國立交通大學

工業工程與管理學系

碩士論文

混合式流程工廠排程之巨集啟發式演算法的比較

A Comparison of Meta-heuristics Algorithms for the Hybrid Flow Shop Scheduling Problem

研究生:李忠霖

指導教授：巫木誠 博士

中華民國九十六年六月

混合式流程工廠排程之巨集啟發式演算法的比較

A Comparison of Meta-heuristics Algorithms for the Hybrid Flow Shop Scheduling Problem

研究生：李忠霖　　Student： Zhong-Lin Li

指導教授：巫木誠 博士　　Advisor： Dr. Muh-Cherng Wu

國立交通大學

工業工程與管理學系

碩士論文

A Thesis

Submitted to Department of Industrial Engineering and Management

College of Management

National Chiao Tung University

In Partial Fulfillment of the Requirements

For the Degree of Master of Science

In

Industrial Engineering

June 2007

Hsin-Chu, Taiwan, Republic of China

中華民國九十六年六月

混合式流程工廠排程之巨集啟發式演算法的比較

研究生：李忠霖　　指導教授：巫木誠 博士

國立交通大學工業工程與管理研究所

中文摘要

在流程工廠排程問題(Flow shop scheduling；FS)中，若一個工作站有多部機台，而且一個作業可能需要多部機台合作才能加工，此種問題稱之為混合式流程工廠排程問題 (Hybrid flow shop scheduling problem；HFSP)。針對 HFSP 問題，過去已有許多演算法，目前以 Ying & Lin (2006)的演算法表現較佳，本研究結合共識因子和田口方法提出多種巨集演算法(meta-heuristics)，希望找出一種演算法，能在績效上改進 Ying & Lin (2006)的演算法。本研究使用 120 個案例，分成 12 種情境實驗測試。與 Ying & Lin (2006)演算法相比，本研究所發展的演算法僅在 42%的情境表現較佳，卻在 58%的情境中表現較不如過去的演算法。

關鍵字：混合式流程工廠、排程、演算法

A Comparison of Meta-heuristics Algorithms for

the Hybrid Flow Shop Scheduling Problem

Student：Zhong-Lin Li Advisor：Dr. Muh-Cherng Wu

Institute of Industrial Engineering

National Chiao Tung University

ABSTRACT

A hybrid flow-shop scheduling problem (HFSP) has two distinct features. A workstation may include more than one machine; and an operation may require more than one machine to process it. Much literature on HFSP has been published, in which the algorithm proposed by Ying & Lin (2006) is the most leading one. We applied the notions of consensus and Taguchi genetic operators and proposed various meta-heuristics algorithms. Extensive numerical tests that include 120 problem instances, categorized into 12 scenarios, have been carried out. Compared with the algorithm proposed by Ying & Lin (2006), our algorithm excel in 42% scenarios and lose in 58% scenarios.

Keywords: Hybird flow-shop; Scheduling; Algorithms

誌謝

本論文能夠順利完成，首先要感謝恩師 巫木誠教授悉心的教導，老師敞開的大門讓我的創意與想法有了揮灑的空間，也適時的在關鍵之處給我提點，讓學生我在研究所兩年的求學過程中，不論是研究知識或待人處事方面都有相當長足的進步，也是未來人生旅途上寶貴的資產，在此致上最崇高的敬意。同時感謝許錫美教授、彭德保教授、陳文智教授，在論文口試時所給予的寶貴建議，讓本論文更臻完備。

感謝博士班學長蘇泰盛、施昌甫的經驗分享與引領，學長吳政翰的研究指導，還有研究室同學與學弟妹們無時無刻的加油打氣與噓寒問暖，讓我的研究生涯更加豐富與充實。並感謝摯友兼研究夥伴振富、進銘、富騰、光棓，不論是在研究的互相學習過程，或者生活的陪伴，得到了相當多的助益與關懷。此外，感謝在我遇到問題時，所有幫助過我的朋友們。

最後，要特別感謝我的父母與兩個姐姐，由於你們在背後的支持與鼓勵，使我沒有後顧之憂，得以專心完成學業，你們是支撐我最大的支柱，願將此刻的喜悅與榮耀和你們一同分享，謝謝你們。

忠霖　于風城交大

中華民國九十六年六月

目錄

表目錄

圖目錄

第一章　緒論

1.1 研究背景

現場作業排程是生產規劃重要工作之一，主要內容決定各項作業的加工機台與優先順序，以期各類資源做最適當的配置，進而達成特定衡量準則最佳化目的。良好的排程不僅可以提昇生產的效率與服務品質，更可以降低生產成本，強化企業的競爭能力，以提高企業的獲利能力。若依照各作業的加工性質來分類，生產排程可分為流程工廠排程(Flow Shop)、零工工廠排程(Job Shop)、開放工廠排程(Open Shop)等等。

在傳統的流程工廠中，如圖 1.1 所示，每項作業的加工途程為固定，而每個工作站都僅有一台機器。至於流程工廠排程，最主要的決策即是工件排序(Job sequence)，因為排序的不同會影響排程績效。如圖 1.2，原本Machine 1 的排序為 $J_1 \rightarrow J_2 \rightarrow J_3$，若將$J_2$和$J_3$的順序互換之後，總完工時間將從 21 減為 17，由此我們可以得知，好的排序會產生較佳的時間績效結果。

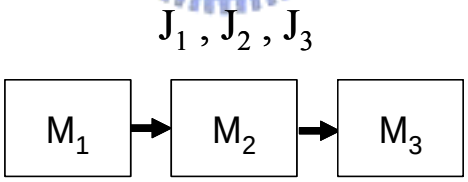

圖 1.1 加工途程固定

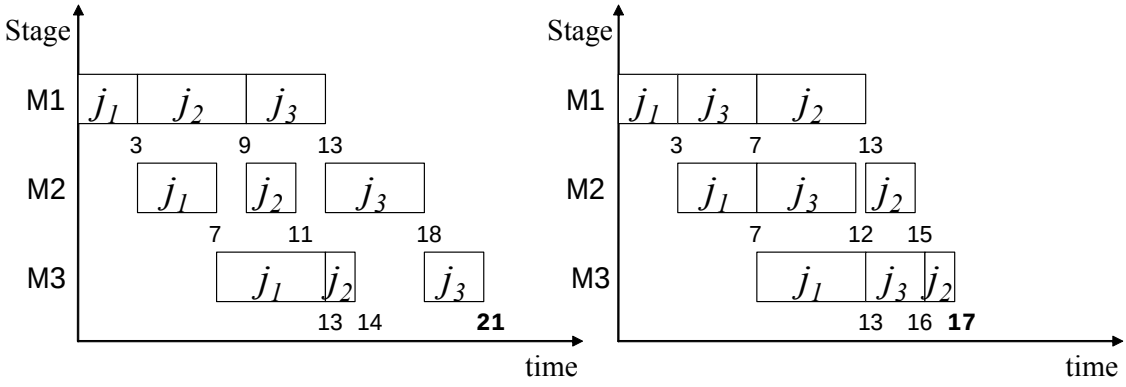

圖 1.2 工件排序明顯影響績效

若我們將平行機台概念與流程工廠加以結合，如圖 1.3 所示，意即至少有一工作站擁有兩台以上的平行機台，就是所謂的混合式流程工廠(Hybrid Flow Shop Problem；HFSP)，由於混合式流程工廠之生產排程較流程工廠更有彈性，因而於實務上被廣泛地應用在電子、鋼鐵、紡織、塑化等產業。

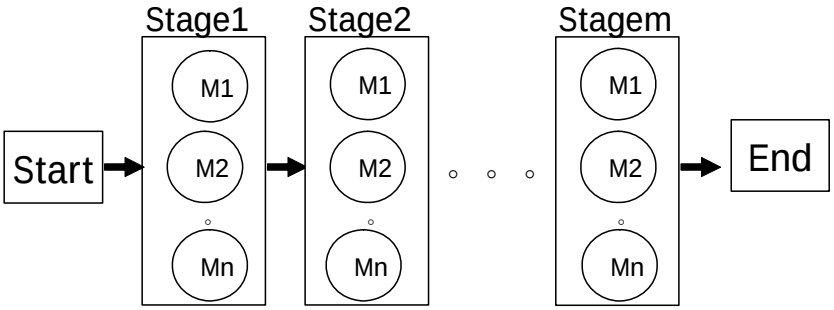

圖 1.3 平行機台概念

HFSP 其包含以下作業特性：(1)Each stage：每個 stage 擁有多部平行機台，且其為非批次機台，此外每個資源在同一時間只能加工一個作業而且不能被中斷(non-preemptive)，另外資源必須將工作加工完成才能釋放。(2)Job/Operation：某些作業同時須多部機台以進行合作加工，如表 1.1 所示；不同工件其加工途程皆固定，而每個工件在各工作站的加工時間為固定且事先已知。(3)Different jobs：作業間彼此獨立。

表 1.1 2 個階段 3 個機器的工件參數

Job j	Job j in stage i (O_{ij})	Processing time (T_{ij})	Processor requirement (S_{ij})
J_1	O_{11} O_{21}	$T_{11}=5$ $T_{21}=4$	$S_{11}=2$ $S_{21}=1$
J_2	O_{12} O_{22}	$T_{12}=3$ $T_{22}=1$	$S_{12}=3$ $S_{22}=1$
J_3	O_{13} O_{23}	$T_{13}=2$ $T_{23}=4$	$S_{13}=3$ $S_{23}=2$

鑒於混合式流程工廠在產業的大量應用，因此近年來不斷吸引許多學者對其相關的問題進行研究，求解 HFSP 的方法可以概分為派工法則(Dispatch rule)、分枝定界法(Branch-and-bound)、最佳化演算法(optimization algorithm)以及近似演算法(approximation algorithm)等方法。在這些方法中，以派工法則最為簡單易懂，其計算時間最短，但求解品質較差；分枝定界法和最佳化演算法的求解品質佳，但因為需要花費大量的計算時間，較適用於計算小型的題目，在實務上並不符合企業經營的成本效益及需求；近似演算法相形之下就可以在較短的時間內，產生一個令人滿意的近似最佳解。目前已知用於求解 HFSP 問題的近似演算法包括：塔布搜尋法(Tabu search)(Oguz et al., 2004)、基因演算法(Genetic algorithm)(Serifoglu & Ulusoy, 2004)、螞蟻演算法(Ant colony system)(Ying & Lin, 2006)等。

1.2 研究問題

本研究所探討HFSP的定義如下：假設有n個工件(jobs)，每個工件要依序經過m個工作站(stage)進行加工；每個工件必須於第j個工作站加工完成後，才可以往第j+1 個工作站進行加工；各工作站均含Pm_j($j = 1,...,m$)個齊一平行機器(parallel and identical machine)的資源；對於第i個工件在工作站j的加工時間及使用資源分別以$p_{i,j}$及$size_{i,j}$表示；每一個資源在同一時間只能加工一個工件而且不能被中斷(non-preemptive)；資源必須將工件加工完成才能釋放；求解如何排程各項工件的加工順序，使總完工時間($C_{\max}$)達到最小。

1.3 假設與限制條件

本研究所探討的 HFSP 之限制條件及相關假設如下：

(1) 靜態模式，不考慮各工件的抵達時間(arrival time)。

(2) 不考慮到期日(due date)。

(3) 不考慮各機器的整備時間(setup time)。

(4) 每個工件的待命時間(ready time)為 0。

(5) 工作站之間的緩衝(buffer)容量無限制。

(6) 工件數(n)為固定且事先已知。

(7) 工作站數(m)為固定且事先已知。

(8) 每個工作站的資源數(Pm_j)為固定且事先已知。

(9) 每個工件在各工作站的加工時間($p_{i,j}$)為固定且事先已知。

(10)每個工件在各工作站的使用資源($size_{i,j}$)為固定且事先已知。

1.4 研究目的

對於混合式流程工廠排程，過去已經有相當多的文獻致力於此問題，但是由於問題解空間相當的大，至今仍未有效找到真正最佳解，因此之於解的品質仍然有成長的空間存在。本研究利用工件排序為決策，以最大完工時間(makespan)最小化為績效指標，應用共識因子、田口方法與標竿演算法：螞蟻系統演算法等方法，來進行有系統的組合以及測試實驗，以期確認是否可以發展出較佳的演算法，並作系統化的優缺點分析。

1.5 論文章節安排

本論文後續章節安排如下：第二章為相關文獻回顧，回顧 HFSP 及解題方法相關文獻。第三章敘述研究方法，其中包含演算法的整體架構介紹，以及相關的演算法組合與實驗分配，最後再就解的表達與演算法作詳述。第四章則是實驗的結果與比較，還有組合方法優缺點的分析。第五章為結論與未來的一些可行研究方向。

第二章　文獻回顧

2.1 混合式流程工廠排程問題

HFSP 最早是由 Arthanari & Ramamurthy(1971)兩位學者所提出。在排程問題中，大多數都屬於 NP-hard 的問題，混合式流程工廠排程問題也一樣，即使是最簡單的 HFSP，例如 Gupta(1988)所研究的兩階段 HFSP，第一階段只有一台機器，第二階段有兩台平行齊一(parallel and identical)機器的情況，也是 NP-hard 問題。

關於 HFSP 的類型，根據 Richard & Wei(1999)兩位學者的分類，可分為三類：(1)兩階段 HFSP(Two-stage HFSP)，(2)三階段 HFSP(Three-stage HFSP)，(3)k 階段 HFSP(m>3) (m-stage HFSP)，目前對於 HFSP 的研究，大部分僅針對兩階段的 HFSP，對於三階段的 HFSP 或 k 階段的 HFSP 進行探討則相對較少。

2.2 HFSP 相關解題方法

針對一個 NP-hard 問題，要找到一個好的解題方法是有相當難度的。而利用派工法則為基礎所設計的演算法則是發展行之多年。例如，Gupta & Tunc (1991)所提出的演算法來自於 LPT 法則的靈感，將工作依第二站加工時間以 LPT 排出順序，再將第一站中加工時間最短的工作移至最前，以減少第二站的機器閒置等待時間。Guinet et al. (1996)提出的啟發式演算法，排序的階段所用的法則為 SPT 和 LPT。Viger et al. (1996)利用 EDD 法則的概念來求解 HFSP 最大延遲時間最小化問題。一個好的派工法則通常都能夠作為啟發式演算法的起始解，用以求解更大型且情況更複雜的問題。

當要求更精確的解時，就必須用更複雜的方法來求解，分枝定界法(Branch and bound；B&B)就是一個常見的選擇。分枝定界法需要考慮到下界、分枝以及節點的刪除，而求得邊界以及刪除節點的法則需要比派工法則更多的計算時間與

空間。Brah & Hunsucker (1991)、Rajendran & Chaudhuri (1992)提供了許多不同的B&B 法來求解多階段 HFSP 的最佳解。

上述解題方法在解題品質或者速度上皆有某些方面的瑕疵，無法在實務上能有高效率的表現，因此為了在有限的時間及空間中求得最佳解，導致大量的研究投入於近似演算法。近似演算法皆針對總完工時間最小為績效指標，包含如下：Sivrikaya-Serifoglu & Tiryaki (2002)利用模擬退火法(Simulated annealing)；Oguz et al. (2004) 使用塔布搜尋法(Tabu search)求解；Serifoglu & Ulusoy (2004)以基因演算法(Genetic algorithm)求解；Ying & Lin (2006)以螞蟻演算法(Ant colony system)求解；而其中尤以 Ying & Lin (2006)表現最佳。

2.3 測試資料庫及下界

為了測試本研究所提出演算法的組合與標竿演算法對於績效指標的結果，利用 Oguz et al. (2004)所提供的標準題庫來進行檢驗。問題的組合依據工作件數(n = 5,10,20,50)及工作站數(m = 2,5,8)所組成共 12 種，每個組合有 10 個測試題目，資料庫可由 Oguz 個人網站下載 (http://www.acad.polyu.edu.hk/~lgtceyde/)。另外由於每篇文獻演算法用於跑程式使用電腦規格不盡相同，為了求實驗出發點的公平，本研究採用演算法計算過程產生相同的解個數來作為停止條件，就是在同樣的解個數產生過程內進行解的品質比較。

Serifoglu & Ulusoy (2004)兩位學者為了資料庫的測試標準，提出了下界(lower Bound；LB)的計算公式，公式如下：

$$LB = \max_{i \in M} \left\{ \min_{j \in J} \left\{ \sum_{k=1}^{i-1} p_{k,j} \right\} + \frac{1}{p_{m_i}} \sum_{j \in J} p_{i,j} \times size_{i,j} + \min_{j \in J} \left\{ \sum_{k=i+1}^{m} p_{k,j} \right\} \right\}$$

而績效的衡量則是利用所求得之最小完工時間(C_{max})與下界(LB)差距的百分比

($\frac{C_{\max} - LB}{LB} \times 100\%$)作為比較來求得。

第三章　研究方法

本研究希望能在相關的演算法組合內找到較佳的求解結果與配方。本章首先是對演算法整體架構(Generic Framework)的介紹，其中包含演算法之間實驗的組合與分配，然後則是對於實際實驗的詳細進行作解說，最後在就架構內各模組演算法作完整的詳述。

3.1 演算法整體架構的介紹

本研究將演算法架構主要分成四個模組，包含了初始解產生器、兩個更新解產生器與更新解修正器，如圖 3.1。

在初始解產生器部份，包括了(1)Random[A1]和 (2)ACO + Local search[A2]兩種產生方法，此部份是為了產生多組解，以供後續更新解產生器憑藉來進行解的有效更新與產生。

更新解產生器分為兩個部份，產生器 1 包含了 consensus operator，其中共識因子又區分為三個參數組合種類，如表 3.1 所示，包含了：(1)世代產生解數目：100 個[B1]和 1000 個[B2]，用以區分每個世代產生解的數量是否會影響共識因子建構共識矩陣，進一步影響解的品質；(2)Weighting：分為 no weighting[B3]、linear weighting[B4]以及 power weighting[B5]三種，來分辨是否要依據解的品質好壞來給予解所代表的權重因素；(3)Rate：將共識比例乘上一個調整參數 r 值，用來更改共識比例的大小，以增加演算法的求解彈性，其 r 值也分為三種，第一種 r = 1[B6]，第二種為了減少共識比例的主導性，設定 r = 0.8[B7]，第三種為了增加演算法共識比例的多元化，當共識比例 $p > 0.5$ 時，$r = 0.8$；$p < 0.5$ 時，$r = 1.2$[B8]。產生器 2 包含了(1)無和(2)Taguchi methods[C1]。另外，在更新解產生比例部分，就 consensus operator 與 Taguchi methods 解數目的比例，我們採用 9:1 進行產生。

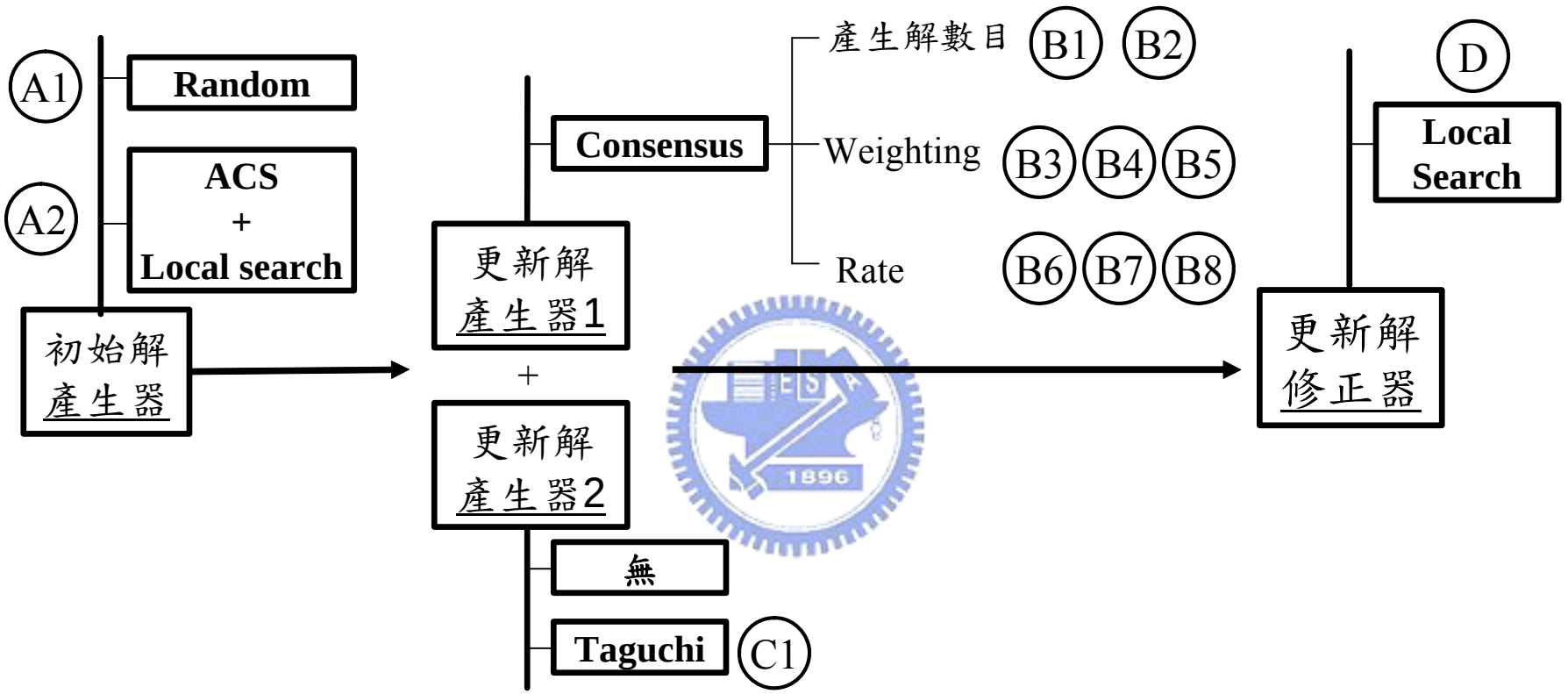

圖 3.1 General Framework

表 3.1 共識因子參數組合

編號	種類
產生解數目	
B1	100
B2	1000
Weighting	
B3	no weighting
B4	linear weighting
B5	power weighting
Rate	
B6	r = 0.1
B7	r = 0.8
B8	r = 0.8 or r = 1.2

更新解修正器則只有鄰近搜尋法(Local search) [D1]，用以將從前面更新解產生器得到的解，利用範圍性的區域搜尋，企圖找到較佳的解。而在投入更新解修正器的解數目，我們採用 100 個來自更新解產生器最佳的解作為修改的標的。

本研究主要的求解過程，就如上面所述，由一開始的初始解產生器→更新解產生器→更新解修正器，逐步來進行解的搜尋。另外由於各個模組的演算法相當多，因此將上述提到個各個演算方法進行有效的組合，最後產生多個用以求解的 version，然後逐一將每個 version 與標竿演算法一起進行程式的驗證，希望能從中找到有更佳績效的演算法組合(version)。

而本研究的標竿演算法組合(V0)如圖 3.2 所示，其架構具有世代循環的觀念，架構分為兩個部分：首先，在螞蟻演算法利用探索與追隨的概念來產生較好的解；而後 Local search 則是對每個世代最佳解利用微調方式，在區域內進行範圍型的搜尋；最後再利用每個世代的最佳解進行螞蟻演算法內的費洛蒙更新機制，用來增強最佳解的重要程度，也就是有世代循環的概念存在。

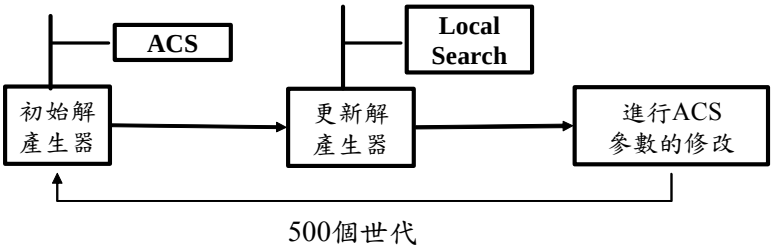

圖 3.2 標竿演算法架構

3.2 演算法實驗的組合與分配

本研究期望從上一節提到的 general framework 裡面的眾多方法中，找到最佳演算組合，所以在本節當中，將上述提到的方法與以有效的組合與分類，以供在後續實例研究當中，能有完整的 version 展示。

實驗組合如表 3.2 所示，其中初始解產生器兩種方法：Random[A1]和 ACO + Local search[A2]；更新解產生器 1 的共識因子有世代產生解數目(2 種)、Weighting(3 種)、Rate(3 種)三種主要種類，而進行配對之後則有參數組合 2×3×3 = 18 種。更新解產生器 2 有兩種情形：無與 Taguchi methods。更新解修正器單只一種。最後可以得到本研究的實驗組合會有 2×18×2 =72 個 version 數目，也就是後續實例驗證要實驗的組合數目。

表 3.2 演算法實驗組合

初始解	更新器1		更新器2	修正器
A1	V1：B1 & B3 & B6	V11：B2 & B3 & B7	C1	D1
	V2：B1 & B3 & B7	V12：B2 & B3 & B8		
	V3：B1 & B3 & B8	V13：B2 & B4 & B6		
	V4：B1 & B4 & B6	V14：B2 & B4 & B7		
	V5：B1 & B4 & B7	V15：B2 & B4 & B8		
	V6：B1 & B4 & B8	V16：B2 & B5 & B6		
	V7：B1 & B5 & B6	V17：B2 & B5 & B7		
	V8：B1 & B5 & B7	V18：B2 & B5 & B8		
	V9：B1 & B5 & B8			
	V10：B2 & B3 & B6			
A1	V19：B1 & B3 & B6	V29：B2 & B3 & B7	無	
	V20：B1 & B3 & B7	V30：B2 & B3 & B8		
	V21：B1 & B3 & B8	V31：B2 & B4 & B6		
	V22：B1 & B4 & B6	V32：B2 & B4 & B7		
	V23：B1 & B4 & B7	V33：B2 & B4 & B8		
	V24：B1 & B4 & B8	V34：B2 & B5 & B6		
	V25：B1 & B5 & B6	V35：B2 & B5 & B7		
	V26：B1 & B5 & B7	V36：B2 & B5 & B8		
	V27：B1 & B5 & B8			
	V28：B2 & B3 & B6			
A2	V37：B1 & B3 & B6	V47：B2 & B3 & B7	C1	D1
	V38：B1 & B3 & B7	V48：B2 & B3 & B8		
	V39：B1 & B3 & B8	V49：B2 & B4 & B6		
	V40：B1 & B4 & B6	V50：B2 & B4 & B7		
	V41：B1 & B4 & B7	V51：B2 & B4 & B8		
	V42：B1 & B4 & B8	V52：B2 & B5 & B6		
	V43：B1 & B5 & B6	V53：B2 & B5 & B7		
	V44：B1 & B5 & B7	V54：B2 & B5 & B8		
	V45：B1 & B5 & B8			
	V46：B2 & B3 & B6			
A2	V55：B1 & B3 & B6	V65：B2 & B3 & B7	無	
	V56：B1 & B3 & B7	V66：B2 & B3 & B8		
	V57：B1 & B3 & B8	V67：B2 & B4 & B6		
	V58：B1 & B4 & B6	V68：B2 & B4 & B7		
	V59：B1 & B4 & B7	V69：B2 & B4 & B8		
	V60：B1 & B4 & B8	V70：B2 & B5 & B6		
	V61：B1 & B5 & B6	V71：B2 & B5 & B7		
	V62：B1 & B5 & B7	V72：B2 & B5 & B8		
	V63：B1 & B5 & B8			
	V64：B2 & B3 & B6			

3.2.1 Pilot study(引導實驗)

若實驗組合裡 72 個 version 全部都要跑完總共 12(3×4)個問題情境組合(scenario),將會需要耗費相當大量的時間,因此先就每個 version 跑過 pilot study,而在 pilot study 內每個 version 則進行[n = 20]、[m = 5]的實驗測試。然後希望在上述每個模組的 pilot study 數據結果上,逐步進行實驗與分析,分辨演算法之間的優劣,由績效來做出取捨,以找出較佳績效的演算法組合,對於研究結果進行分析,進而再從 pilot study 較佳績效的 version 進行 comprehensive study 實驗。

3.2.2 Comprehensive study (整合實驗)

從 pilot study 分析出來的較佳 version,進行全數問題情境組合的程式運算,也就是要跑完 12 個 scenario,然後就最後的結果,找出最佳的演算法配方組合,然後進行實驗結果的分析,對其優缺點作討論,然後就 pilot study 與 comprehensive study 結果作有效整合,並對整個演算法架構提出可能的改善建議。

3.3 演算法模組

本研究演算法模組主要分為兩個部分,如圖 3.3 所示,包含解的表達法與演算法結構兩個部分。在演算法結構部分就初始解產生器、更新解產生器、更新解修正器等模組內的研究方法逐一作介紹。

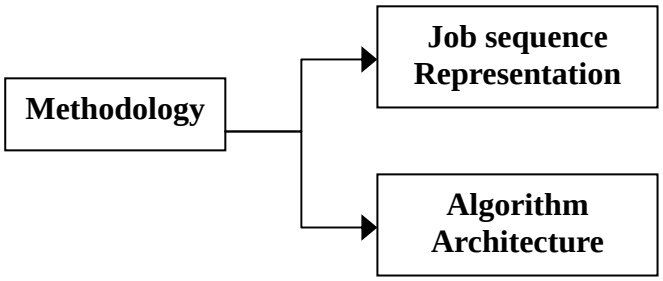

圖 3.3 演算法模組

3.3.1 解的表達法

在stage i 所有工件的排序(Job sequence)，以$X(i)=[J_1;J_2;\ldots;J_m]$表示，$X$代表一組解，$m$代表工件的個數。釋例：若stage 1 的工件順序$J_1 \rightarrow J_5 \rightarrow J_3 \rightarrow J_2 \rightarrow J_4$，則可表示為$X(1)= [J_1; J_5; J_3; J_2; J_4]$ 。

3.3.2 Makespan 的計算

在得到了一組排序解(stage 1 的加工順序)，以及原有給定的工件途程，就可利用 FCFS (First come , first served)法來進行 stage 2 到 stage m 的排程。FCFS 其乃遵循(1)最早可開始加工及(2)有足夠資源兩個法則，作為決定各工作站指派工作順序之依據。如圖 3.4 所示最後可得到一組完整排程以及總完工時間(Makespan)。

圖 3.4 Makespan 的計算

3.3.3 初始解產生器

初始解產生器本研究利用隨機方法(Random)與螞蟻演算法(Ant colony system)+鄰近搜尋法(Local search)兩種。鄰近搜尋法由於與後續要介紹的更新解修正器方法相同，所以留待 3.3.5 章節再作介紹。

(一) 隨機方法

單純利用隨機方式產生一組工件排序解，隨機產生的優點是快速大量產生且方便，缺點則是解的品質容易較不盡理想。

(二) 螞蟻演算法

螞蟻演算法是由 Dorigo (1992)提出的巨集式啟發演算法，主要靈感系來自真實螞蟻之群體合作覓食行為。當尋找食物時，螞蟻會分泌一種稱為費洛蒙(pheromone)的化學物質於行經的路徑上，較後行的螞蟻則依據各路徑上先行螞蟻分泌之費洛蒙累積量的多寡來選取欲行走的路徑，若某路徑上的累積費洛蒙量愈大，則會給予後行的螞蟻愈大的刺激，並導致較大的跟蹤機率。由於經由較短路徑上的蟻群流量較大，因此其分泌之費洛蒙量累積的速度將比較長路徑上累積速度為快，始得蟻群行走路徑朝向蟻窩與食物之間的最短路徑進行收斂。

螞蟻演算法首先對演算法作初始化，包括問題的定義和各個參數的設定等，其中參數有螞蟻的個數(k)、局部費洛蒙揮發比例(ρ)和全域費洛蒙揮發比例(α)，還有控制要選擇「利用」(exploitation)還是「探索」(exploration)的相對比例參數(q_0)。

1.狀態轉移法(State transition rule)

演算法中，要決定J_i下一個要執行的工件J_j，必須依照公式(1)的狀態轉移法則：

$$j=\begin{cases}\arg\max\limits_{u\in J(i)}\left\{[\tau(i,u)]\times[\eta(i,u)]^{\beta}\right\} & \text{if } q\le q_0\\ J & \text{otherwise}\end{cases} \qquad (1)$$

其中τ代表J_i→J_j工件順序的費洛蒙濃度，η代表貪婪法則的結果，乃是螞蟻對下一個工件j的偏好(heuristic desirability)，$J(i)$代表尚未加工工件的集合，另外q是一個介於 0 和 1 之間的隨機變數，q_0則是一個參數($0\le q_0\le 1$)。而J也是一個隨機變數，由公式(2)的隨機比例法則(random-proportional rule)：

$$\text{轉移機率}P_{ij}=\begin{cases}\dfrac{[\tau(i,j)]\times[\eta(i,j)]^{\beta}}{\sum\limits_{u\in J(i)}[\tau(i,j)]\times[\eta(i,j)]^{\beta}} & \text{若 } j \text{ 屬於 } J(i)\\ \text{otherwise}\end{cases} \qquad (2)$$

所產生的轉移機率分配來抽出。P_{ij}指的是加工到J_i時，下一個job會選擇尚未加工工件集合中job j的機率。

所以當一隻螞蟻從工件J_i要選擇下一個執行工件J_i時，要先產生一個隨機亂數q，若$q \leq q_0$，則利用公式(1)從$J(i)$挑選出最佳的工件；反之，則按照公式(2)產生的機率分配來隨機選取工件。

貪婪法則則是依據每種問題，一開始以五種派工法則求其 makespan，取最小者作為貪婪法則所採用的方法。這五種派工法則如下所示：(1)最短加工時間(SPT)、(2)最末站最短加工時間(BPT)、(3)修改 CDS(Campbell et al., 1970)演算法、(4)修改 Raghu and Rajendran rule(RR)(Raghu & Rajendran, 1993)、(5)修改 Palmer's rule(Palmer, 1965)。決定所選擇的最適派工法則之後，將該派工法則所求得的 makespan 設定為目前所得到的最佳解。決定貪婪法則所使用之派工法則後，將派工法則的值利用正規化為介於 0 與 1 之間的值，進而得到每個工件的η值。

2.局部更新法(Local updating rule)

當一隻螞蟻建立一個完整解之後，我們要依照局部更新法，將它所經路徑的費洛蒙進行更新，局部更新法過程使拜訪過的工件順序上費洛蒙減少，因而對螞蟻吸引力減小，目的是為了要使之後的螞蟻能夠找到更多不一樣的解，進而避免產生過於強勢路徑，使螞蟻無法進行探索動作，而導致收斂局部最佳解。局部更新法公式(3)如下：

$$\tau(i,j) = (1-\rho)\times\tau(i,j) + \rho\times\tau_0 \qquad (3)$$

其中，$0<\rho<1$，為一個控制費洛蒙揮發比例的參數，τ_0為初始費洛蒙濃度，設定為(LB^{-1})。

3.全域更新法(Global updating rule)

當所有螞蟻都已經建構完成各自的解之後，找出至今最佳解來進行全域的費洛蒙更新，全域更新法公式(4)如下：

$$\tau(i,j) = (1-\alpha)\times\tau(i,j) + \alpha\times\Delta\tau(i,j) \qquad (4)$$

$$\Delta\tau(i,j) = \begin{cases} \dfrac{m}{C_{best}} & \text{若 } i \text{、} j \text{ 屬於至今最佳解的工件排序} \\ 0 & \text{otherwise} \end{cases}$$

而其觀念是只有目前表現最佳的解其內的工件執行順序才具有遺留費洛蒙的權力，以使得下個世代中，螞蟻能再次利用相類似的排序；而另外所有工件執行順序的費洛蒙濃度皆會揮發。到這裡為止稱作一個世代，當世代的次數尚未達到預先設定的數目，則繼續演算法搜尋最佳解；反之，則結束演算法。

4.求解流程

雖然僅針對第一站的工件排序，但是本研究在螞蟻演算法求解過程中使用了F&B-ACS，同時採用了 forward 及 backward 兩個方向的搜尋，所以紀錄費洛蒙量的矩陣需要兩個，每個矩陣的大小為 $n\times n$，其中元素 τ_{ij} 代表的是第 j 個工件排在第 i 個工件後的費洛蒙量。

3.3.4 更新解產生器

更新解修正器包含兩種演算法：共識因子(consensus operator)與田口方法(Taguchi methods)。

(一) 共識因子

對於產生新工件排序的排程問題，一般傳統的方法像基因演算法採用的演化技術(evolutionary techniques)例如:單點交配(one-cut crossover)，通常採用隨機兩個染色體區段交配，交配的過程中並不考慮上一代好的基因，而盲目搜尋(blind search)， 甚至會有不合理解。我們主要發展這共識因子(consensus operator)，希望可以根據上一代好的工件排序中尋找資訊，建立一個導引(guided)機制，根據這引導機制讓好的優先順序保留，不好的優先順序則自然淘汰，以產生好的工件

排序，敘述如下。

共識因子產生方式主要分為兩個步驟：

Step 1: 建立引導機制：共識矩陣 Consensus_matrix(如圖 3.5 所示)

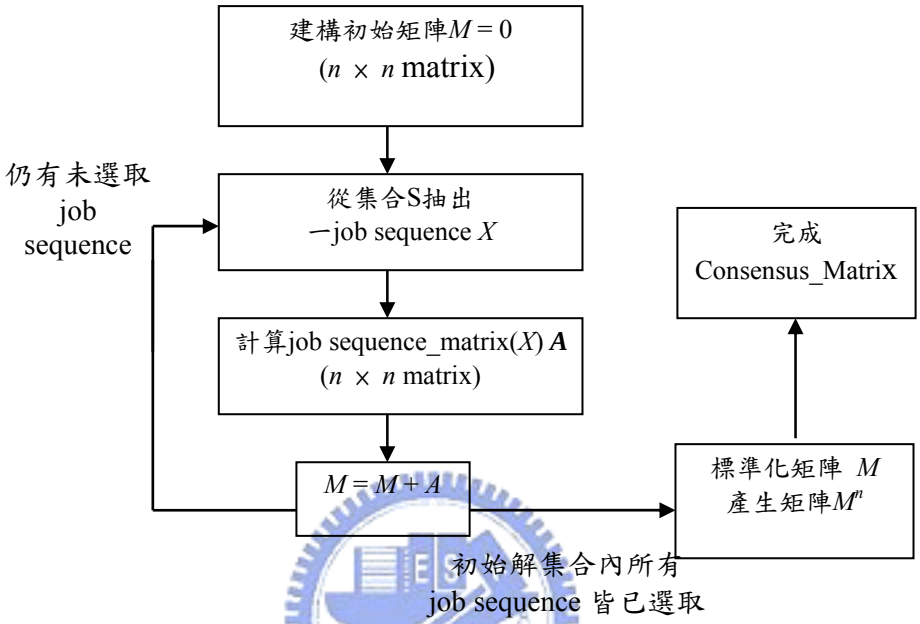

圖 3.5 建立共識矩陣流程

引導機制：共識矩陣 Consensus_matrix，M^n；初始為一個 0 矩陣 M ($n \times n$ matrix ，n為工件的個數)，從母體集合(集合S)中，藉Job sequence_matrix(X_k)流程去計算工件之間的優先關係矩陣A，形成新矩陣M (M=M+A)。其中Job sequence_matrix(X_k)流程，X_k代表輸入的工件排序，如圖 3.6 所示，我們初始一個工件優先關係矩陣A=[a_{ij}](1≤ $i, j \le n$)也為 0 矩陣($n \times n$ matrix)，只要工件排序X_i中，任兩工件J_i、J_j，工件J_i在工件J_j之前，a_{ij}就加 1。另外rank(J_i)為工件排序內工件i的優先順序。

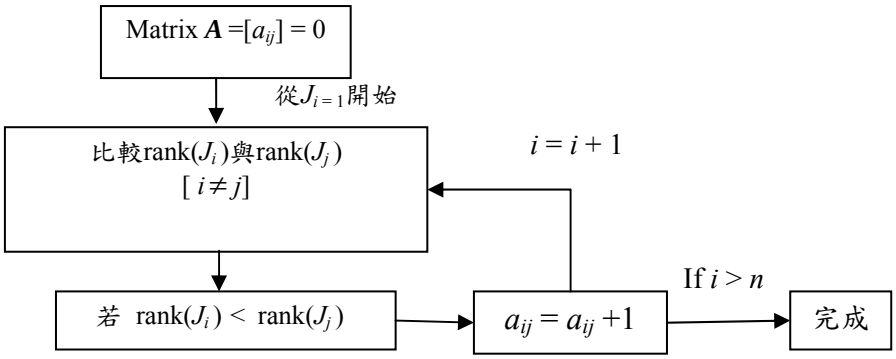

圖 3.6 Job sequence_matrix(X_k)流程

釋例，給定一組工件排序J_1→ J_5→ J_3→ J_2→ J_4，藉由Job sequence_matrix(X_k)流程，產生如表 3.3 的工件優先關係矩陣A。

表 3.3 工件優先關係矩陣 A

a_{ij}		後面(j)				
		1	2	3	4	5
前面(i)	1	X	1	1	1	1
	2	0	X	0	1	0
	3	0	1	X	1	0
	4	0	0	0	X	0
	5	0	1	1	1	X

此外，計算工件優先關係矩陣在此分為三種方法，包含了no weighting、linear weighting以及power weighting，前面提及的每次累加a_{ij}加 1 是no weighting，而linear weighting並不只是每個解代表權重相同，而是考慮到解的品質好壞，也就是總完工時間，而經過權重比較之後的累加值w_{ij}公式(5)如下：

$$w_{ij} = \frac{f(X_{worst}) - f(X_i)}{f(X_{worst}) - f(X_{best})} \qquad (5)$$

其中$f(X_{worst})$代表解集合中的最差解，$f(X_{best})$則是最佳解，經過此權重計算後可以得到每條解的相對重要性。而 power weighting 則是把上述的權重公式予以平方，增強權重的效果。

我們還需要把共識矩陣M標準化為$M^n = M/\mathrm{N}(S)$，N(S)代表初始解集合所有

的工件排序個數。標準化的共識矩陣M^n表達了機台上所有工件「先後關係」的機率，還有其特性為互補$M_{ij}^n+M_{ji}^n=1$，如表 3.4 所示，$M_{32}^n+M_{23}^n=1$，這矩陣提供資訊，說明在S集合中，百分 90 的工件排序，認為工件J_3在工件J_2前執行比較好。因此我們可以知道任兩工件之間J_i應該在J_j之前執行的共識比率。

表 3.4 模擬 5 個工件標準化後的共識矩陣

Matrix M		後面(j)				
		1	2	3	4	5
前面(i)	1	X	30	15	6	27
	2	0	X	3	30	12
	3	15	27	X	24	15
	4	24	0	6	X	18
	5	3	18	15	12	X

標準化矩陣 M
除以N(S)=30

Matrix M^n		後面(j)				
		1	2	3	4	5
前面(i)	1	X	1	0.5	0.2	0.9
	2	0	X	0.1	1	0.4
	3	0.5	0.9	X	0.8	0.5
	4	0.8	0	0.2	X	0.6
	5	0.1	0.6	0.5	0.4	X

Step 2：根據引導機制 Consensus_matrix 產生一個新的工件排序 Y

首先建立一個空集合$Q=\phi$，與所有工件的集合$X=\{J_i;\ 1\le i\le n\}$，抽取工件J_i從X中，因此$X = X-\{J_i\}$，而根據Consensus_matrix去決定J_i的優先值$h(J_i)$(稱為Consensus工件優先值)，把已經有工件優先值的J_i放進Q集合中，為$Q = Q \cup \{J_i\}$，直到抽取到X為空集合才停止。而新的染色體可以根據每個工件J_i的工件優先值去排序產生，工件優先值愈小工件的順序愈前面。

我們已知Consensus_matrix，還有在Q集合中的所有工件=$\{J_1,\ldots,J_{i-1}\}$和在Q中已計算的工件優先值$h(J_k)$，$J_k \in Q$，如何計算工件J_k的工件優先值，我們可以分為下列五個步驟:

(1)決定$h(q_i)$合理區域:

因工件有優先執行順序的限制，我們首先要判定不違反優先執行順序的限制出$h(J_i)$的合理區域，設定初始合理區域的開始點為$h_s = 0$，結束點為$h_e = Z$。對於每一個J_k，$J_k \in Q$，如果$m_{ik} = 0$ 和 $h(J_k) > h_s$則 $h_s = h(J_k)$，和如果$m_{ik} = 1$ 和 $h(J_k) <$

h_e則$h_e = h(J_k)$，更新後的合理區域，表示新的J_i在這區域中都不會有違反優先執行順序的限制。

(2) 在合理區域中選擇參考工件J_r:

已知合理區域所有的Q_h，$Q_h = \{J_k | h_s \le h(J_k) \le h_e, J_k \in Q\}$，為了尋找合理區域中的參考工件(reference job)J_r，我們希望尋找共識比例最大的，因此如下式(1)為J_i在J_k之後機率最大，和式(2)J_i為在J_r之前機率最大，兩者相比機率最大如式(3)，為我們要尋找的參考工件J_r，也代表工件J_i在工件J_r之前的共識比例為$p = M(J_i,J_r)$

$$k^* = Arg\ Max\ \{ M(J_k,J_i) \}\ ,\ J_k \in Q_h \quad (1)$$

$$r^* = Arg\ Max\ \{ M(J_i,J_r) \}\ ,\ J_r \in Q_h \quad (2)$$

$$r = Arg\ Max\ \{ M(J_i,J_{r^*}),\ \ M(J_{k^*},J_i) \} \quad (3)$$

(3) 根據 Consensus_matrix，執行伯努力(Bernoulli)試驗:

為了讓工件執行順序的產生變異，我們執行一次伯努力(Bernoulli)試驗，根據J_i與J_r的共識比例，當作機率，實驗成功代表工件J_i在工件J_r之前執行，因此 $h(J_i) < h(J_r)$，合理區域更新為$h_e = h(J_r)$;或是實驗失敗代表工件J_i在工件J_r之後執行，因此$h(J_i) > h(J_r)$，合理區域更新為$h_s = h(J_r)$。

(4) 更新合理區域:

實驗成功: 合理區域更新為$h_e = h(J_r)$

實驗失敗: 合理區域更新為$h_s = h(J_r)$

(5) 停止條件:

我們尋找到當合理區域中可插入的工件數$n(Q_h)$少於兩個，便可決定其$h(J_i)$值。因此$n(Q_h) > 2$ 我們繼續尋找新的參考工件J_r，回到步驟(2)，直到$n(Q_h) \le 2$，則決定$h(J_i) = (h_s + h_e)/2$。

茲以一釋例說明如何決定$h(J_i)$，如圖 3.7 所示，在已知Q=$\{J_1, J_2, J_3, J_4\}$，

$Q_h=\{J_k|\ h_s \le h(J_k) \le h_e, J_k \in Q\}$下，若要插入工件$J_5$，決定$h(J_5)$。第一步驟:先決定$h(J_i)$合理區域為$h_s= h(J_1)$，$h_e= h(J_4)$。第二步驟:為了尋找參考工件$J_r$，根據(1)(2)(3)式，可以求得$r=J_2$。第三步驟:根據$J_2$做伯努力(Bernoulli)試驗，如果失敗，$h(J_5) > h(J_2)$。第四步驟:合理區域更新為$h_s = h(q_r)= h(J_2)$。第五步驟: $n(Q_h) = 2$，停止搜尋，$h(J_5)=(h(J_2)+ h(J_4))/2$，$J_5$插入在$J_4$和$J_2$之間。

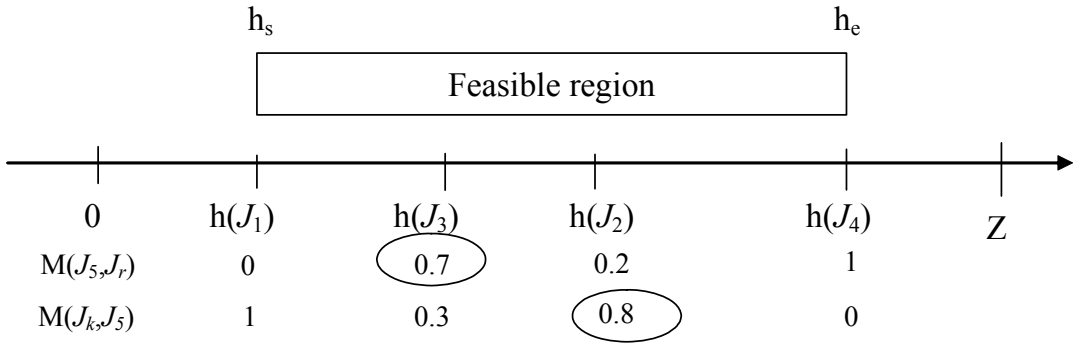

圖 3.7 選擇參考任務 J*r*

調整共識因子影響力

從實驗結果發現，由共識因子產生的解，強烈的由眾人的意見也就是共識矩陣所主導，連帶使得產生的解容易掉入局部最佳化的情形當中，因此為了改進這個缺點，而且增加演算法產生解的彈性，我們將工件J_i對於參考工件J_r的共識比例p，乘上一個因子調整參數r。此調整參數包含了三種情形。第一種，也就是前面所提到的$r = 1$。第二種為了減少共識比例的主導性，設定$r = 0.8$，就是所謂的降級因子。第三種為了增加演算法共識比例的多元化，同時應用了升級與降級因子，當$p > 0.5$ 時，採用降級因子$r = 0.8$；$p < 0.5$ 時，採用昇級因子$r = 1.2$。

(二) 田口方法

在 1949 年，田口玄一(Genichi Taguchi)博士於日本電信實驗室工作時，發現傳統實驗設計方法在實務上並不適用，進而發展田口方法(Taguchi methods)(Taguchi, 1986&1987；黎正中, 2000；蘇朝墩, 2002)，而成為他所謂品質工程的基本原理，而其精神就是一種穩健設計的實驗方法。田口所發展的是一透過實驗進行系統參數最佳化設計的方法，具實際的應用性，而非以困難的統計為

依歸。

在此田口方法中，是利用直交表來產生實驗的進行。在一般統計全因子設計中，當因子數目增加時，實驗次數會隨之增加；而部分因子設計則會增加實驗方法的複雜性。在此田口方法中，是利用直交表來產生實驗的進行，利用直交表來收集資料，能讓我們以較少的實驗而獲得更可靠的因子效果估計量。

直交表的觀念

直交表種類繁多，先將代表直交表的符號定義說明如下，如圖 3.8：

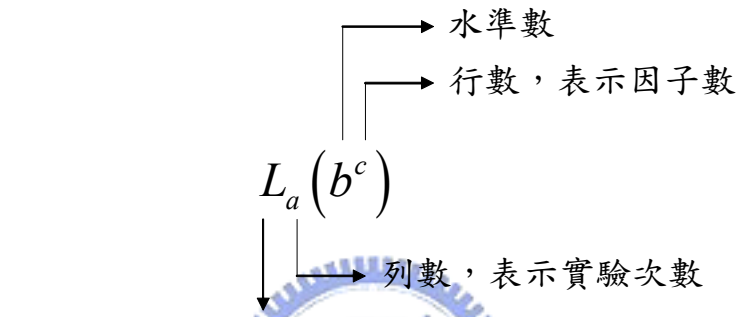

圖 3.8 田口直交表符號定義

表 3.5 為 $L_8\left(2^7\right)$ 直交表，表中本體內之 1 和 2 數字分別表示因子的水準一和水準二。直交表中的每一行代表實驗中的某一個特定因子的變化情況。「行」的編號，可供因子或交互作用配置其上之用；*L*8 直交表上共有 7 行，代表最多能配置七個因子。列數等於直交表的實驗次數，*L*8 直交表的實驗次數為 8，故實驗編號由 1 至 8。

表 3.5 $L_8(2^7)$直交表

實驗編號	行						
	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
2	1	1	1	2	2	2	2
3	1	2	2	1	1	2	2
4	1	2	2	2	2	1	1
5	2	1	2	1	2	1	2
6	2	1	2	2	1	2	1
7	2	2	1	1	2	2	1
8	2	2	1	2	1	1	2

若以本 HFSP 問題進行應用，可以將 2 個水準轉換成 2 個 chromosome 解，7 個因子轉換成 7 個工件，而行與列交集的數字所代表意義：「1」代表第 2 個 chromosome 的 rank，「2」代表第 2 個 chromosome 的 rank。但是由於本問題的工件數 $n = 5,10,20,50,100$ 並沒有完全相符因子數目的直交表，所以採用 $2^c \geq n$ 來得到我們所要的直交表。

求解流程

值得注意，為了因應 Taguchi methods 演算法特性，首先要將 chromosome 所代表的工件排序轉換為所有工件的執行次序，如圖 3.9 所示，以避免有不合理解的產生。

Rank	1	2	3	4	5
Job	J_1	J_5	J_3	J_2	J_4

→

Job	J_1	J_2	J_3	J_4	J_5
Rank	*1*	*4*	*3*	*5*	*2*

圖 3.9 工件排序轉換為工件執行順序

進而首先input兩組chromosome解(C_1,C_2)，依直交表的實驗例子算出每個例子的makespan值(m_i)取倒數，然後對每個工件J_i計算累計個別屬於C_1、C_2的 1/ m_i值加總(屬於C_1的累計值標示為Sm_1，同理屬於C_2的累計值標示為Sm_2)，

假設兩條chromosome解轉換後的工件執行次序C_1 = {1; 6; 7; 3; 5;2; 4}，C_2 = {1; 7; 6; 2; 5; 3; 4}來當例子，由於有七個工件，所以採用 $2^3 = 8$ 直交表，此直交表產生 8 個實驗組合例子，而其中第 8 個工件自動省略。接下來計算每個例子的

makespan值(m_i)並隨之取倒數，然後對每個工件J_i計算屬於C_1的Sm_1、屬於C_2的Sm_1，然後依此決定最佳的工件組合C_T。由於makespan值是望小特性，取倒數後的累積加總值Sm_1、Sm_1則轉變為望大特性。由表 3.6，可得知J_1到J_3和J_7是採用C_2的基因，J_4到J_6是用C_1的基因，最後由此流程得到一組完整的chromosome解C_T = {1; 7; 6; 3; 5;2; 4}

表 3.6 田口方法釋例

L8	J_1 基底a	J_2 基底b	J_3 基底c	J_4 a+b	J_5 b+c	J_6 c+a	J_7 a+b+c	makespan	1/makespan
1	1	1	1	1	1	1	1	25	1/25
2	1	1	2	1	2	2	2	28	1/28
3	1	2	1	2	1	2	2	22	1/22
4	1	2	2	2	2	1	1	26	1/26
5	2	1	1	2	2	1	2	25	1/25
6	2	1	2	2	1	2	1	23	1/23
7	2	2	1	1	2	2	1	27	1/27
8	2	2	2	1	1	1	2	22	1/22
C_T	2	2	1	2	1	1	2	20	

3.3.5 更新解修正器

此修正器由標竿演算法作者在文獻中所提出，是指針對一個工件排序，用有系統的方法找出它的鄰近解(neighborhood solution)，企圖在區域解中找到一個更好的解來改善原本的解，也就是所謂的鄰近搜尋法(local search)。本研究的鄰近搜尋法包括下列三個步驟：

步驟一：

鄰近搜尋的方式，是將每個排序中的工件，分別與其後最近的六個工件做交換，然後各自進而產生一組新的排程，並計算該排程的最大完工時間。

步驟二：

若鄰近搜尋的解有改善或是與原順序的完工時間相等，則記錄下來，作為後續之用，並以此順序繼續往後搜尋。

步驟三：

反覆步驟一和步驟三，直到所有的工件都被交換過之後才停止。

第四章 實驗結果與分析

在此研究中，希望階段性的利用pilot與comprehensive study，逐步的找出最佳的幾個演算法組合，然後在與標竿演算法進行績效的比較。本研究程式以Visual C++ 6.0 撰寫，所進行的電腦CPU為Intel Pentium4 3.0GHz。在此實驗中，每個version的演算中止條件為在演化過程中只能產生的chromosome解限制數量下，來進行績效的比較。每個情境其chromosome解的限制數量不為相同，其最大解限制數量則是以標竿演算法跑完 500 世代所會產生的解數目為依據，依此值依序往下取 0.5、0.25 倍的解數目來作為績效的求解點。每個演算法要跑的情境中每個問題需要重複跑 5 次程式。績效的衡量是利用所求得之最小完工時間(C_{max})與下界差距的百分比($\frac{C_{\max}-LB}{LB}\times 100\%$)的平均作為比較來求得，而平均則是從每個情境的內的 10 個例題來加總取平均所得(Average percentage deviation of the of the makespan values from the lower bounds)。

從第 3.2 節的實驗組合得知由於 version 數過多，因此先行進行 pilot study，然後在模組間進行演算法績效的優劣比較，以篩選出較佳的演算法組合配方，隨後才進行全情境實驗的 comprehensive study。

4.1 引導實驗(Pilot study)

由於 pilot study 數據過於多組(72)，為了要在同個模組內找出哪些方法較為突出，因此下面利用 pilot study 模組績效優劣尋找過程來找出較佳的演算法組合。另外由於共識因子內部參數組合過多，因此先就初始解產生器與更新解產生器 2 兩個部份來進行模組內方法的比較，最後再從共識因子找出最佳的參數方法組合。此 72 個 version 的 pilot study 的數據顯示如表 4.1。

表 4.1 pilot study 實驗結果

Version	實驗組合	Average percentage deviation		
		n = 20 ， m = 5		
		chromosme數目		
		15000	30000	60000
V1	A1 + B1 & B3 & B6 + C1 + D1	7.8	4.71	3.61
V2	A1 + B1 & B3 & B7 + C1 + D1	7.82	4.72	3.68
V3	A1 + B1 & B3 & B8 + C1 + D1	8.01	4.91	3.71
V4	A1 + B1 & B4 & B6 + C1 + D1	7.66	4.78	3.65
V5	A1 + B1 & B4 & B7 + C1 + D1	7.75	4.85	3.65
V6	A1 + B1 & B4 & B8 + C1 + D1	7.73	4.9	3.72
V7	A1 + B1 & B5 & B6 + C1 + D1	7.98	4.86	3.68
V8	A1 + B1 & B5 & B7 + C1 + D1	7.86	4.79	3.65
V9	A1 + B1 & B5 & B8 + C1 + D1	7.85	5	3.76
V10	A1 + B2 & B3 & B6 + C1 + D1	7.29	4.47	3.51
V11	A1 + B2 & B3 & B7 + C1 + D1	7.57	4.51	3.56
V12	A1 + B2 & B3 & B8 + C1 + D1	7.41	4.64	3.63
V13	A1 + B2 & B4 & B6 + C1 + D1	7.22	4.42	3.43
V14	A1 + B2 & B4 & B7 + C1 + D1	7.58	4.53	3.51
V15	A1 + B2 & B4 & B8 + C1 + D1	7.72	4.67	3.64
V16	A1 + B2 & B5 & B6 + C1 + D1	7.14	4.18	3.37
V17	A1 + B2 & B5 & B7 + C1 + D1	7.77	4.74	3.63
V18	A1 + B2 & B5 & B8 + C1 + D1	7.51	4.72	3.68
V19	A1 + B1 & B3 & B6 + D1	8.12	5.01	3.78
V20	A1 + B1 & B3 & B7 + D1	7.98	5.05	3.79
V21	A1 + B1 & B3 & B8 + D1	7.94	5.08	3.81
V22	A1 + B1 & B4 & B6 + D1	8.02	5	3.77
V23	A1 + B1 & B4 & B7 + D1	8.1	4.99	3.77
V24	A1 + B1 & B4 & B8 + D1	8.15	5.26	3.92
V25	A2 + B1 & B5 & B6 + D1	7.93	5.07	3.81
V26	A1 + B1 & B5 & B7 + D1	7.95	5.03	3.85
V27	A1 + B1 & B5 & B8 + D1	8.01	5.14	3.95
V28	A1 + B2 & B3 & B6 + D1	7.86	5.06	3.7
V29	A1 + B2 & B3 & B7 + D1	8.1	4.99	3.77
V30	A1 + B2 & B3 & B8 + D1	7.93	4.85	3.74
V31	A1 + B2 & B4 & B6 + D1	7.53	4.67	3.63
V32	A1 + B2 & B4 & B7 + D1	7.83	4.74	3.63
V33	A1 + B2 & B4 & B8 + D1	7.63	4.82	3.67
V34	A1 + B2 & B5 & B6 + D1	7.84	4.93	3.73
V35	A1 + B2 & B5 & B7 + D1	8.06	4.92	3.78
V36	A1 + B2 & B5 & B8 + D1	7.94	5.08	3.88

V37	A2 + B1 & B3 & B6 + C1 + D1	7.17	4.36	3.46
V38	A2 + B1 & B3 & B7 + C1 + D1	7.24	4.5	3.51
V39	A2 + B1 & B3 & B8 + C1 + D1	7.35	4.5	3.54
V40	A2 + B1 & B4 & B6 + C1 + D1	7.46	4.34	3.46
V41	A2 + B1 & B4 & B7 + C1 + D1	7.38	4.39	3.49
V42	A2 + B1 & B4 & B8 + C1 + D1	7.43	4.43	3.55
V43	A2 + B1 & B5 & B6 + C1 + D1	7.57	4.54	3.55
V44	A2 + B1 & B5 & B7 + C1 + D1	7.24	4.5	3.48
V45	A2 + B1 & B5 & B8 + C1 + D1	7.57	4.55	3.58
V46	A2 + B2 & B3 & B6 + C1 + D1	6.99	4.29	3.36
V47	A2 + B2 & B3 & B7 + C1 + D1	6.86	5.11	3.33
V48	A2 + B2 & B3 & B8 + C1 + D1	7.03	4.32	3.44
V49	A2 + B2 & B4 & B6 + C1 + D1	6.85	4.05	3.27
V50	A2 + B2 & B4 & B7 + C1 + D1	7.16	4.2	3.32
V51	A2 + B2 & B4 & B8 + C1 + D1	7.15	4.42	3.43
V52	A2 + B2 & B5 & B6 + C1 + D1	7.29	4.29	3.35
V53	A2 + B2 & B5 & B7 + C1 + D1	7.2	4.41	3.45
V54	A2 + B2 & B5 & B8 + C1 + D1	7.55	4.53	3.51
V55	A2 + B1 & B3 & B6 + D1	7.59	4.73	3.62
V56	A2 + B1 & B3 & B7 + D1	7.47	4.63	3.62
V57	A2 + B1 & B3 & B8 + D1	7.65	4.84	3.67
V58	A2 + B1 & B4 & B6 + D1	7.69	4.64	3.57
V59	A2 + B1 & B4 & B7 + D1	7.57	4.51	3.56
V60	A2 + B1 & B4 & B8 + D1	7.51	4.72	3.68
V61	A2 + B1 & B5 & B6 + D1	7.86	4.79	3.64
V62	A2 + B1 & B5 & B7 + D1	7.47	4.69	3.68
V63	A2 + B1 & B5 & B8 + D1	8.15	5.03	3.76
V64	A2 + B2 & B3 & B6 + D1	7.63	4.59	3.53
V65	A2 + B2 & B3 & B7 + D1	7.85	4.77	3.63
V66	A2 + B2 & B3 & B8 + D1	7.32	4.56	3.59
V67	A2 + B2 & B4 & B6 + D1	7.49	4.45	3.46
V68	A2 + B2 & B4 & B7 + D1	7.39	4.3	3.41
V69	A2 + B2 & B4 & B8 + D1	7.03	4.32	3.5
V70	A2 + B2 & B5 & B6 + D1	7.34	4.53	3.44
V71	A2 + B2 & B5 & B7 + D1	7.86	4.79	3.62
V72	A2 + B2 & B5 & B8 + D1	7.79	4.73	3.69

4.1.1 初始解產生器模組演算法績效比較

由於從表 4.1 可看出 A2(螞蟻演算法)幾乎在每個其他模組相同組合情況下的績效皆優於 A1(Random)；另外從表 4.1，我們各自擷取出 A1 (V13 和 V16) 與 A2 (V49 和 V50) 績效表現最佳的兩個 version 整理如表 4.2，也可看出 A2 優於 A1。不論是在一對一比較或者整體最佳比較，A2 都勝過 A1，因此在後續的 comprehensive study，在本模組將只考慮包含 A2 的 version。

表 4.2 A1 與 A2 最佳的比較

方法	Version	實驗組合				Average percentage deviation n = 20 ， m = 5 chromosme數目		
						15000	30000	60000
A1	V13	A1	B2 & B4 & B6	C1	D1	7.22	4.42	3.43
	V16		B2 & B5 & B6			7.14	4.18	3.37
A2	V49	A2	B2 & B4 & B6			6.85	4.05	3.27
	V50		B2 & B4 & B7			7.16	4.2	3.32

4.1.2 更新解產生器 2 模組演算法績效比較

同樣的，在更新解產生器 2 模組的比較情形如同 4.1.1 節，在表 4.1 內，C1 (Taguchi methods)有無 version 各自之間的比較，都是擁有 C1 的 version 表現較佳；而在表 4.3，把 C1 有無的最佳表現擷取出來比較，有 C1 的較佳。由以上實驗數據可得出，在更新解產生器 2 內，C1 是必要的演算方法。因此在後續的 comprehensive study，本模組也將只考慮包含 C1 的 version。

表 4.3 C1 有無的最佳比較

方法	Version	實驗組合				Average percentage deviation n = 20 ， m = 5 chromosme數目		
						15000	30000	60000
C1	V49	A2	B2 & B4 & B6	C1	D1	6.85	4.05	3.27
	V50		B2 & B4 & B7			7.16	4.2	3.32
無C1	V68		B2 & B4 & B7	無C1		7.39	4.3	3.41
	V70		B2 & B5 & B6			7.34	4.53	3.44

4.1.3 更新解產生器 1 模組演算法績效比較

在此模組內是就共識因子演算法的參數組合與以詳加比較，以找出最佳表現的組合方法。從 4.1.1 與 4.1.2 節，我們可以得到在初始解產生器擁有 A2 與更新解產生器 2 擁有 C1 的 version 表現較優良，因此我們以擁有 A2 與 C1 為前提，來找尋表 4.1 中表現最佳的五個 version (V46、V47、V49、V50 和 V52) ，得到如表 4.4 所示。我們可以看出，表現最好的五個 version 在共識因子世代產生更新解參數部分，都是以 B2 (1000 個)為參數選擇；而在 weighting 參數選擇部分，則可得到 B4 (linear weighting)明顯勝出；最後在 Rate 部分，B6 與 B7 表現互有勝負，但是由於要確實找出最佳的一組參數組合來進行下一階段的綜合實驗，而且整體最佳 V49 是由 B6 組合而成，因此此部分選擇 B6 為代表。由上面流程我們可以得到，在更新解產生器 1 模組內，我們選擇 B2 & B4 & B6 作為本模組的參數組合代表。

表 4.4 擁有 A2 與 C1 的 5 個最佳 version

Version	實驗組合				Average percentage deviation n = 20 ， m = 5 chromosme數目		
					15000	30000	60000
V46	A2	B2 & B3 & B6	C1	D1	6.99	4.29	3.36
V47		B2 & B3 & B7			6.86	5.11	3.33
V49		B2 & B4 & B6			6.85	4.05	3.27
V50		B2 & B4 & B7			7.16	4.2	3.32
V52		B2 & B5 & B6			7.29	4.29	3.35

4.1.4 Pilot study 結果與分析

經過上面的 pilot study 的績效分析流程，我們可以得到 V49(A2+ B2 & B4 & B6 + C1 + D1)為引導實驗中最得到的最佳組合，實際上 V49 其績效表現也是最佳。我們隨後將利用 V49 為基礎，進行下一階段的 comprehensive study。

4.2 綜合實驗(Comprehensive study)

從 4.1 節的引導實驗中，我們可以得到 V49 為此階段實驗的基礎，我們將以此來與標竿演算法進行全情境績效的比較。另外在 4.1.2 節更新解產生器 1 模組的比較內，明顯可看出 C1(Taguchi methods)的有無對實驗結果有相當程度的影響，我們期待對 C1(Taguchi methods)做更進一步的研究，因此希望將 C1 產生的解數目在更新解產生器部分的比例與以提高，因此我們將更新解產生器內 consensus operator 與 Taguchi methods 解數目比例分成兩種，而形成兩種 version：V49 (9：1) 與 V73 (8：2)。因此在綜合實驗部份包含的 version 種類整理如表 4.5 所示，另外我們將以 V49 與 V73 來和 V0 進行綜合實驗，我們得到了表 4.6 的數據結果。

表 4.5 演算法組合

Version	實驗組合
V0	A2+D1
V49	A2 + B2 & B4 & B6 + C1 + D1 (B與C比例9：1)
V73	A2 + B2 & B4 & B6 + C1 + D1 (B與C比例8：2)

表 4.6 Comprehensive study 數據結果

percentage deviation	n = 5								
	m = 2			m = 5			m = 8		
	V0	V49	V73	V0	V49	V73	V0	V49	V73
2500	***20.75***	22.95	23.99	***29.84***	33.01	32.9	***28.71***	31	32.95
5000	***16.65***	17.77	16.87	***23.72***	24.16	24.35	***23.62***	23.74	24.73
10000	***14.72***	14.58	14.55	***21.43***	21.31	21.39	***19.68***	19.48	19.52
percentage deviation	n = 10								
	m = 2			m = 5			m = 8		
	V0	V49	V73	V0	V49	V73	V0	V49	V73
7500	***3.6***	4.17	3.86	***13.65***	14.15	13.92	***25.06***	27.17	25.85
15000	***1.92***	2.2	2.07	***10.43***	10.87	10.58	***17.74***	19.84	18.14
30000	***1.58***	1.5	1.51	***9.52***	9.59	9.68	***16.45***	16.44	16.42
percentage deviation	n = 20								
	m = 2			m = 5			m = 8		
	V0	V49	V73	V0	V49	V73	V0	V49	V73
15000	***4.04***	6.51	6.64	***4.85***	7.25	7.18	***8.82***	10.06	9.76
30000	***2.3***	3.03	3.85	***3.65***	4.51	3.98	***7.55***	7.94	8.03
60000	***1.91***	2.08	2.12	***3.12***	3.25	3.17	***6.71***	6.87	7.01
percentage deviation	n = 50								
	m = 2			m = 5			m = 8		
	V0	V49	V73	V0	V49	V73	V0	V49	V73
37500	***3.86***	5.45	6.08	***2.64***	3.38	3.76	**3.98**	7.16	5.05
75000	***2.89***	3.35	3.44	***1.89***	2.22	2.38	**3.06**	3.95	3.53
150000	***2.42***	2.79	2.64	***1.57***	1.82	1.76	**2.63**	2.85	2.92

4.2.1 實驗結果與分析

從實驗數據我們可以得到兩種結果：在小情境問題中(n = 5，n = 10)，V49 和 V73 表現都較 V0 佳(除了 n = 10，m = 5)，而 V49 與 V73 之間的績效倒是互有勝負，而且沒有很明顯的差異。至於在大情境問題當中(n = 20，n = 50)，V0 則一面倒的優於其他兩個 version。

對此現象，我們進行以下討論與分析：在小情境問題中，可能 consensus operator 與 Taguchi methods 世代循環的概念較適用於在小問題，以致能產生較佳的解品質。而且由數據可以看出，V49 和 73 在前兩個解限制數量的表現皆優於 V0，但在最終解限制條件下，大問題皆表現較差，可以推論可能由於大情境問題的解空間過於龐大，始得在演算法後段要產生極佳解品質時，consensus operator 與 Taguchi methods 產生解品質收斂速度不及標竿演算法，因此造成了在大情境問題中其績效皆無法勝過標竿演算法。

4.2.2 與其他過去文獻比較結果

在本節將 Ying & Lin (2006)內其比較的對象擷取出來，其採用方法是塔布搜尋法(Tabu search)(Oguz et al., 2004)，然後與 V49 和 V73 進行績效的比較，得到了表 4.7 如下。

從實驗結果可以看的出來，在 12 個情境當中，V49 與 V73 贏了 10 個情境，輸掉的 2 個情境為(n = 5，m = 2)、(n = 50，m = 2)。

表 4.7 過去文獻比較數據結果

Average percentage deviation	n = 5								
	m = 2			m = 5			m = 8		
	Tabu	V49	V73	Tabu	V49	V73	Tabu	V49	V73
績效	***11.42***	14.58	14.55	***38.98***	21.31	21.39	***38.49***	19.48	19.52
Average percentage deviation	n = 10								
	m = 2			m = 5			m = 8		
	Tabu	V49	V73	Tabu	V49	V73	Tabu	V49	V73
績效	***3***	1.5	1.51	***29.42***	9.59	9.68	***46.53***	16.44	16.42
Average percentage deviation	n = 20								
	m = 2			m = 5			m = 8		
	Tabu	V49	V73	Tabu	V49	V73	Tabu	V49	V73
績效	***2.88***	2.08	2.12	***24.4***	3.25	3.17	***42.47***	6.87	7.01
Average percentage deviation	n = 50								
	m = 2			m = 5			m = 8		
	Tabu	V49	V73	Tabu	V49	V73	Tabu	V49	V73
績效	***2.23***	2.79	2.64	***10.51***	1.82	1.76	***21.04***	2.85	2.92

4.2.3 研究架構的改善方向

在經過一連串引導與綜合試驗之後，我們從結果可以對本架構提出進一步提出可能的研究方向：

(1)初始解產生器

在利用螞蟻演算法產生新解過程中，或許可以考慮將原本世代循環中利用最佳解更新參數改成由最佳的多個解同時進行參數更新，概念來自共識因子中採用多數人意見的優點。

(2)更新解產生器

在共識因子部份，由於共識矩陣的品質好壞代表了產生新解的發展性，因此我們在抽取用以產生共識矩陣的解時，除了考慮解的品質外，也可以進一步對解的差異化進行統整性的考量，或許可以進一步提升共識矩陣的效率。

而在田口方法，我們可以嘗試將直交表產生的解來作為後續修正器解的選擇，以避免浪費了其中可能有好解的可能性。

(3)更新解修正器

對於交換排序的方法可以做多方嘗試，像是區段性的交換或者交換的數量增加，都是可以考慮的方向。

第五章　結論與研究方向

在本研究中，利用了共識因子與田口方法來和標竿演算法：螞蟻演算法作有系統的整合，然後進行績效的比較，最後並對方法的優缺點提出分析與可能的改善。

在共識因子部分，利用了眾人的優越意見，來逐代利用共識產生新解，但是其缺點在於參考的意見來源範圍可能過於狹隘，容易若入區域最佳的情形。田口方法則是利用有系統的實驗來產生品質優良的解，但是在求解過程中，產生了過多利用不到的解，進而影響求解與收斂的速度。

本研究提出的演算法組合雖然無法勝過標竿演算法：Ying & Lin (2006)應用的求解方法，但是在某些小情境問題仍舊有研究與探討的空間。另外也超越了標竿演算法所比較的對象。另外，也是首次嘗試將共識因子與田口方法應用於此 HFSP 問題當中。

未來的研究方向，可以從三方面著手：(1)共識因子的抽樣方式同時考慮品質與差異性。(2)修改以增加田口方法產生新解的效率。(3)整體架構內，不同模組間對於世代循環概念的加入與應用。

參考文獻

吳政翰，“DHC 系統之任務指派與排程的新型基因算法”，國立交通大學工業工程與管理學系碩士論文，2006。

蘇朝墩，“品質工程”，中華民國品質學會，2002。

黎正中，李家琪，“田口方法的應用與發展”，品質學報，第 7 卷第 2 期，頁 117-137，2000。

Arthanari, T.S., and Ramamurthy, K.G., “An extention of two machines sequencing problem,” *Operational Research*, 8, pp. 10-22, 1971.

Brah, S.A. and Hunsucker, J.L., “Branch and bound algorithm for the flowshop with multiple processors,” *European Journal of Operational Research, 51*, pp. 88-99, 1991.

Campbell, H.G., Dudek, R.A. and Smith, M.L., “A heuristic algorithm for the n-job,m-machine sequence problem,” *Manag.Sci., 16*, pp. 630-637, 1970.

Dorigo, M., “Optimization,learning and natural algorithm,” *Biosystem*, 43, pp. 73-81, 1992.

Guinet, A., “A computational study of heuristics for two-stage flexible flowshops,” *Journal of Operational Research* , 34, pp. 1399-1415, 1996.

Gupta, J. N. D. and Tunc, E. A., “Schedules for a two stage hybrid flow shop with parallel machines at the second stage,” *International Journal of Production Research*, 29, pp. 1489-1502, 1991.

Gupta, J.N.D., “Two-stage hybrid flowshop scheduling problem,” *Operational Research Society, 39*, pp. 359-364, 1988.

Hanijanto Soewandi and Salah Elmaghraby, “Sequencing on two-stage hybrid flowshops with uniform machines to minimize makespan,” *IIE Transactions*, 35, pp. 467-477, 2003.

Kuo-Ching Ying and Shin-Wei Lin, “Multiprocessor task scheduling in multistage

hybrid flow-shops: an ant colony system approach," *International Journal of Production Research*, 44, pp. 3161-3176, 2006.

Oguz Ceyda, Lin, Bertrand M. T., and Edwin Cheng, T. C., "Two-stage flowshop scheduling with a common second-stage machine," *Computer & Operations Research*, 24, pp. 1169-1174, 1997.

Oguz, C., Zinder, Y., Do, V.H. Janiak, A., and Lichtenstein, M., "Hybrid flow-shop scheduling problems with multiprocessor task systems," *European Journal of Operational Research*, 152, pp. 115-131, 2004.

Palmer, D.S., "Sequencing jobs through a multi-stage process in the minimum total time-a quick method of obtaining a near optimum," *Oper. Res.Quart.*, 16, pp. 101-107, 1965.

Rajendran, C. and Chaudhunri,D., "A multi-stage parallel-processor flowshop problem with minimum flowtime," *European Journal of Operational Research*, 57, pp. 111-122, 1992.

Raghu, T.S. and Rajendran, C., "An efficient dispatching rule for scheduling in a job shop," *Int. J. Prod. Econ.,*, 32, pp. 301-313, 1993.

Serifoglu, F.S., Ulusoy, G., "Multiprocessor task scheduling in multistage hybrid flow-shops: a genetic algorithm approach," *Journal of the Operational Research Society*, 55, pp. 504-512, 2004.

Viger, A., Commdeur, P. and Proust,C., "A branch and bound approach to minimize the total completion time in a k-stage hybrid flowshop," *Proceedings 1996 IEEE Conference on Emerging Technologies and Factory Automation*, 1, pp. 15-20, 1996.