上限時，則根據先進先出法則更新名單。

禁忌列表是用來紀錄最近多次迭代過程中所獲得的解，作為提供禁忌限制記憶機制，以避免重複搜尋造成的循環，每次迭代所產生的鄰近解，皆會檢視此鄰近解是否已存在於禁忌列表中，若已存在名單中則重新搜尋新的鄰近解。

3.5.4 終止條件

現行解所產生的最佳鄰近解中，即使沒有優於最佳解，此最佳鄰近解仍可取代現行解，沒有改善的次數則須累計一次，而當沒有改善的連續次數到達所設定的次數(epoch length)，則終止演算法，並以目前的最佳解為此次演算法的最終解。

下圖是禁忌搜尋法應用於 JNWF-SCM 排程問題的流程圖：

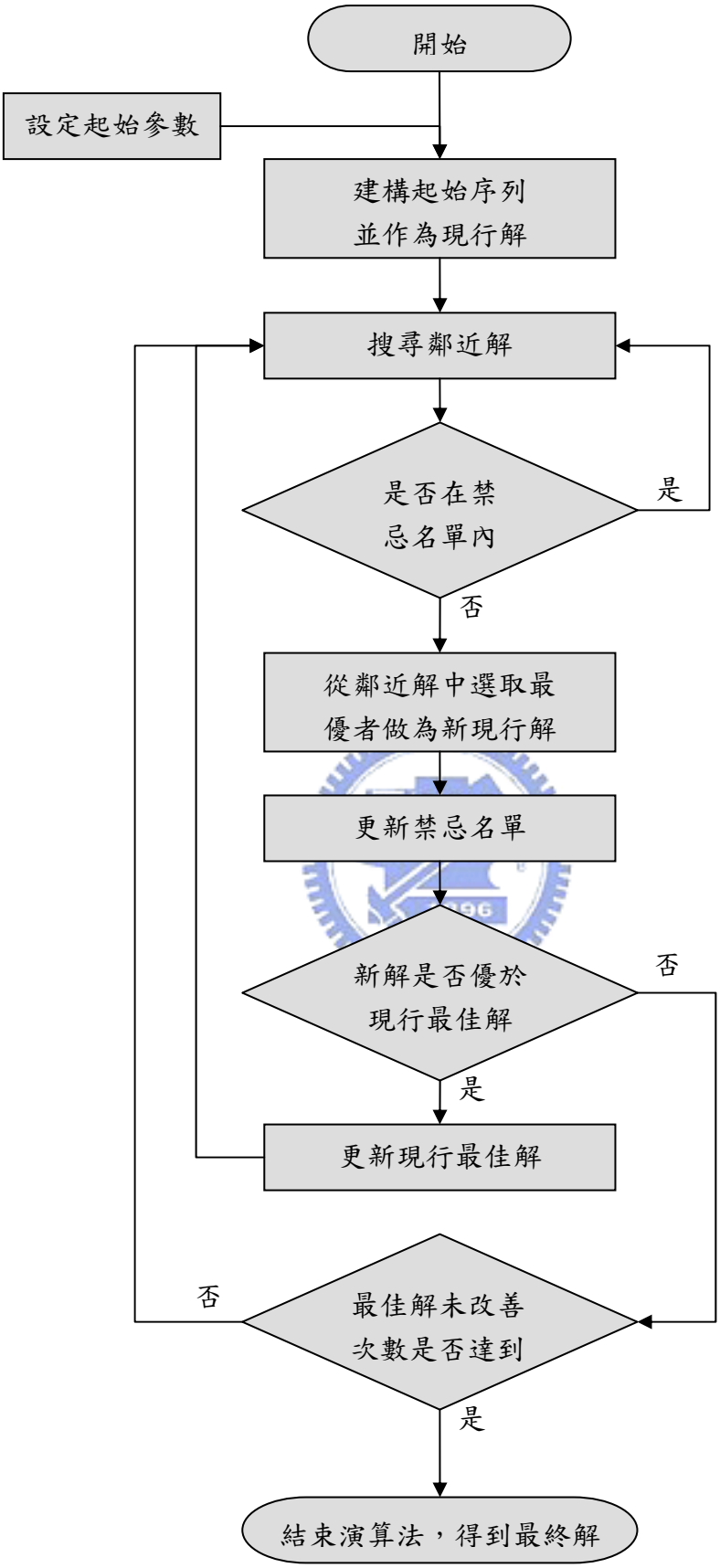

圖 3.15 禁忌搜尋法流程圖

第四章　改良式禁忌搜尋法

為了能更符合 JNWF-SCM 排程問題的特性，本章改良 EXTS1 與 EXTS2，設計一套新的禁忌搜尋法(簡稱 ITS)來求解 JNWF-SCM 排程問題。ITS 與 EXTS1、EXTS2 兩演算法相異的地方，在於起始解之產生與鄰近解產生的不同，本章將針對此兩部份做說明。

4.1　符號定義

除了第三章的符號與變數之外，本章增加了以下變數：

變數定義：

$\pi_{\sigma_k(i)}$　：　第 k 次迭代的解中，第一部機台上的第 i 段閒置時間。$(i=1,2,...,...,n-1)$

$\pi_{\sigma_k(i)} = ST^a_{\sigma_k} - FT^b_{\sigma_k}$，$a$ 與 b 代表排程中兩相鄰工件。

$t_{i,j}$　：　兩工件 i, j 之間的共用機台數。

$W_{i,j}$　：　兩工件 i, j 之間的共用機台數權重。

4.2　起始解之產生

ITS 演算法的起始解概念：

(1) 選取起始解的第一個工件 $J_{\sigma_0(1)}$：第一部機台加工時間最短的工件。

(2) 選取起始解第一個工件之後的工件 $J_{\sigma_0(i)}$，$i=2,3,...,n$：以雙目標加權準則選取最適合的工件。

以下做針對起始解的概念作詳細的說明：

本研究的 JNWF-SCM 排程問題，因所有工件必須經過第一部機台加工，第一個工件在第一部機台加工時，其他機台是處於閒置的狀態，為了減少這些機台的閒置時間，故選擇在第一部機台加工時間最短的工件，做為起始解工件優先序

列的第一個工件。如圖 4.1，工件 J_2 比工件 J_1 適合當第一個工件。

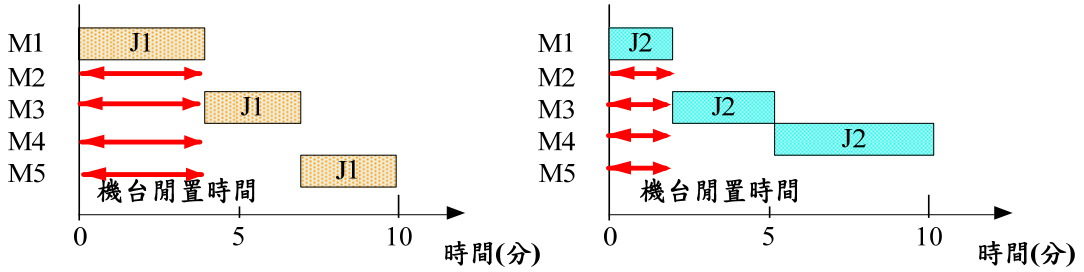

圖 4.1　起始工件的機台閒置時間

雙目標加權準則的第一個考量目標觀點，著重於第一部機台的閒置時間。因所有工件都要在第一部機台上加工，減少第一部機台的閒置時間是起始解演算法重要的方向。首先計算第一部機台的閒置時間，例如佔用機台 M_1 優序為 $J_1 \rightarrow J_2$，則在第一部機台造成的閒置時間為 2 分鐘，如圖 4.2。相反的，若佔用機台 M_1 優序改為 $J_2 \rightarrow J_1$，則在第一部機台造成的閒置時間為 0 分鐘，如圖 4.3。

雙目標加權準則的第二個考量目標觀點，著重於相鄰工件的完工延遲時間。為了最小化總排程時間，工件優先序列中每加入一個新的工件，都希望增加的完工時間能夠愈短愈好，藉由相鄰工件的完工延遲時間為考量目標，以期縮小總排程時間。如圖 4.2 中，佔用機台優序為 $J_1 \rightarrow J_2$，工件 J_1 的完工時間為第 9 分鐘，而工件 J_2 的完工時間為第 13 分鐘，因此工件 J_2 對於工件 J_1 造成多出來的完工延遲時間為 4 分鐘；同理，圖 4.3 中，工件 J_1 對於工件 J_2 造成多出來的完工延遲時間為 2 分鐘。

選取起始解的第二個與第二個之後的工件，將對以上兩目標做等權重的加權準則，以選取最適合的工件。

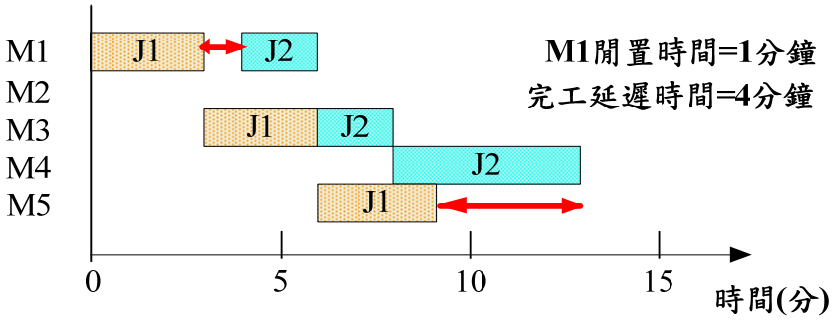

圖 4.2　相鄰工件示意圖(1)

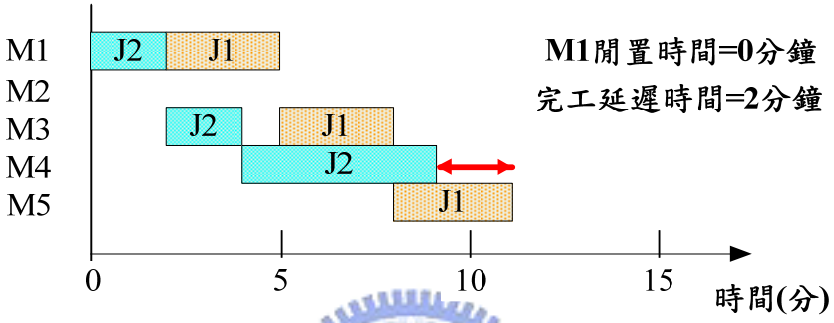

圖 4.3　相鄰工件示意圖(2)

藉由已知的工件加工時間資訊中，可以計算得知所有兩兩工件排序時，所產生的第一部機台閒置時間與工件完工延遲時間，因此從這些資訊將可求得起始解的工件優先序列。以下為 ITS 演算法的起始解產生步驟：

步驟一：找尋在第一部機台加工時間 P_j^1 最短的工件，作為序列的起始工件 $J_{\sigma_0(1)}$，若有相同加工時間的工件，則選取總加工時間 $T_j = \sum_{h=1}^{m} P_j^h$ 值最小的工件。

步驟二：計算工件 $J_{\sigma_0(1)}$ 與其他工件的 $D_{J_{\sigma_0(1)}j}$、$S_{J_{\sigma_0(1)}j}$，與 $E_{J_{\sigma_0(1)}j}$ ($E_{J_{\sigma(1)}j} = D_{J_{\sigma_0(1)}j} + S_{J_{\sigma_0(1)}j}$)。

步驟三：從未排序的工件中，找 $E_{J_{\sigma_0(1)}j}$ 最小值的工件，作為序列第二的工件 $J_{\sigma_0(2)}$。若有相同最小值 $E_{J_{\sigma_0(1)}j}$ 相同的，則隨機選一個工件。

步驟四：同步驟二與步驟三，從尚未排入序列的工件中，找尋序列的下一個工件，

直到所有的工件皆已排入起始解序列中則停止。

步驟五：起始解演算法結束。且總排程時間為最大的 $FT_{\sigma_0}^{J_{\sigma_0(i)}}$ 將代表此序列 $J\sigma_0$ 的總排程時間。

舉例說明，以下表 4.1 的工件為例：

表 4.1　JNWF-SCM 的工件加工時間表(3)

加工時間(分) 機台 / 工件	M_1	M_2	M_3	M_4	M_5	**Total**
J_1	1	-	8	-	2	**11**
J_2	2	-	5	7	3	**17**
J_3	2	8	-	3	-	**13**
J_4	2	-	2	-	5	**9**
J_5	2	2	1	-	6	**11**

表 4.2　工件在第一部機台閒置時間：D_{ij}

$i \rightarrow j$ 在 M_1 閒置時間(分) 工件 J / 工件 i	J_1	J_2	J_3	J_4	J_5
J_1	7	6	0	6	5
J_2	6	5	2	11	10
J_3	0	4	6	0	6
J_4	1	0	0	3	2
J_5	2	1	0	5	4

表 4.3　工件之完工延遲時間：S_{ij}

$i \rightarrow j$ 工件 j / 完工延遲時間(分) / 工件 i	J_1	J_2	J_3	J_4	J_5
J_1	8	13	3	5	6
J_2	2	7	0	5	6
J_3	0	10	8	.2	6
J_4	5	10	6	5	6
J_5	4	9	4	5	6

表 4.4　總和表：E_{ij}

$i \rightarrow j$ 工件 j / $D_{ij}+S_{ij}=E_{ij}$ (分) / 工件 i	J_1	J_2	J_3	J_4	J_5
J_1	15	19	3	11	11
J_2	8	12	2	16	16
J_3	0	14	14	-2	12
J_4	6	10	6	8	8
J_5	6	10	4	10	10

步驟一：選取 P_i^1 值最小的工件為序列第一個工件 $J_{\sigma(1)}$，故以工件 J_1 為序列 Jσ 第一個工件。

步驟二：計算 $S_{J_1 j}$、$D_{J_1 j}$ 與 $E_{J_1 j}$，分別如表 4.2、表 4.3 與表 4.4 所示。

步驟三：從表 3.7 中可得到工件 J_3 為序列 Jσ 第二個工件。

步驟四：以同樣的方法對未排入的工件一個個排入序列中，最後得到此例的起始解為 $J\sigma=\{J_1, J_3, J_4, J_5, J_2\}$

步驟五：起始序列中最大的 $FT_{\sigma}^{J_{\sigma(i)}}$ 為 $FT_{\sigma}^{J_3}$=31，此序列 Jσ 的總排程時間為 31 分鐘。

4.3 鄰近解之產生法

鄰近解搜尋的目的，是希望起始解的表現能夠有更好的改善，因此，若能有目標的根據一個趨勢或是搜尋方向來對工件做移位，則能有效提高解的改善效率。由於 JNWF-SCM 排程問題的特性，所有工件皆經過第一部機台加工，故針對第一部機台的閒置時間來著手；工件的加工機台亦不盡相同，相鄰工件間的共用機台關係是一個良好的搜尋方向。

故本文提出的 ITS 演算法，以上一章所述的三種鄰近解方法：交換法(exchange)、插入法(insertion)與倒反法(invertion)，根據問題特性設計出一共五種新的鄰近解搜尋方法。此五種鄰近解搜尋方法說明如下：

(1) 第一種鄰近解搜尋法：兩工件交換法

假設目前的序列為 Jσ_k

步驟一：計算第一部機台上所有閒置時間

$\pi_{\sigma_k(i)}=(ST_{\sigma_k}^{b}-FT_{\sigma_k}^{a})+1$, $i=1,2,...,n-1$，a 與 b 代表此段閒置時機的前後相鄰工件。如圖 3.10 所示。

步驟二：由序列 Jσ_k 中隨機抽取兩組閒置時間 $\pi_{\sigma_k(i)}$、$\pi_{\sigma_k(j)}$ $(i\neq j)$。其中被抽取的機率設定為：

$$P_{\pi_{\sigma_k(i)}}=\frac{\pi_{\sigma_k(i)}}{\sum_{i=1}^{n-1}\pi_{\sigma_k(i)}} \qquad (1)$$

步驟三：從步驟二抽到的兩組閒置時間 $\pi_{\sigma_k(i)}$、$\pi_{\sigma_k(i)}$ 中，可以得知兩者分別的相鄰工件 J_a、J_b 與 J_c、J_d。將工件 J_b 與工件 J_d 做兩點交換法得到第一個鄰近解。

舉例說明，表 4.1 為例，起始解已知為 $\{J_1, J_3, J_4, J_5, J_2\}$，其對應的閒置時間區段以及甘特圖，如表 4.5 及圖 4.4 所示，假設隨機抽取閒置時間抽中 π_2 與 π_4，並得知 π_2 的相鄰工件為 J_3 與 J_4，π_4 的相鄰工件為 J_5 與 J_2，故以工件 J_4 與工件 J_2 作交換，得到的鄰近解為 $\{J_1, J_3, J_2, J_5, J_4\}$。

表 4.5　閒置時間區段抽取機率表

i	a	b	$\pi_{\sigma_k(i)} = ST^a_{\sigma_k} - FT^b_{\sigma_k}$	$P_{\pi_{\sigma_k(i)}}$
1	1	3	1-1+1=1	$\frac{1}{1+5+3+2} = 0.09$
2	3	4	7-3+1=5	$\frac{5}{1+5+3+2} = 0.46$
3	4	5	11-9+1=3	$\frac{3}{1+5+3+2} = 0.27$
4	5	2	14-13+1=2	$\frac{2}{1+5+3+2} = 0.18$

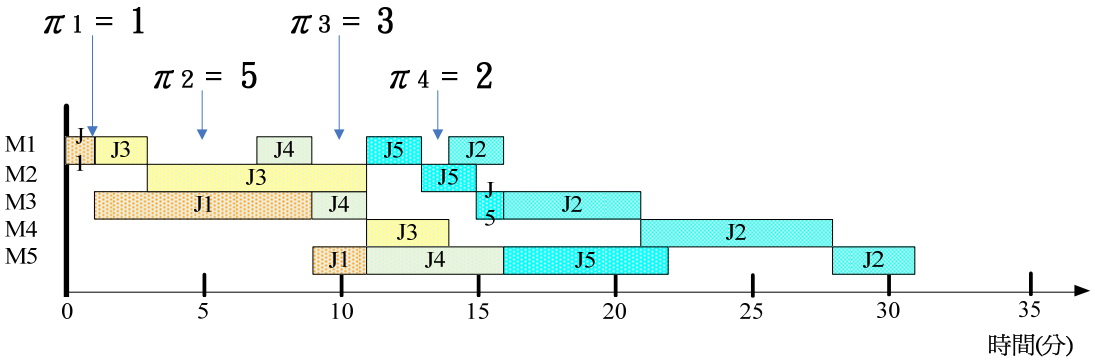

圖 4.4　閒置時間區段示意圖

(2) 第二種鄰近解搜尋法：三工件交換法

假設目前的序列為 $J\sigma_k$

步驟一：計算第一部機台上所有閒置時間 $\pi_{\sigma_k(i)}$, $i=1,2,...,n-1$ 。

步驟二：由序列 $J\sigma_k$ 中隨機抽取三組閒置時間 $\pi_{\sigma_k(i)}$ 、 $\pi_{\sigma_k(j)}$ 、 $\pi_{\sigma_k(k)}$ $(i \neq j \neq k)$ 。其中被抽取的機率同(1)式。

步驟三：從步驟二抽到的三組閒置時間 $\pi_{\sigma_k(i)}$ 、 $\pi_{\sigma_k(j)}$ 、 $\pi_{\sigma_k(k)}$ 中，可以得知三者分別的相鄰工件 J_a 、 J_b ， J_c 、 J_d 與 J_e 、 J_f 。將工件 J_d 、工件 J_d 與工件 J_f 做三點交換法得到第二個鄰近解。

舉例說明，表 4.1 為例，起始解已知為 $\{J_1, J_3, J_4, J_5, J_2\}$ ，其對應的甘特圖如圖 4.4 所示，假設隨機抽取閒置時間抽中 π_2 、 π_3 與 π_4 ，並得知 π_2 的相鄰工件為 J_3 與 J_4 ， π_3 的相鄰工件為 J_4 與 J_5 ， π_4 的相鄰工件為 J_5 與 J_2 ，故以工件 J_4 、工件 J_4 與工件 J_2 作交換，得到的鄰近解為 $\{J_1, J_3, J_2, J_4, J_5\}$ 。

(3) 第三種鄰近解搜尋法：兩工件間倒反序列法

假設目前的序列為 $J\sigma_k$

步驟一：計算第一部機台上所有閒置時間 $\pi_{\sigma_k(i)}$, $i=1,2,...,n-1$ 。

步驟二：由序列 $J\sigma_k$ 中隨機抽取二組閒置時間 $\pi_{\sigma_k(i)}$ 、 $\pi_{\sigma_k(j)}$ $(i \neq j)$ 。其中被抽取的機率同(1)式。

步驟三：從步驟二抽到的二組閒置時間 $\pi_{\sigma_k(i)}$ 、 $\pi_{\sigma_k(j)}$ 中，可以得知二者分別的相鄰工件 J_a 、 J_b 與 J_c 、 J_d 。將工件 J_b 與工件 J_c 之間的工件做兩點間倒反法得到第三個鄰近解。

舉例說明，表 4.1 為例，起始解已知為 $\{J_1, J_3, J_4, J_5, J_2\}$ ，其對應的甘特圖如圖 4.4 所示，假設隨機抽取閒置時間抽中 π_2 與 π_4 ，並得知 π_2 的相鄰工件為 J_3 與 J_4 ， π_4 的相鄰工件為 J_5 與 J_2 ，故以工件 J_4 與工件 J_5 之間的工

件做倒序排列，得到的鄰近解為$\{J_1, J_3, J_5, J_4, J_2\}$。

(4) 第四種鄰近解搜尋法：單一工件插入法(一)

假設目前的序列為$J\sigma_k$

步驟一：計算第一部機台上所有閒置時間$\pi_{\sigma_k(i)}$，$i=1,2,...,n-1$。

步驟二：由序列$J\sigma_k$中隨機抽取一組閒置時間$\pi_{\sigma_k(i)}$。其中被抽取的機率同(1)式。並且得知此閒置時段的相鄰工件J_a與J_b。

步驟三：計算工件J_a與其他工件中，任一工件的共用機台的關係式。假設工件J_i與工件J_j共用機台數有t部，共用機台數的關係式為[20]：

$$W_{J_i,J_j} = t_{Ji,Jj} + m(m - t_{Ji,Jj}) = m^2 - t_{Ji,Jj}(m-1) \qquad (2)$$

當工件J_i與工件J_j的共用機台數愈小，則參數W_{J_i,J_j}愈大。

步驟四：由工件J_a中，除了J_b的其餘工件，隨機選取一個工件J_j。此工件被抽取的機率與工件J_a的共用機台數有關。工件J_j被抽取的機率設定為：

$$P'_{J_a,J_j} = \frac{W_{J_a,J_j}}{\sum_{r=1}^{n} W_{J_a,J_r}} \quad , r \neq a \qquad (3)$$

步驟五：將步驟四抽到的工件J_j，在排序上插入到工件J_a的後面，得到第四個鄰近解。

舉例說明，表 4.1 為例，起始解已知為$\{J_1, J_3, J_4, J_5, J_2\}$，其對應的甘特圖如圖 4.4 所示，假設隨機抽取閒置時間抽中π_2，並得知π_1的相鄰工件為J_3與J_4，故計算工件J_3與其他工件的共用機台數權重，並計算被抽取機率，如表 4.6。假設抽到工件J_1，則工件J_1插入到工件J_3的後面，得到的鄰近解為$\{J_3, J_1, J_4, J_5, J_2\}$。

表 4.6　工件共用機台權重與機率表(1)

$i \rightarrow j$ 共用機台權重值 (工件 j / 工件 i)	J_1	J_2	J_3	J_4	J_5
J_3	21	17	-	21	17
抽中機率	0.28	0.22	-	0.28	0.22

(5) 第五種鄰近解搜尋法：單一工件插入法(二)

假設目前的序列為 $J\sigma_k$

步驟一：從序列 $J\sigma_k$ 中，可得到最晚完成加工的工件 J_{last}

步驟二：計算工件 J_{last} 與其他工件中，任一工件的共用機台的關係式，並且依照此關係式隨機抽取一個工件 J_j。其中共用機台關係式與被抽取的機率分別同(1)式與(2)式。

步驟三：將步驟二抽到的工件 J_j，在排序上插入到工件 J_{last} 的後面，得到第五個鄰近解。

舉例說明，表 4.1 為例，起始解已知為 $\{J_1, J_3, J_4, J_5, J_2\}$，其對應的甘特圖如圖 4.4 所示，已知最晚完成加工的工件為 J_2，故計算工件 J_2 與其他工件的共用機台數權重與工件的抽取機率，如表 4.7。假設抽到工件 J_3，則工件 J_3 插入到工件 J_2 的後面，得到的鄰近解為 $\{J_1, J_4, J_5, J_2, J_3\}$。

表 4.7　工件共用機台權重與機率表(2)

$i \rightarrow j$ 共用機台權重值 (工件 j / 工件 i)	J_1	J_2	J_3	J_4	J_5
J_2	13	-	17	13	13
抽中機率	0.23	-	0.31	0.23	0.23

第五章　模擬與實驗分析

因受限於本研究討論的情境屬於新課題，尚未有此情境適合的資料庫作為執行的標竿(benchmark)，故本章針對 JNWF-SCM 排程問題，設計不同的工件數(n)、機台數(m)、工件加工時間參數(p)、以及停止條件(ep)等一系列的案例資料，並以 ITS、EXTS1、EXTS2 與 TS2(李東森[20])來執行，以比較之間的績效。

本章先以一個小型案例，說明本研究提出的三種演算法的結果。

5.1　案例說明

給定小型案例的工件數為 n=7，機台數為 m=5，加工時間 p=U[1,15]，停止條件 ep=10，說明 ITS、EXTS1 與 EXTS2 的演算結果。工件的加工時間以 U[1,15]的機率分配隨機產生，各工件在機台 M_1 至 M_5 的加工時間以 0.3 的機率為 0，產生結果如下表：

表 5.1　JNWF-SCM 的工件加工時間表(4)

加工時間(分) 機台 / 工件	M_1	M_2	M_3	M_4	M_5	**Total**
J_1	7	-	15	5	-	**27**
J_2	2	-	13	-	6	**21**
J_3	15	10	5	-	12	**42**
J_4	4	2	5	-	4	**15**
J_5	8	13	5	13	4	**43**
J_6	6	11	9	9	8	**43**
J_7	13	-	-	3	7	**23**

接著利用 ITS、EXTS1 與 EXTS2 三種演算法分別求得此案例的排程結果。

首先求得的起始解分別為：ITS＝$\{J_2, J_7, J_4, J_1, J_3, J_5, J_6\}$，

EXTS1＝$\{J_2, J_6, J_5, J_1, J_4, J_3, J_7\}$，EXTS2＝$\{J_2, J_6, J_3, J_5, J_1, J_7, J_4\}$，三者的甘特圖如圖 5.1，總排程時間分別為 104 分鐘、99 分鐘、93 分鐘。

並以 Epoch length=10 為停止條件，完整執行三種方法後求得的最後結果分別為：ITS ＝$\{J_2, J_5, J_3, J_6, J_7, J_1, J_4\}$，EXTS1 ＝$\{J_2, J_6, J_4, J_1, J_5, J_3, J_7\}$，EXTS2＝$\{J_2, J_6, J_7, J_0, J_5, J_3, J_4\}$，三者的甘特圖如圖 5.2，其中以 ITS 得到的總排程時間最短，只有 85 分鐘，表 5.2 是從 ITS 演算結果得到的工件投料時間表。

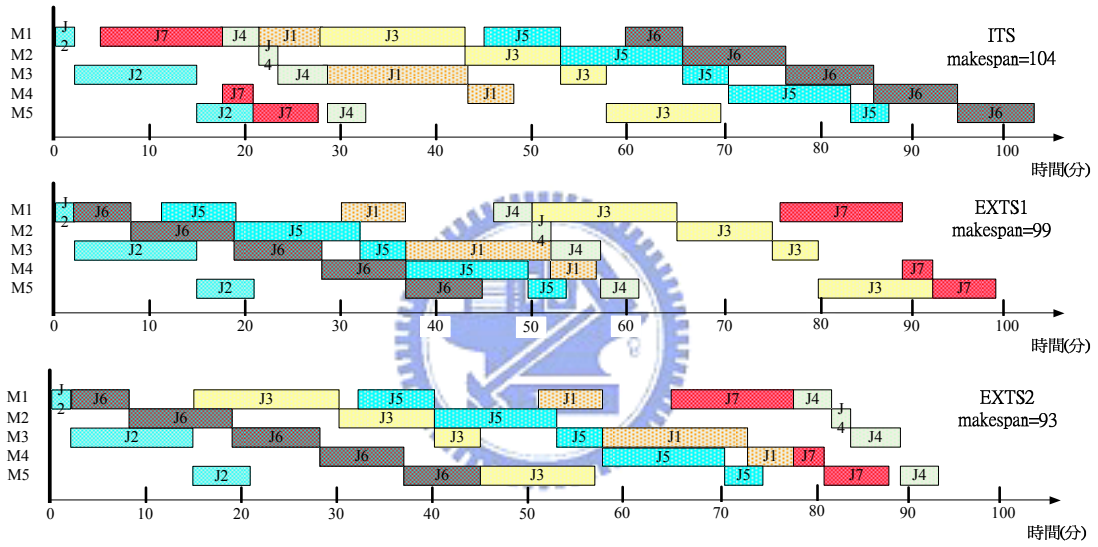

圖 5.1　三種方法之起始解甘特圖

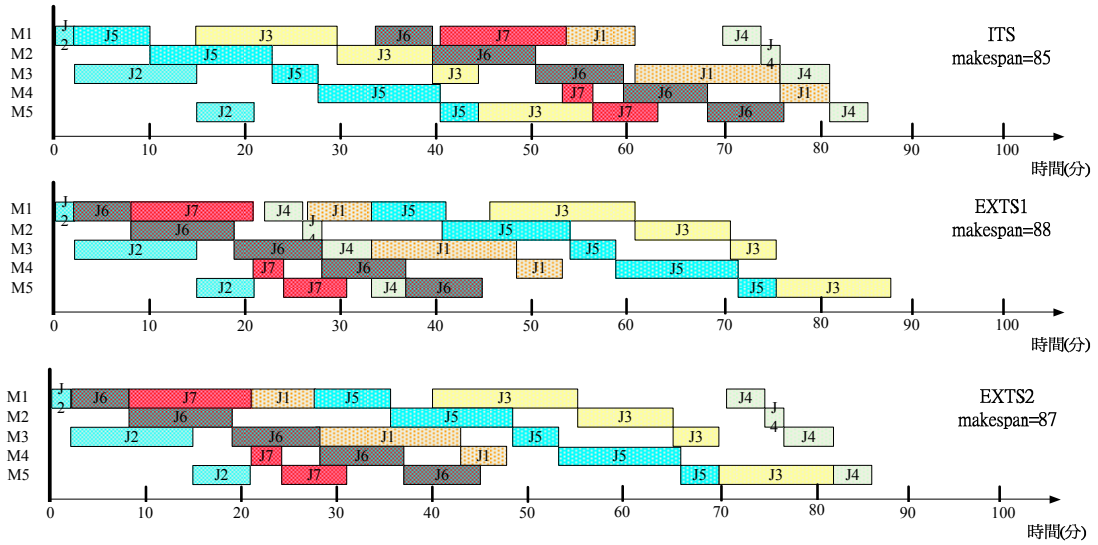

圖 5.2　三種方法之最終解甘特圖

表 5.2　工作投料時間表

		Machine1		Machine2		Machine3		Machine4		Machine5	
		Start	Finish	Start	Finish	Start	Finish	Start	Finish	Start	Finish
job	1	54	61	-	-	61	76	76	81	-	-
	2	0	2	-	-	2	15	-	-	15	21
	3	15	30	30	40	40	45	-	-	45	57
	4	70	74	74	76	76	81	-	-	81	85
	5	2	10	10	23	23	28	28	41	41	45
	6	34	40	40	51	51	60	60	69	69	77
	7	41	54	-	-	-	-	54	57	57	64

5.2　績效評估

本研究設計了數個 JNWF-SCM 排程問題來進行測試，在模擬的實驗中，工件數 n 分別為 20、30、50、100、200；機台數 m 分別為 5、10、15、25；工件在各機台的跳機台狀況以 0.3 的機率發生，其餘機台的加工時間 p 則由母體以均勻分配產生，範圍分別為 U[1,20]、U[1,50]、U[1,100]；停止條件 Epoch length(ep)分別為 50 與 100；所以一共有 120 種組合的案例。四種演算法皆以工件加工時間的母體分配，對每一種案例組合隨機執行 30 次，故總共需要 14400 次實驗。各演算法中，針對每一種組合 30 次的執行結果，統計出總排程時間平均值(以下簡稱總排程時間)、總排程時間平均標準差(以下簡稱標準差)與其運算時間平均值(以下簡稱運算時間)來做探討與分析。

本研究提出的三個演算法與 TS2 皆採用 C++程式語言撰寫與執行，並利用 Microsoft Excel 2002 軟體做數據的整理與圖表統計分析，所有的測試皆在 AMD Athlon 3500 2.21GHz，記憶體 990RAM 以及 Windows XP 作業系統的平台上執行。

所有案例組合經過執行後，皆整理於本論文的附錄，以下將針對案例的結果做績效的分析與說明，分成解的品質與求解時間二方面來評估。

解的品質方面：

(1) 全部 120 種組合中，ITS 在總排程時間上的表現，全數優於 EXTS1、EXTS2 與 TS2。

(2) 大型案例中，ITS 相較於 EXTS1 與 EXTS2 有非常明顯差距的優勢，如表 5.3 與表 5.4 中，優勢百分比普遍的比小型案例來的高，其中「improve」該欄表示 ITS 在平均總排程時間上優於 EXTS1、EXTS2 與 TS2 的平均優勢百分比。

平均優勢百分比的計算式為：$\frac{avg.\ makespan_{other\ algorithm} - avg.\ makespan_{ITS}}{avg.\ makespan_{other\ algorithm}}$

(3) 所有組合中，ITS 在 Epoch length 增加的情況下，總排程時間也仍保持著明顯優勢，如圖 5.4 為例。

(4) 所有組合中，ITS 的標準差與另三者演算法比較下，有時較大但亦有時較小，綜合來說，四種演算法在標準差表現上是相近的。

表 5.3　四種演算法在小型案例之績效

n	m	p	ep	Makespan							
				ITS		EXTS1		EXTS2		TS2	
				mean	improve	mean	improve	mean	improve	mean	improve
30	5	20	50	365.9	0.0%	389.4	6.1%	393.1	6.9%	389.1	6.0%
			100	356.3	0.0%	387.7	8.1%	389.2	8.4%	387.4	8.0%
		50	50	887.5	0.0%	939.4	5.5%	958.4	7.4%	955.8	7.1%
			100	870.0	0.0%	937.0	7.2%	938.3	7.3%	937.1	7.2%
		100	50	1808.6	0.0%	1887.0	4.2%	1882.9	3.9%	1867.6	3.2%
			100	1774.0	0.0%	1868.5	5.1%	1869.4	5.1%	1864.1	4.8%
	10	20	50	516.0	0.0%	524.5	1.6%	520.7	0.9%	523.7	1.5%
			100	507.4	0.0%	522.6	2.9%	519.0	2.2%	511.8	0.9%
		50	50	1255.5	0.0%	1293.3	2.9%	1269.0	1.1%	1275.5	1.6%
			100	1202.4	0.0%	1282.5	6.2%	1267.8	5.2%	1261.5	4.7%
		100	50	2473.8	0.0%	2551.1	3.0%	2567.3	3.6%	2568.6	3.7%
			100	2434.3	0.0%	2516.9	3.3%	2539.6	4.1%	2465.0	1.2%
						avg.	**4.7%**	**avg.**	**4.7%**	**avg.**	**4.2%**

表 5.4　四種演算法在大型案例之績效

n	m	p	ep	Makespan							
				ITS		EXTS1		EXTS2		TS2	
				mean	improve	mean	improve	mean	improve	mean	improve
200	15	20	50	3646.0	0.0%	3846.0	5.2%	3791.0	3.8%	3708.4	1.7%
			100	3629.9	0.0%	3807.9	4.7%	3786.4	4.1%	3691.4	1.7%
		50	50	8910.3	0.0%	9475.3	6.0%	9347.3	4.7%	9107.7	2.2%
			100	8905.7	0.0%	9415.0	5.4%	9331.6	4.6%	9039.1	1.5%
		100	50	17692.0	0.0%	18903.8	6.4%	18708.9	5.4%	18137.9	2.5%
			100	17675.6	0.0%	18811.8	6.0%	18477.8	4.3%	18059.3	2.1%
	25	20	50	4653.7	0.0%	4977.5	6.5%	4964.9	6.3%	4777.9	2.6%
			100	4641.0	0.0%	4974.4	6.7%	4949.3	6.2%	4756.9	2.4%
		50	50	11401.7	0.0%	12401.6	8.1%	12290.8	7.2%	11692.3	2.5%
			100	11380.5	0.0%	12350.0	7.8%	12200.7	6.7%	11688.7	2.6%
		100	50	22738.7	0.0%	24569.7	7.5%	24370.7	6.7%	23361.2	2.7%
			100	22672.7	0.0%	24486.2	7.4%	24345.8	6.9%	23263.9	2.5%
						avg.	**6.5%**	**avg.**	**5.6%**	**avg.**	**2.2%**

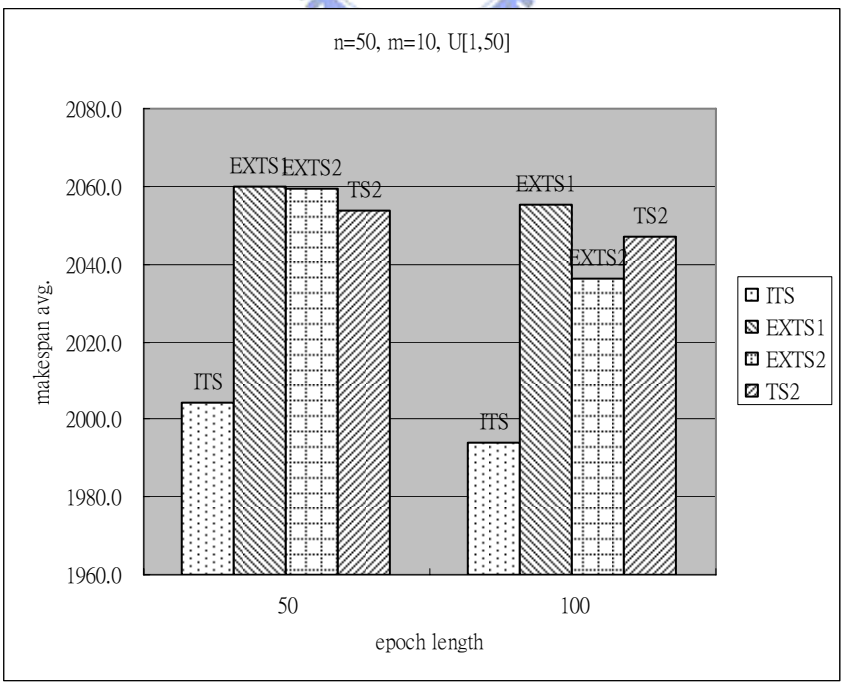

圖 5.3　四種方法在不同 Epoch length 情況下平均總排程時間

求解時間方面：

(1) ITS 的求解時間表現在小型案例上並非最優，但與 EXTS1、EXTS2、TS2 差距不大，如圖 5.4 的例子所示。

(2) 大型案例上，ITS 比其他四種方法則有非常明顯的優勢，如圖 5.5 的例子所示。

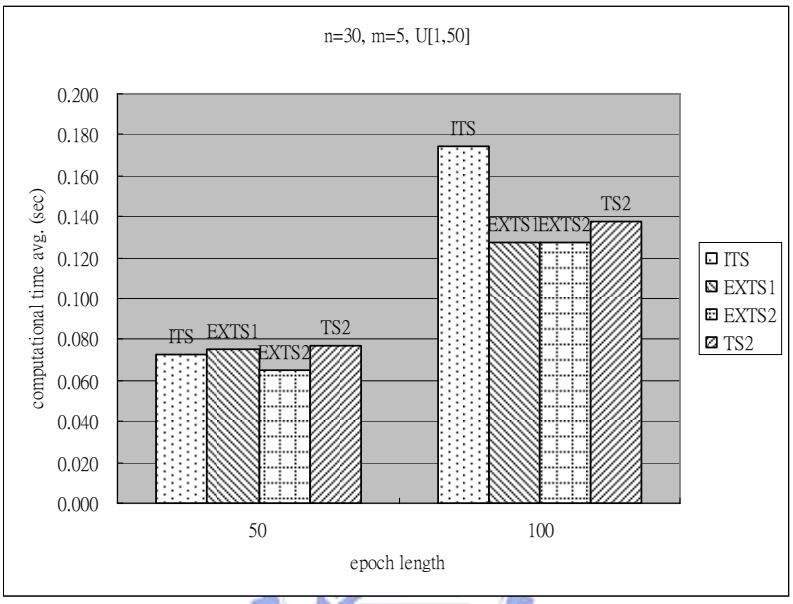

圖 5.4　小型案例中三種方法的求解時間

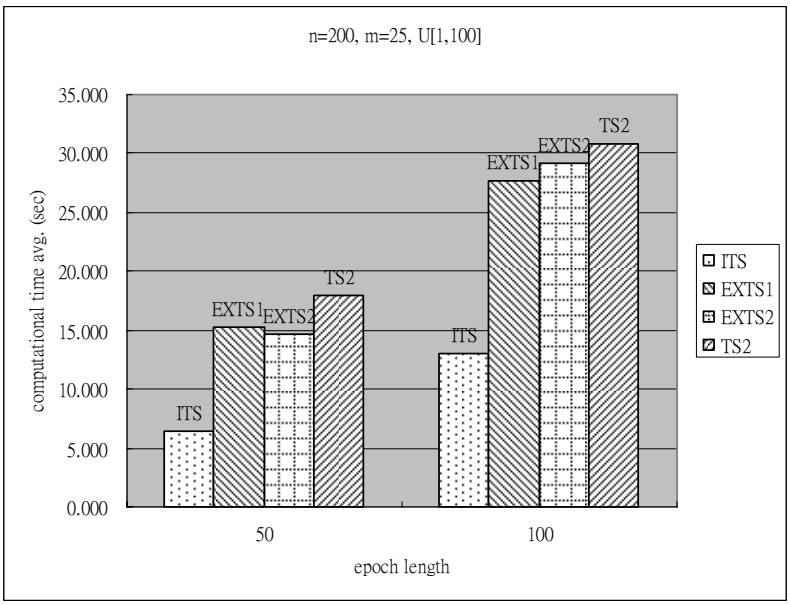

圖 5.5　大型案例中三種方法的求解時間

5.3 小結

以下對本章做簡單的結論：

1. 在各種類型的案例上，ITS 所得到的總排程時間表現皆優於 EXTS1、EXTS2 與 TS2，尤其在大型案例中，不僅總排程時間，運算時間皆明顯優於另外三者，故 ITS 比其他方法更適合處裡較大型的案例。
2. ITS 在處裡小型案例中，運算時間表現劣於 EXTS1 與 EXTS2，這是因為 ITS 的運算較為複雜。
3. ITS 在處裡大型案例中，總排程時間與運算時間的表現皆優於 EXTS1 與 EXTS2，這是因為 ITS 的起始解優於另兩者演算法的起始解，故演算結果的表現比較優。
4. ITS 比 TS2 的績效更優，顯示出本研究所提的方法，確實針對 JNWF-SCM 排程問題來求解。

第六章　結論與未來研究方向

本研究是在 JNWF-SCM 生產線情境下，探討如何決定工件的投料時間，並以最小化總排程時間(makespan)為目標。此種問題的複雜度屬於 NP-hard，求解時間與工件數或機台數成指數遞增關係，所以當工件數或機台數太大時，無法用解析法或數值法來求解。

故本研究提出了一套適用於 JNWF-SCM 的演算法 ITS，另外分別從文獻中 NWF 排程問題的兩個演算法 Bertolissi[1]與 Rajendran[14]，分別做為本研究 EXTS1 與 EXTS2 演算法的起始解。並設計了數個 JNWF-SCM 排程問題的案例，以 ITS、EXTS1、EXTS2 與 TS2 [20]執行這些案例。

6.1　結論

本研究所討論的情境為多機台 JNWF-SCM 排程問題，設計了三種演算法來求解。過去研究[10]曾有討論二機台環境的排程問題，但並不適合多機台環境的排程問題；另有文獻探討多部機台 NWF-J 排程問題，比 JNWF-SCM 情境更為一般性。然而從案例結果發現，在小型案例上，由於 ITS、 EXTS1、 EXTS2 與 TS2 所求得的解皆靠近最佳解，故此四種演算法求得的總排程時間差距不大，但 ITS 仍優於其他三者；在大型案例上，ITS 求得的解則明顯的優於其他三方法。然而，在相同的系統參數下處裡的案例時，在小型案例上，由於 ITS 在鄰近解搜尋有較複雜的運算，故 ITS 的運算時間沒有較 EXTS1 與 EXTS2 來的短；但大型案例上，ITS 的運算時間則明顯的優於 EXTS1 與 EXTS2。

產業結構的變遷，業者傾向客製化產品來滿足顧客需求，提高服務水準，因此 JNWF-SCM 生產線已經是普遍的現象。本論文針對 JNWF-SCM 排程問題提出一個解決的方法，可以幫助工業界中縮短產品的生產週期，達到迅速交貨滿足顧客，或是在一些產業及工廠裡減少存貨的成本，包括備載原物料、生產線上的在製品、完成品的墩積等所產生的浪費，藉由事先良好的排程，以期可以有效的

控管這些資源，避免不必要的浪費，使得企業能夠以高品質、低成本、保證交期來提高市場競爭力，顯現出本研究探討的課題之重要性。

6.2 未來研究方向

以下幾點為延伸之討論範疇，在未來發展方面，亦可由此方向進行：

1. 突破單一機台的限制，結合各階段多部等效之平行機台，來處裡 JNWF-SCM 排程問題。
2. 為了更符合現場生產情境，工件的類型可依據實際案例來做分類或套入參數的設定。
3. 未來可發展其他啟發式演算法，例如模擬退火法(Simulated Annealing)、螞蟻群聚最佳化(Ant Colony Optimization)、基因演算法(Genetic Algorithm)等 meta-heuristic，並彼此比較求解效率。

參考文獻

【1】Bertolissi, E(2000), “Heuristic algorithm for scheduling in the no-wait flow-shop,” *Journal of Materials Proceeding Technology* Vol.107, pp.459-465.

【2】Bonney, M. C. and S.W. Gundry(1976), “Solutions to the constrained flowshop sequencing problem,” *Operational Research* Vol.27, pp.869-883.

【3】Campbell, H.G., R.A. Dudek, and M.L. Smith(1970), "A heuristic algorithm for the n-job, M-machine sequencing problem", *Management Science B* Vol.16, pp.630-637.

【4】Chen, C., V. Neppalli, and N. Aljaber(1996), ”Genetic algorithms applied to the continuous flow shop problem,” *Computers and Industrial Engineering* Vol 30. pp.919-29.

【5】Fink, A. and S. Voß(2003), “Solving the continuous flow-shop scheduling problem by metaheuristics,” *European Journal of Operational Research* Vol.151, pp.400-414.

【6】Framinana, J. M. and C. Schuster(2006), “An enhanced timetabling procedure for the no-wait job shop problem:a complete local search,” *Computers & Operations Research* Vol.331, pp.1200-1213.

【7】Gangadharan, R. and C. Rajendran(1993), “Heuristic algorithms for scheduling in the no-wait flowshop,” *International Journal of Production Economics* Vol.32, pp.285-290.

【8】Ghosh, D. and G. Sierksma(2002), “Complete local search with memory,” *Journal of Heuristics* Vol.8, pp.571-84.

【9】Gilmore , P.C. and R.E. Gomory(1964), ”Sequencing a one state variable machine: a solvable case of the travelling salesman problem,” *Operations Research* Vol. 12, pp.655–679.

【10】Glass, C.A., J.N.D. Gupta, and C.N. Potts(1999), ”Two-machine no-wait flow

shop scheduling with missing operations," *Mathematiccs of Operations Research* Vol. 24, pp. 911-923.

【11】Goyal, S. K.(1973), "On the Flow-Shop Sequencing Problem with No Wait in Process", *Operational Research Quarterly* Vol. 24, No. 1, pp.130-133.

【12】Nawaz, M., E. Enscore ,and I. Ham(1983), "A heuristic algorithm for the m-machine, n-job flowshop sequencing problem," *The International Journal of Management Science* Vol. 11, pp. 91-95.

【13】Palmer, D. S.(1965), "Sequencing jobs through a multi-stage process in the minimum total time - A quick method of obtaining a near optimum,"*Operational Research Quarterly* Vol. 16, pp.101-107.

【14】Rajendran, C.(1994), "A No-Wait Flowshop Scheduling Heuristic to Minimize Makespan," *The Journal of the Operational Research Society* Vol. 45, no. 4, pp. 472-478.

【15】Reddi, S. S. and C. V. Ramamoorthy(1972), "On the Flow-Shop Sequencing Problem with No Wait in Process," *Operational Research Quarterly* Vol. 33, No. 3, pp.323-331.

【16】Rock, H.(1984), "The three-machine no-wait flowshop problem is NP-complete," *Journal of the Association for Computing Machinery* Vol.31, pp.336–345.

【17】Sahni, S. and Y. Cho(1979), "Complexity of scheduling shops with no wait in process," *Mathematics Operation Research* Vol.4, pp.448-457.

【18】Schustera, C. J. and J. M. Framinanb(2003), "Approximative procedures for no-wait job shop scheduling," *Operation Research Letters* Vol.31, pp.308-318.

【19】Shyu, S. J., B.M.T. Lin and P.Y. Yin(2004), "Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time," *Computers & Industrial Engineering* Vol.47, pp.181-193.

【20】李東森 (2007)：跳躍式 no-wait 生產線工件排程方法之研究。國立交通大學工業工程與管理學研究所碩士論文。

附錄 A

Bertolissi[1] 演算方法：

Step1：任兩工件交互排列成 J_i - J_j 與 J_j - J_i ，比較兩種排序的總流程時間總合 (sum of the total flowtime)。一共有 C_2^n 組。

Step2：若 J_i - J_j 表現比 J_j - J_i 較佳，則 J_i 標記一次。

Step3：統計所有工件的標記次數，

Step4：依工件標記次數，由大到小排列取得一組序列，即為起始序列。若有二個以上工件計次.相同時，則以工件的總作業時間取小者排前。

Rajendran[14] 演算方法：

Step1：將所有工件分成二群，A 群與 B 群分別為： $A = \{i \mid R_i \geq \frac{(1+m)}{2}\}$ ，

$B = \{i \mid R_i < \frac{(1+m)}{2}\}$ ，其中 $R_i = \frac{\sum_{h=1}^{m} i \times p_i^h}{\sum_{h=1}^{m} p_i^h}$ 。

Step2：將 A、B 兩群做各自群內的排列，以計算值 $\sum_{h=1}^{m}(m-h+1)P_i^h$ 為依據，A 組內工件依此計算值由小到大排列，得到一組新序列 A’，B 組內工件依此計算值由大到小排列，得到一組新序列 B’。 若有二個以上工件的值相同，則以工件的總作業時間取小者排前。

Step3：結合 A’與 B’，形成一組完整的序列 A’-B’，即為起始序列。

附錄 B

以下表格為 ITS、EXTS1、EXTS2、TS2 在各種不同的 n、m、p、ep 組合之下的案例彙整表，包含了總排程時間平均值、總排程時間平均標準差與其運算時間平均值。

表 B.1　演算法在大型案例之求解結果(1)

n	m	p	ep	Makespan								Time			
				ITS		EXTS1		EXTS2		TS2		ITS	EXTS1	EXTS2	TS2
				mean	Std.	mean	Std.	mean	Std.	mean	Std.				
20	5	20	50	253.6	23.4	271.6	21.0	265.5	25.0	258.8	21.3	0.040	0.051	0.051	0.033
			100	238.9	26.7	261.9	21.0	262.3	21.5	258.6	18.4	0.086	0.117	0.085	0.062
		50	50	599.6	63.3	657.5	60.7	640.9	59.6	649.7	58.7	0.045	0.049	0.047	0.037
			100	599.1	83.2	632.9	55.2	638.3	44.4	639.8	61.4	0.082	0.093	0.086	0.037
		100	50	1200.9	114.0	1285.4	130.8	1270.2	107.0	1279.5	112.8	0.046	0.054	0.055	0.034
			100	1142.4	124.5	1276.5	119.3	1257.1	73.9	1232.9	124.9	0.090	0.108	0.078	0.066
	10	20	50	353.9	24.8	368.1	21.8	356.6	24.9	361.3	23.7	0.094	0.100	0.094	0.076
			100	350.2	25.6	364.3	26.0	355.8	17.9	361.2	26.7	0.159	0.172	0.165	0.136
		50	50	866.8	67.2	894.3	44.1	879.2	57.4	890.4	56.5	0.091	0.088	0.080	0.082
			100	813.0	83.4	881.7	55.9	865.6	67.9	861.4	67.3	0.167	0.179	0.174	0.142
		100	50	1729.5	183.4	1781.0	114.2	1774.2	111.5	1764.8	126.0	0.083	0.094	0.085	0.078
			100	1715.0	153.7	1748.4	103.4	1728.5	112.9	1718.8	124.8	0.160	0.163	0.167	0.127
	15	20	50	455.9	25.1	462.2	25.9	460.7	22.5	461.5	23.0	0.137	0.128	0.137	0.133
			100	433.7	29.4	449.4	26.6	443.9	24.9	454.0	22.4	0.277	0.239	0.218	0.219
		50	50	1106.9	61.9	1148.9	54.0	1130.4	45.3	1144.7	62.4	0.124	0.139	0.136	0.127
			100	1057.3	71.1	1111.4	64.0	1096.4	65.1	1116.3	64.9	0.257	0.251	0.257	0.217
		100	50	2180.5	117.6	2253.7	106.4	2239.7	91.2	2219.6	126.0	0.136	0.131	0.116	0.131
			100	2150.8	139.9	2236.3	123.8	2204.2	121.7	2210.8	120.4	0.255	0.261	0.260	0.232
	25	20	50	612.3	25.0	622.1	30.9	622.0	30.2	627.5	28.1	0.514	0.236	0.250	0.215
			100	608.3	36.0	620.6	30.4	617.0	27.1	619.8	35.1	0.497	0.447	0.390	0.446
		50	50	1525.6	83.4	1531.2	72.7	1542.9	59.8	1537.6	76.7	0.229	0.251	0.215	0.248
			100	1503.7	83.8	1521.9	61.7	1511.7	66.6	1517.1	61.3	0.470	0.414	0.409	0.413
		100	50	2973.7	146.4	3130.0	141.8	3049.9	123.0	3051.0	152.5	0.241	0.221	0.184	0.222
			100	2950.6	136.6	3039.4	147.7	3071.6	106.6	2994.2	126.4	0.476	0.513	0.218	0.476

表 B.2　演算法在大型案例之求解結果(2)

n	m	p	ep	Makespan								Time			
				ITS		EXTS1		EXTS2		TS2		ITS	EXTS1	EXTS2	TS2
				mean	Std.	mean	Std.	mean	Std.	mean	Std.				
30	5	20	50	365.9	27.3	389.4	28.3	393.1	28.7	389.1	19.8	0.094	0.079	0.075	0.070
			100	356.3	36.4	387.7	24.9	389.2	24.9	387.4	27.1	0.193	0.154	0.131	0.119
		50	50	887.5	67.3	939.4	65.4	958.4	61.1	955.8	84.4	0.072	0.076	0.065	0.077
			100	870.0	81.9	937.0	87.5	938.3	48.8	937.1	72.0	0.175	0.128	0.127	0.138
		100	50	1808.6	126.9	1887.0	118.8	1882.9	118.9	1867.6	126.2	0.066	0.079	0.071	0.069
			100	1774.0	171.8	1868.5	117.4	1869.4	139.8	1864.1	156.8	0.142	0.130	0.130	0.124
	10	20	50	516.0	26.2	524.5	23.9	520.7	33.7	523.7	30.7	0.180	0.136	0.137	0.156
			100	507.4	34.1	522.6	25.0	519.0	29.1	511.8	24.7	0.317	0.290	0.258	0.314
		50	50	1255.5	80.2	1293.3	54.5	1269.0	61.0	1275.5	77.6	0.157	0.143	0.133	0.138
			100	1202.4	73.6	1282.5	62.5	1267.8	69.9	1261.5	59.4	0.340	0.247	0.233	0.291
		100	50	2473.8	165.2	2551.1	124.5	2567.3	115.1	2568.6	120.9	0.150	0.138	0.133	0.137
			100	2434.3	136.2	2516.9	129.9	2539.6	149.1	2465.0	107.3	0.331	0.245	0.278	0.281
	15	20	50	641.7	34.5	649.8	36.8	651.4	29.8	645.9	35.2	0.259	0.220	0.194	0.238
			100	633.1	27.2	634.7	30.5	639.4	28.0	635.0	25.1	0.502	0.462	0.432	0.434
		50	50	1579.9	72.2	1591.6	67.7	1599.6	71.8	1597.0	83.3	0.265	0.220	0.228	0.248
			100	1549.7	60.1	1569.6	71.2	1587.5	57.7	1578.9	85.0	0.564	0.473	0.400	0.431
		100	50	3123.8	152.1	3164.5	148.3	3203.8	135.9	3178.8	165.9	0.293	0.240	0.211	0.241
			100	3087.9	161.5	3153.0	134.2	3139.1	130.3	3140.1	173.1	0.517	0.442	0.364	0.402
	25	20	50	878.0	32.5	885.9	28.3	884.4	29.4	885.8	34.1	0.461	0.450	0.342	0.435
			100	852.4	36.4	870.6	31.6	867.5	30.7	872.3	33.4	1.108	0.780	0.752	0.928
		50	50	2113.4	92.5	2168.6	72.2	2138.6	78.9	2165.5	64.2	0.501	0.450	0.754	0.451
			100	2107.6	101.1	2127.3	70.5	2124.1	90.6	2121.9	84.0	0.899	0.799	0.385	0.909
		100	50	4246.7	170.4	4339.3	168.1	4353.0	157.9	4353.0	180.9	0.418	0.432	0.385	0.429
			100	4205.9	138.8	4232.1	142.7	4247.8	125.2	4234.2	160.6	1.100	0.877	0.845	0.982

表 B.3　演算法在大型案例之求解結果(3)

n	m	p	ep	Makespan								Time			
				ITS		EXTS1		EXTS2		TS2		ITS	EXTS1	EXTS2	TS2
				mean	Std.	mean	Std.	mean	Std.	mean	Std.				
50	5	20	50	609.9	32.2	640.5	35.9	634.6	27.8	641.1	34.5	0.164	0.167	0.142	0.166
			100	593.2	36.1	631.1	30.8	630.7	36.3	640.4	44.1	0.313	0.317	0.250	0.156
		50	50	1504.0	82.8	1595.1	97.1	1572.6	98.1	1576.7	92.2	0.161	0.173	0.176	0.144
			100	1468.0	156.5	1558.9	104.6	1543.9	105.6	1543.9	84.4	0.427	0.309	0.295	0.291
		100	50	2959.9	223.1	3070.8	158.3	3119.1	186.7	3087.8	169.7	0.205	0.303	0.158	0.145
			100	2905.5	164.9	3062.8	171.2	3084.9	159.7	2969.0	122.9	0.372	0.321	0.273	0.334
	10	20	50	820.2	36.8	841.3	33.6	837.5	31.5	838.3	34.0	0.344	0.359	0.339	0.310
			100	804.7	38.1	837.0	27.1	834.8	33.9	819.7	40.8	0.682	0.695	0.591	0.649
		50	50	2004.4	99.7	2060.1	100.5	2059.6	111.0	2078.7	82.8	0.349	0.236	0.322	0.314
			100	1994.1	83.3	2055.1	105.2	2036.2	87.1	2047.7	107.4	0.730	0.340	0.585	0.735
		100	50	4087.2	174.6	4129.2	128.6	4122.8	228.5	4146.7	183.7	0.322	0.412	0.312	0.300
			100	3942.1	156.5	4088.6	163.3	4068.6	162.4	4080.9	151.1	0.589	0.628	0.707	0.541
	15	20	50	1015.3	37.6	1022.3	39.9	1024.3	38.1	1036.7	37.1	0.596	0.530	0.496	0.713
			100	1015.0	28.5	1019.4	39.4	1015.8	32.3	1021.3	42.5	1.039	0.982	0.958	1.206
		50	50	2503.9	94.0	2538.4	81.4	2524.2	93.7	2571.6	110.6	0.606	0.520	0.528	0.668
			100	2489.7	95.8	2520.8	82.3	2508.0	95.1	2508.7	77.3	1.060	0.997	0.921	1.053
		100	50	4995.1	183.9	5078.6	179.2	5019.9	137.9	5051.7	162.5	0.483	0.629	0.585	0.535
			100	4911.4	194.5	4994.5	202.0	5003.0	172.3	5000.9	146.8	1.274	0.965	0.910	1.292
	25	20	50	1356.6	32.0	1381.7	35.3	1374.0	39.6	1384.1	53.1	0.851	1.113	0.957	0.995
			100	1347.9	37.0	1363.9	35.1	1364.2	39.9	1358.9	41.1	1.899	2.097	1.800	2.490
		50	50	3350.9	117.7	3422.0	83.1	3385.0	98.3	3380.7	91.9	0.986	1.132	1.087	1.083
			100	3315.5	89.8	3367.5	89.0	3353.7	94.7	3359.9	117.3	1.834	2.427	1.870	2.609
		100	50	6675.2	195.9	6794.5	211.9	6734.9	259.5	6807.0	233.9	0.967	1.110	0.848	1.134
			100	6620.3	178.9	6688.6	238.3	6640.6	161.3	6672.0	222.7	1.601	1.934	1.929	2.377

表 B.4　演算法在大型案例之求解結果(4)

n	m	p	ep	Makespan								Time			
				ITS		EXTS1		EXTS2		TS2		ITS	EXTS1	EXTS2	TS2
				mean	Std.	mean	Std.	mean	Std.	mean	Std.				
100	5	20	50	1225.2	46.4	1278.3	51.3	1259.3	41.4	1264.0	42.3	0.655	0.526	0.447	0.503
			100	1215.4	54.3	1253.7	46.2	1247.7	47.7	1240.6	38.6	1.031	0.894	0.793	1.106
		50	50	2981.2	111.6	3065.2	112.5	3059.3	119.0	3114.9	131.5	0.565	0.547	0.460	0.506
			100	2965.7	117.2	3063.4	86.5	3058.9	122.2	3112.0	116.3	0.818	0.844	0.793	0.966
		100	50	5868.4	232.7	6114.0	309.7	6139.0	171.6	6184.5	210.4	0.954	0.489	0.465	0.587
			100	5811.8	209.6	6059.7	276.0	6137.5	262.4	6106.7	230.7	1.156	1.015	0.717	0.869
	10	20	50	1604.4	49.7	1608.5	59.6	1621.5	47.3	1617.6	49.2	0.939	1.063	0.913	0.955
			100	1586.1	46.1	1607.4	48.5	1618.9	55.0	1616.9	34.9	1.794	1.870	1.704	1.965
		50	50	3933.8	121.1	3985.0	107.5	4008.4	112.8	4001.0	105.4	0.916	1.049	0.913	1.038
			100	3878.4	120.0	3984.8	116.8	4006.3	111.2	3962.5	118.6	1.910	1.790	1.694	1.801
		100	50	7845.2	211.4	7957.6	213.3	7915.4	206.3	7985.3	188.9	0.804	1.066	0.968	1.035
			100	7735.0	239.2	7920.4	214.7	7863.8	202.7	7811.6	298.6	1.928	2.945	1.776	1.875
	15	20	50	1925.9	38.4	1990.2	52.8	1970.1	53.4	1970.7	57.6	1.313	1.787	1.535	1.490
			100	1913.3	57.6	1976.3	55.6	1962.9	51.8	1968.8	59.5	2.543	3.331	2.682	3.139
		50	50	4771.5	104.7	4872.1	98.5	4868.3	94.5	4819.4	142.8	1.391	1.947	1.633	1.540
			100	4719.5	106.3	4858.0	117.7	4849.2	160.3	4803.8	151.4	2.336	3.194	2.735	2.865
		100	50	9459.2	170.8	9772.2	313.5	9700.0	214.0	9653.7	243.6	1.254	1.754	1.622	1.474
			100	9369.5	232.2	9655.5	213.7	9599.6	181.1	9684.0	247.5	2.749	3.098	2.716	2.576
	25	20	50	2520.0	46.1	2615.7	49.8	2594.7	57.6	2587.8	42.1	2.629	3.218	3.211	3.077
			100	2520.7	50.8	2609.1	56.3	2570.0	45.1	2573.8	47.5	5.192	6.169	5.501	5.951
		50	50	6183.0	141.9	6406.9	138.2	6365.8	165.7	6371.2	125.7	2.323	6.023	3.398	3.087
			100	6173.2	138.7	6393.4	119.0	6329.8	159.4	6305.9	129.6	4.796	3.456	6.026	5.902
		100	50	12328.1	227.7	12903.0	298.3	12755.0	226.7	12622.7	196.8	2.542	3.549	2.986	2.806
			100	12260.2	231.3	12708.0	251.0	12687.1	219.1	12576.4	251.3	4.636	7.278	5.599	5.011

表 B.5　演算法在大型案例之求解結果(5)

n	m	p	ep	Makespan								Time			
				ITS		EXTS1		EXTS2		TS2		ITS	EXTS1	EXTS2	TS2
				mean	Std.	mean	Std.	mean	Std.	mean	Std.				
200	5	20	50	2395.1	62.3	2485.4	57.4	2456.9	64.8	2469.8	48.1	2.473	2.852	2.146	1.801
			100	2381.0	67.3	2456.2	67.9	2441.0	65.7	2454.1	55.3	4.566	3.657	3.438	3.577
		50	50	5898.4	133.9	6107.1	190.3	6123.5	179.9	6036.1	139.7	2.107	2.338	1.945	1.651
			100	5867.6	168.8	6065.2	181.7	6033.5	161.0	6031.8	167.3	3.569	4.330	3.065	2.754
		100	50	11640.1	340.7	12212.0	287.1	12037.2	265.5	12042.7	304.7	2.110	2.248	1.860	1.176
			100	11628.7	304.4	12160.9	406.6	11996.5	369.2	12022.2	285.5	4.298	4.072	3.021	2.228
	10	20	50	3042.1	58.6	3173.4	89.6	3139.6	82.7	3108.9	52.0	2.863	4.561	4.322	2.168
			100	3038.5	40.8	3165.9	76.2	3136.4	73.9	3093.0	53.6	5.392	7.577	7.159	3.761
		50	50	7525.8	124.1	7790.4	167.7	7712.5	155.4	7618.7	122.4	2.736	4.422	4.201	2.131
			100	7471.1	155.0	7787.0	158.4	7675.5	163.6	7570.2	155.3	4.767	7.176	7.858	4.225
		100	50	14963.2	317.0	15521.3	296.6	15460.2	368.2	15209.2	290.5	2.566	4.311	3.887	1.946
			100	14797.9	281.0	15486.3	324.9	15452.6	360.4	15126.5	256.3	4.679	7.101	6.464	3.892
	15	20	50	3646.0	51.0	3846.0	88.5	3791.0	74.3	3708.4	63.1	4.122	6.364	6.158	4.514
			100	3629.9	63.8	3807.9	85.0	3786.4	64.6	3691.4	53.4	7.527	12.386	12.689	4.514
		50	50	8910.3	133.3	9475.3	158.7	9347.3	185.6	9107.7	131.3	4.000	7.072	7.116	3.969
			100	8905.7	156.7	9415.0	168.0	9331.6	191.0	9039.1	137.1	7.791	11.968	12.920	7.407
		100	50	17692.0	306.0	18903.8	292.7	18708.9	339.0	18137.9	266.6	3.998	6.559	6.864	3.854
			100	17675.6	225.3	18811.8	344.8	18477.8	334.7	18059.3	249.1	6.949	12.163	12.984	7.316
	25	20	50	4653.7	84.2	4977.5	85.2	4964.9	76.9	4777.9	64.0	7.489	14.853	13.483	9.022
			100	4641.0	66.3	4974.4	72.2	4949.3	69.2	4756.9	78.9	12.821	24.747	22.960	15.269
		50	50	11401.7	150.1	12401.6	195.3	12290.8	177.5	11692.3	150.2	6.507	14.527	13.521	7.607
			100	11380.5	177.5	12350.0	177.9	12200.7	155.8	11688.7	142.4	12.858	26.949	22.349	14.731
		100	50	22738.7	222.8	24569.7	320.6	24370.7	460.9	23361.2	295.6	6.431	15.272	14.723	9.276
			100	22672.7	250.3	24486.2	443.0	24345.8	390.3	23263.9	333.4	13.052	27.637	29.222	18.621