國立交通大學

工業工程與管理學系

博士論文

以兩種染色體表達法求解具工件族特性之排程問題

A Comparison of Two Chromosome Representation Schemes Used in Solving a Family-Based Scheduling Problem

研 究 生：陳振富

指導教授：巫木誠 博士

中華民國一百零一年七月

以兩種染色體表達法求解具工件族特性之排程問題

A Comparison of Two Chromosome Representation Schemes Used in Solving a Family-Based Scheduling Problem

研 究 生：陳振富　　Student：Chen-Fu Chen

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

博士論文

A Dissertation

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 Doctor of philosophy

in

Industrial Engineering and Management

March 2012

Hsin-chu, Taiwan, Republic of China

中華民國一百零一年七月

以兩種染色體表達法求解具工件族特性之排程問題

研究生：陳振富　　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

本篇論文探討在運用進化演算法(meta-heuristic algorithms)搭配新的解表達法(solution representation)時的影響性。本研究以一個流線型製造單元系統(PMFS; permutation manufacturing-cell flow shop scheduling problem)的排程問題為研究範疇來討論此影響性。上述排程問題，過去研究都使用同一解表達法（簡稱 S_{old}），本研究提出一個新的解表達法(簡稱 S_{new})，戴邦豪(2010)、林耿漢(2011)與李奕勳(2011)分別應用 S_{new} 於基因演算法(genetic algorithm)、禁忌搜尋演算法(tabu search)與蟻群最佳化演算法(ant colony optimization)。本研究的第一個重點將探討為何 S_{new} 比採用 S_{old} 的解品質好之原因分析。透過數據的分析，可以獲得以下的結果。在禁忌搜尋演算法中 S_{old} 有較高的機率會陷入「**迴圈**」之中。在基因演算法與蟻群最佳化演算法中， S_{old} 容易在解的特定決策參數上陷入「**同質化**」現象；而此現象導致其進化過程無法再搜尋到更好的解。本研究的第二個重點是討論同時改善進化演算法（混用基因演算法與禁忌搜尋演算法)與改善解表達法(採用 S_{new})對解品質的影響，實驗結果顯示兼採此兩種方法，解的品質的改善效果會比只採單一方法更好。本研究的主要貢獻是證實改善解表達法(solution representation)對進化演算法的重要性，此概念可進一步延伸到進化演算法的相關應用。

關鍵詞：進化演算法，禁忌搜尋法，基因演算法，蟻群最佳化演算法，解表達法，排程

A Comparison of Two Chromosome Representation Schemes Used in Solving a Family-Based Scheduling Problem

Student：Chen-Fu Chen　　Advisor：Dr. Muh-Cherng Wu

Department of Industrial Engineering and Management
National Chiao Tung University

Abstract

This dissertation examines the effect of using new solution representation in the application of meta-heuristic algorithms. The effect is justified by solving a scheduling problem, called permutation manufacturing-cell flow shop scheduling problem (PMFS). Most prior studies on PMFS used a common solution representation (called S_{old}); we propose a new one (called S_{new}). Based on experiments (Tai 2010, Lin 2011, Li 2011), S_{new} appears to outperform S_{old} while the two representations are embedded in tabu search, genetic algorithm (GA), and ant colony optimization (ACO). This dissertation attempts to explain why S_{new} outperforms S_{old} in these three algorithms. Through empirical analyses, the following findings are obtained. In the tabu algorithms, S_{old} tends to have a higher probability of being trapped into a loop. In the GA and ACO algorithms, S_{old} tends to result in a *homogeneous* set of solutions for some critical scheduling decisions, and reduces the probability of getting better solutions. This dissertation also proposed a new algorithm *GA_Tabu_*S_{new}, which outperforms all prior algorithms in solving the PMFS problem.

Keywords: Meta-heuristic algorithms, Tabu search, Genetic Algorithm, Ant colony Optimization, Solution Representation, Scheduling

誌謝

本論文得以順利完成，首要感謝的是巫木誠教授的悉心指導與教誨。在學術上，巫教授總是分享自己的經驗與做研究所需的態度；時常強調誠信的重要性，並教導如何使用有效率的方法去解決問題。除此之外，巫老師提倡有紀律的生活以及有系統的思考模式，能幫助在學術研究上吸收知識並剖析擷取核心觀念，此種方式讓我獲益頗多。同時也感謝許錫美教授、彭德保教授、王文派教授與謝玲芬教授在論文口試時，所給予的寶貴意見與指導，讓本論文更加完備。

感謝佛羅里達州立大學工業與製造工程系主任 Prof. Chuck Zhang 和 Prof. Ben Wang 提供學生機會可以到佛羅里達州立大學實習一個月，學習許多不同領域的知識。也感謝喬治亞理工學院的張義隆教授推薦與吳建福教授的幫助，讓學生到喬治亞理工學院訪問一年，充實了供應鏈與資料庫相關課程，也了解國外學術研究與國內的差異性。

在研究所的五年中，要感謝同實驗室的學長：施昌甫學長、蘇泰盛學長與邱志文學長，能時常教導並鼓勵我做研究。還有同實驗室的學弟妹：黃亮銓、李奕勳、林耿漢與其他許許多多的學弟妹，能與我相互的討論研究。

最後，特別感謝我最親愛的家人，在論文撰寫期間給我許多的支持。特別是我的父母親陳明瑞先生與簡月嬌女士，感謝您們多年來的辛勞與關懷，您們的體諒讓我得以專心在論文的研究上。也感謝我的妻子邱羣雅小姐的支持。在此，謹以此論文獻給我最敬愛的家人、師長與朋友。

陳振富 于 新竹交大

2012’8’20

目錄

中文摘要.....i
Abstract.....ii
誌謝..... iii
目錄.....iv
圖目錄.....vi
表目錄.....vii
第一章　緒論.....1
1.1 研究背景.....1
1.2 研究動機.....2
1.3 研究目的.....3
1.4 論文架構介紹.....4
第二章　排程問題與相關文獻.....5
2.1 排程相關文獻.....5
2.2 解的表達法相關文獻.....6
2.3 禁忌搜尋法、基因演算法和蟻群最佳化演算法.....8
2.3.1 禁忌搜尋法.....8
2.3.2 基因演算法.....9
2.3.3 蟻群最佳化演算法.....10
2.4 本章小結.....12
第三章　排程問題與新舊表達法.....13
3.1 排程問題.....13
3.2 兩種解表達法.....15
第四章　禁忌搜尋法與結果分析.....18
4.1 禁忌搜尋法的流程與結果.....18
4.1.1 禁忌搜尋法的進化流程.....18
4.1.2 TS_S_{new} 和 TS_S_{old}.....21
4.2 禁忌搜尋法的實驗情境與結果.....22
4.3 禁忌搜尋法的結果分析.....25
4.3.1 禁忌搜尋法的進化流程整理.....25
4.3.2 迴圈特性的實驗.....26
4.3.3 TS_S_{new} 與 TS_S_{old} 的迴圈特性.....28
第五章　基因演算法與結果分析.....32
5.1 基因演算法的流程.....32
5.1.1 GA_S_{old} 的進化流程.....32
5.1.2 GA_S_{new} 的進化流程.....37
5.2 基因演算法的實驗情境與結果.....37

5.3 基因演算法的結果分析.....40
第六章　蟻群最佳化演算法與結果分析.....44
6.1 蟻群最佳化演算法的流程.....44
6.1.1 *ACO_S*$_{new}$ 的進化流程.....44
6.1.2 *ACO_S*$_{old}$ 的進化流程.....48
6.2 蟻群最佳化演算法的實驗情境與結果.....49
6.3 蟻群最佳化演算法的結果分析.....52
第七章　*GA_TS* 進化演算法與結果分析.....56
7.1 *GA_TS* 演算法的流程.....56
7.2 *GA_TS* 進化演算法的實驗情境與結果.....58
第八章　結論與未來研究方向.....62
8.1 結論.....62
8.2 未來研究方向.....63
參考文獻.....65
英文文獻.....65
中文文獻.....67

圖目錄

圖 2.1 多區段設計的染色體.....7

圖 2.2 單區段設計的染色體.....7

圖 3.1 固定序列之流線型單元製造系統.....13

圖 3.2 傳統的染色體表達法 (S_{old}). 16

圖 3.3 新的染色體表達法 (S_{new}).....17

圖 4.1 迴圈特性的示意圖.....26

圖 5.1 PBX 交配運算模組【資料來源：戴邦豪(2010)】.....33

圖 5.2 交換運算模組【資料來源：戴邦豪(2010)】.....34

圖 5.3 插入運算模組【資料來源：戴邦豪(2010)】.....34

圖 5.4 GA_S_{old} 在 MSU9 不同終止條件(T_f)下的結果 40

圖 5.5 MSU9 的問題中比較 GA_S_{old} 和 GA_S_{new} 在不同終止條件下的解品質 42

圖 6.1 蟻群最佳化演算法的例子.....45

圖 6.2 組合式的蟻群最佳化演算法網絡.....48

圖 6.3 比較 ACO_S_{new} 和 ACO_S_{old} 的τ_{ij}在不同的 T_f.....54

圖 6.4 比較 ACO_S_{new} 和 ACO_S_{old} 的解品質在不同的 T_f.....54

圖 7.1 *GA_TS* 進化演算法的流程 57

表目錄

表 2.1 製造單元系統排程文獻.....6

表 4.1 禁忌搜尋法的實驗結果.....23

表 4.2 迴圈特性實驗的結果.....29

表 4.3 不同 p 值的實驗結果.....31

表 5.1 基因演算法的實驗結果.....39

表 5.2 GA_S_{old} 的同質化結果.....41

表 5.3 GA_S_{new}、$GA\_S_{new\text{-}old}$ 和 $GA\_S_{old\text{-}new}$ 的實驗結果【資料來源：戴邦豪(2010)】.....43

表 6.1 蟻群最佳化演算法的實驗結果.....51

表 6.2 MSU9 情境下一個實例的 ACO_S_{old} 從至圖.....53

表 6.3 蟻群最佳化演算法的實驗結果.....55

表 7.1 *GA_TS* 演算法的實驗結果.....60

表 7.2 各種進化演算法的實驗結果比較.....61

第一章 緒論

1.1 研究背景

在眾多不同的領域當中，都會遇到解空間搜尋問題；過去二十年來，針對解空間搜尋問題已經有很多的文獻做深入的探討。過去的文獻大概可以劃分成三類：(1) 數學規劃求解法(mathematic programming)；(2) 啟發規則演算法(heuristic rules)；(3)進化演算法(meta-heuristic algorithms) (Lin *et al.* 2009a)。

數學規劃求解法通常是利用數學理論來做求解的過程，可以找到最佳解；例如：線性規劃、分支定界法、動態規劃法...等。若研究問題是屬於整數規劃問題(integer program)，在求解空間規模小時，數學規劃求解法可以很有效率的使用。但在求解空間規模很大時，數學規劃求解法所需的求解時間會快速地上升，超出合理的求解時間範圍(Lin *et al.* 2009a)。因此在求解大規模問題時，數學規劃求解法就不適合使用，許多學者因而發展其他演算法。

啟發規則演算法(heuristic rules)是根據研究問題的特定性質所發展出來的直覺規則，像是生產排程法則常用的 SPT (shortest processing time)、CR (critical ratio)、EDD (earliest due date)…等。這些方法都是根據某些特定的目標函數所發展出來的法則，只能針對比較簡單的問題。當問題的複雜度上升時，此種方法沒辦法保持原來設想的效果，無法保證解的品質；因此求出的目標值只能成為作為一個參考解，而非最佳解。當問題複雜度越高時，此種求解方法的效果會越來越不理想，因此有進化搜尋演算法的發展。

進化演算法有許多的優點。首先、進化演算法不容易受到目標函數複雜度的限制，因此可應用在複雜的空間搜尋問題。第二、進化演算法在求解大型問題時，求解時間比數學規劃求解法的時間縮減很多。第三、進化演算法的演算流程容易做改善與修正，可以根據問題的特性做改善與修正。由於許多空間搜尋問題複雜

度很高，這幾年很多學者都大量的使用進化演算法來求解問題。

進化演算法通常是根據不同的自然現象、社會學或人類思考模式...等所發展而來。像是基因演算法(GA; genetic algorithm)、蟻群最佳化演算法(ACO; ant colony optimization)、粒子群演算法(PSO; particle swarm optimization)與模擬退火法(SA; Simulated Annealing)，都是根據自然現象所發展出來的進化演算法；而瀰集演算法(MA; memetic algorithm)則是以社會學的角度所研發出來的演算法；而禁忌搜尋法(TS; tabu search)是由人類思考模式所衍伸出來的進化演算法。

這些進化演算法中，原始的基因演算法和瀰集演算法具有全域搜尋的效果；而原始的蟻群最佳化演算法、粒子群演算法、模擬退火法和禁忌搜尋法是屬於區域搜尋的進化演算法。而所謂全域搜尋是指此類進化演算法有較好的跳脫進化機制，不容易讓進化過程陷入區域最佳解之中。而區域搜尋的進化演算法就欠缺這樣的跳脫進化機制，因此會比較容易陷入區域最佳解。但以求解時間而言，全域搜尋的進化演算法需要較長的時間，而區域搜尋的進化演算法通常所需時間相對較短。

這些進化演算法的求解過程都很類似，都是利用一組初始解來產生更好的一組新解，而此新解有很高的機率會成為下一個初始解。而這樣產生新解的過程稱為進化機制(evolutionary mechanism)，而執行這樣的過程一次通常稱為一個世代(generation)。所以，進化演算法也可以稱為是有多世代的進化機制演算法(multiple-generation evolutionary algorithms)。

1.2 研究動機

過去跟進化演算法相關的研究大部分都著重在兩個部分：應用與進化機制的改善。由於進化演算法可以應用在不同的領域上，所以過去一部分研究著重在如何應用這些進化演算法在不同的新問題。而另一部分是針對進化演算法的缺點做

改進，這些研究大部分是針對進化機制做改進；例如：結合全域搜尋與區域搜尋的互補關係，衍伸出混合式進化演算法，瀰集演算法的進化機制可以算是一種混合式進化演算法。

過去研究大多是用相同的解表達法來解同樣的問題，很少文獻對解的表達法(solution representation)做相關的討論。雖然有極少部分的研究去比較不同解表達法的績效，但都沒有針對為何改變解表達法會改善解品質的原因做詳細分析。有鑑於此，本研究重點有二：第一、擬探討解表達法會影響解品質的原因；第二、擬探討同時改善進化機制與解表達法對解品質的影響。

由於過去進化演算法的相關研究頗多將問題的解(solution)稱為染色體(chromosome)，為便於陳述，本論文以下將「解」與「染色體」視為同一名詞，在文中交互採用。

1.3 研究目的

本研究以一個流線型製造單元系統的排程問題(permutation manufacturing-cell flow shop scheduling problem，簡稱 PMFS)為研究範疇，討論解的表達法對解品質的影響。上述排程問題，過去大部分研究都使用同一解表達法（簡稱 S_{old}）；而巫木誠教授(2010)提出一個新的解表達法(簡稱 S_{new})，過去有一些研究延用此種表達法，例如：戴邦豪(2010)、林耿漢(2011)與李奕勳(2011)分別應用此 S_{new} 表達法於基因演算法(genetic algorithm)、禁忌搜尋演算法(tabu search)與蟻群最佳化演算法(ant colony optimization)，實證結果發現採用 S_{new} 確實會比採用 S_{old} 的解品質好。

本論文研究目的有兩大類，第一大類、針對過去研究在不同演算法：基因演算法、禁忌搜尋演算法與蟻群最佳化演算法的實驗結果，探討 S_{new} 為何會比 S_{old} 表現較佳的原因。第二大類、討論同時改善進化演算法（混用 GA 與 tabu)與改

善解表達法(採用 S_{new})對解品質的影響。

1.4 論文架構介紹

本論文接下來的章節安排如下：第二章為文獻探討，包含三個部分：(1) 分析流線型製造單元系統排程的文獻，(2) 分析改善解表達法的相關文獻，(3) 分析禁忌搜尋法與基因演算法的相關文獻。而第三章是探討流線型製造單元系統排程的問題與介紹兩種表達法(S_{new} 和 S_{old})。第四章是探討禁忌搜尋法，將禁忌搜尋法的流程與結果做詳細的描述，並針對實驗結果做詳細的分析與驗證。第五章是探討基因演算法，將基因演算法的流程與結果做詳細的描述，並分析其實驗結果。本研究在第六章討論蟻群最佳化演算法的流程與結果，並分析其實驗結果。本研究在第七章提出一個新的進化演算法（混用基因演算法和禁忌搜尋法），介紹其演算流程並分析實驗結果。同時也討論使用混合式解表達法對解品質的影響。第八章是結論與未來研究方向。

第二章　排程問題與相關文獻

本章分成四節，第一節介紹本研究所討論的排程問題的相關文獻；第二節介紹過去不同解表達法求解排程問題的相關文獻；第三節描述禁忌搜尋法、基因演算法與蟻群最佳化演算法的基本概念；第四節是本章小結。

2.1 排程相關文獻

過去已有許多的相關文獻求解類似排程問題，可須區分成兩大類：啟發規則演算法與進化演算法。使用啟發規則演算法的文獻介紹如下： Schaller *et al.* (2000)首次提出了本研究的排程問題，並設計了這個問題的實驗情境。除了比較之前相關的求解演算法之外，最後還提出幾個有效的演算法來進行分析比較。Reddy & Narendran (2003)同樣也是運用演算法來求解排程問題，然而其討論的排程問題是工件順序相依整備時間而非本研究所討論的工件族順序相依整備時間，其決策只有工件排程而不像本研究有兩個決策。

此排程問題採進化演算法求解的文獻可以分類如下（參閱表 2.1）。採用禁忌搜尋法過去有兩個相關文獻：Logendran *et al.* (2006)是利用三種不同的產生初始解演算法搭配禁忌搜尋法來求解類似的問題，但其問題是屬於彈性流線型生產環境，亦即每個加工階段是屬於平行多機台。結果顯示在平行機台越多時，使用不同初始解演算法會有明顯的差異。Hendizadeh *et al.* (2008)發展改良式的禁忌搜尋法來求解此排程問題，並和過去文獻做比較發現有較好的績效且計算時間更短。Franca *et al.* (2005)比較基因演算法和瀰集演算法搭配階層式區域搜尋法選擇母代求解此問題的績效，其結果顯示搭配階層式區域搜尋法選擇母代的法則會有比較好的績效。Lin *et al.* (2009a)以多種不同演算法：基因演算法、禁忌搜尋法和模擬退火法。Lin *et al.* (2009b)以模擬退火法為基礎，為了不掉落區域最佳解，該研究使用發展一修正機制，結果顯示比過去相關的進化演算法有更好的績效。

表 2.1 製造單元系統排程文獻

主要概念	作者(年份)	演算法	目標值
製造單元系統	Reddy & Narendran (2003)	Heuristic Rules	makespan
	Schaller *et al*. (2000)	Heuristic Rules	makespan
	Logendran *et al.*(2006)	Tabu search	makespan
	Franca *et al*. (2005)	Genetic algorithm, Memetic algorithm	makespan
	Hendizadeh *et al*. (2008)	Tabu search	makespan
	Lin *et al*. (2009a)	Tabu search, Genetic algorithm, Simulated annealing	makespan 與其他五個目標值
	Lin *et al*. (2009b)	Simulated annealing	makespan

2.2 解的表達法相關文獻

過去求解排程問題大部分研究都是使用現有的解表達法，很少文獻會設計新的解表達法。而設計這些新表達法所研究都有一個共通點—所求解的排程問題都是屬於多維度決策的問題，也就是有多個決策維度(決策向量)需要決定，而這些維度的結果又會互相影響。

而過去相關文獻的解表達法方式又可區分為兩種類別：(1) 多區段設計；(2) 單區段設計。而為了便於解釋這兩種類別，本研究將以一個簡單的排程問題為內容。首先考慮一個有 N 個工件需要在 M 台平行機台加工的情境，且每個工件只有一個作業。我們有兩個決策需要決定：(1) 工件指派—每個工件需要指派到哪一個機台上加工？(2) 工件排序—在特定的機台上，工件應該如何排序？因此，此問題有兩個決策維度(決策向量)。

在多區段的設計類別中，某些相關文獻會將上述問題區分成兩個區段(子染色體)，而每個子染色體表示一個決策向量(M'Hallah & Al-Khamis 2011)。如圖 2.1 所示，兩個子染色體是互相連在一起，但分別表示不同的決策向量。在子染色體 1 中的每個元素代表工件，而此子染色體代表了工件的排序。相反地，在子染色體 2 中的每個元代表機台，也就表示每個工件的機台指派決策。

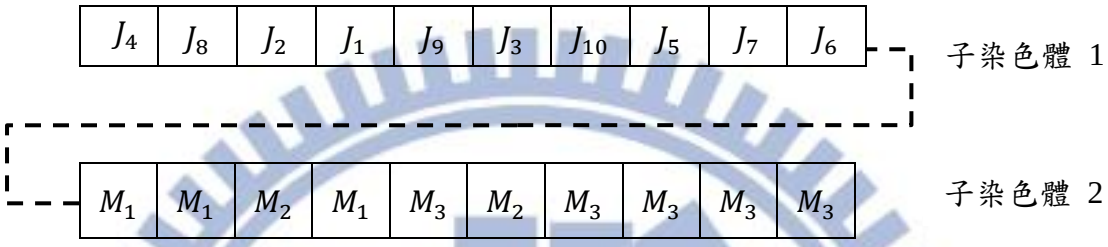


圖 2.1 多區段設計的染色體.

相反地，其他相關文獻就上述問題將染色體設計成單區段來表示此兩個決策向量(Chan *et al.* 2011; Tavakkoli-Moghaddam *et al.* 2009)。此種設計如圖 2.2；此種染色體中表示了工件的排序，且鑲入了兩個「＊」符號表示兩個切割點。因此，此染色體會被區分成三個類別，分別表示指派到不同的機台上，也獲得了機台的指派決策。而在特定機台上的工件的排序則可以靠原染色體上的工件排序來做為依據。

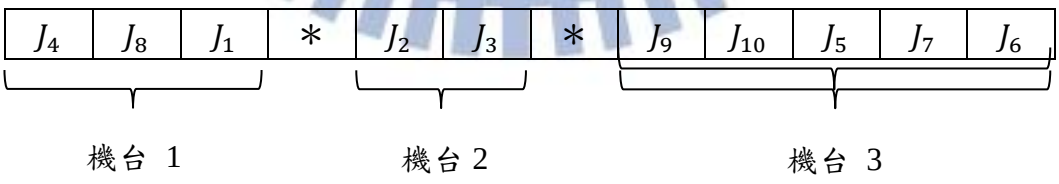


圖 2.2 單區段設計的染色體.

這兩設計的主要區別在於多區段設計是屬於一對一對應的模式；也就是說每一個決策向量就設計成一個區段。最後再將每個區段做整合來獲得排程問題的解。而過去的研究大都採用這樣的方式因為這種方法最為直覺且簡單。相反地，單一

區段設計是屬於多對一對應的模式;也就是多個決策向量設計成單一一個區段。此種設計讓染色體包含了多個不相容的資訊內容,因此我們需要設計相對應的解碼方法從單一區段中獲取多個決策向量的資訊。舉例來說,圖 2.3 中的「＊」符號就是一個新奇的設計來從單一區段獲取兩個決策向量的資訊。為了使用此種單區段設計,我們除了需要設計簡潔的染色體之外,還需要問題導向的解碼方法。因此,不同問題需要不同的解碼方法;也因為其複雜性,大部分的研究都使用過去使用過的染色體表達法。

然而經過本研究整理相關的文獻,發現不同的排程問題需要設計不同的解表達法才能應用。因此,本研究無法應用過去不同的解表達法來求解本研究的排程問題;而本研究所探討的排程問題過去只有傳統的解表達法,因此本研究發展新的解表達法來求解此排程問題。

2.3 禁忌搜尋法、基因演算法和蟻群最佳化演算法

由於戴邦豪(2010)、林耿漢(2011)和李奕勳(2011)已經分別應用 S_{new} 於基因演算法、禁忌搜尋法和蟻群最佳化演算法,這些過去研究的實驗結果都發現採用 S_{new} 比採用 S_{old} 為佳。有鑑於此,本節將簡單描述禁忌搜尋法、基因演算法與蟻群最佳化演算法的基本原理與應用。

2.3.1 禁忌搜尋法

禁忌搜尋法式由 Glover (1989;1990)所提出的進化演算法,其原理是來自於人在做決策時會考慮經驗法則,做出下一次決策方向的思考模式。而禁忌搜尋法已經廣泛的應用到許多不同的領域,例如:路徑指派、零工式排程、流線型排程、旅行者問題與流線型製造單元系統排程問題...等。而其搜尋方式是先從設定一個

初始解當參考點開始進行搜尋，搜尋此點附近的所有鄰近解，從所有鄰近解中找到一組最佳的鄰近解當作是下一個參考點，並重複這樣的步驟直到搜尋條件達成才停止。但為了避免選到同樣的參考點，因此有禁忌列表的產生，此列表記錄著每次交換的結果，並禁止下一次在選到同樣的交換，由此來避免選到重複的參考點。而其在本研究中的詳細步驟將在第三章做描述。

過去許多的禁忌搜尋法的相關文獻屬於應用或改善進化機制。有關應用的部分相關文獻如下：Wen & Huang (1996)運用禁忌搜尋法來求解多階層決策的問題，也就是每位決策者的決策有階層區分，結果顯示比其他啟發規則演算法來的有效且求解時間也較短。Kim *et al.* (1997)利用禁忌搜尋法來建立一個非階層且非集中式的在線視頻點播網路架構，其問題本身除了考慮架設成本外，還需考慮連結成本與程式儲存成本。其結果顯示使用禁忌搜尋法有較好的績效與效率。屬於改善進化機制與應用的部分相關文獻如下：Cordeau & Maischberger (2012)運用平行化的禁忌搜尋法來求解不同類型的車輛途程問題，所謂平行禁忌搜尋法就是改變原本的進化機制，同時考慮多個初始參考點，在進化過程有一定機率可以互相取代，來避免結果受到初始解影響；除此之外還考慮搭配其他區域最佳解，起結果顯示其提出的方法比單單使用禁忌搜尋法績效來的好，而使用平行化禁忌搜尋法可以更有效地找到品質不錯的解。Demir *et al.* (2012)提出改良式的禁忌搜尋法來求解緩衝區的相關問題，其問題主要探討緩衝區的產能和擺放位置。其改良方式是提出三種不同初始解的產生方式與鄰近解搜尋過程的微調機制來改善禁忌搜尋法。

2.3.2 基因演算法

基因演算法是由 Holland (1975)所提出的進化演算法，此方原理是來自於遺傳學上的演化機制而來。而其流程主要是先產生一群母代染色體，其組成元素稱

為基因。然後透過改變基因的組合來獲得更好的染色體，如此不斷的進化來達成物競天擇適者生存的效果。而其改變基因的組合主要有兩種方式：交配與突變。交配主要原理是基於相信良好基因的染色體所組合而成的子代會有良好的績效。而突變是基於染色體經過跳脫式的組合可能會有更好的績效，所以也不能忽略。其在本研究中的詳細流程將在第五章描述。

基因演算法已經非常廣泛的使用在各個領域之中，其數量之多已無法詳細一一描述，因此本研究只針對本研究所討論的排程問題之基因演算法相關文獻做描述。Franca *et al.* (2005) 改良了基因演算法的進化機制，搭配階層式區域搜尋法選擇母代獲得更好的解品質。而 Lin *et al.* (2009a)則是應用基因演算法來求解不同的問題，其目的是比較若本研究的問題分別屬於不固定順序時與固定順序時之績效差異。結果顯示顯示不固定順序會有較好的績效。但是此研究有一缺失，就是其不固定順序的某一初始解來自於固定順序的最佳解；此作法當然會保證不固定順序一定會比固定順序來的好。

2.3.3 蟻群最佳化演算法

螞蟻系統演算法(ant System)是由 Dorigo & Gambardella (2002) 基於自然界螞蟻行為所延伸出來的一種進化演算法。Dorigo *et al.* (2002) 之後修正螞蟻系統演算法太快收斂，較容易落入區域最佳解的缺點，提出改良後的螞蟻演算法(ant colony system)。Dorigo 之後將這一系列演算方法命名為蟻群最佳化演算法。

蟻群最佳化演算法的基本概念是取自螞蟻外出覓食的自然行為。在覓食的過程中，螞蟻會在行經巢穴與食物間的路徑上留下費洛蒙(pheromone)，其目的是為後來出發的螞蟻留下路徑的標記。而在覓食的初期，螞蟻會四處隨機走動，並留下費洛蒙。當一隻螞蟻遇到許多不同費洛蒙的路徑時，會判斷並選擇一條路徑前進。而此螞蟻會在其經過的路徑會留下費洛蒙，進而使該路徑的費洛蒙濃度增

高。而沒經過的路徑，會隨時間的流逝而逐漸蒸發，讓其路徑上的濃度降低。

而螞蟻在路徑的選擇上，主要是取決於路徑上的費洛蒙濃度。路徑上的濃度愈高愈能吸引螞蟻選擇。因此，費洛蒙是螞蟻間選擇路徑的溝通工具。除此之外，由於行經的路徑較短的螞蟻會較快地往返，使路徑上單位時間內所留下的費洛蒙較多。相反地，路徑較長也會使往返的時間也較長，其單位時間內所留下的費洛蒙就會比較少，進而使路徑上的費洛蒙濃度漸漸降低，愈來愈不能吸引螞蟻選擇。經過一段時間後，此起彼落的加強作用下，很快地所有螞蟻便會使用較短的路徑去覓食。

過去蟻群最佳化演算法多應用於求解流程式(Flowshop)生產排程問題。Merkle & Middendorf (2005)利用蟻群最佳化演算法搭配論文提出的隨機螞蟻概念來求解具固定序列特性之單機台流程式製造排程問題。因為在傳統蟻群最佳化演算法中，初始解對之後的解品質有非常大的左右性，所以在論文中設置了隨機螞蟻機制，其在尋找下一個造訪點機制上有別於一般的人工螞蟻，主要用來再加強蟻群最佳化演算法的全域搜尋的能力，減輕初期求解對後期的左右性，減慢求解的收斂速度。Gajpal & Rajendran (2006)提出新蟻群最佳化演算法來求解具固定序列特性之流程式製造排程問題，其特點在於能在求解最小化完成時間變異的指標中能比其它演算法表現的好。Chang *et al.* (2008)提出了 Generating artificial chromosomes for Genetic Algorithm (ACGA)來求解具固定序列特性之單機台流程式生產排程問題，來避免因為蟻群最佳化演算法的揮發概念造成陷入停滯的現象。其特點是在基因演算法的機率模型中加入蟻群最佳化演算法的揮發機制，此機制讓產生新染色體時更能發現其它可能的解(全域搜尋)，進而達到更好的解品質。而 Yagmahan & Yenisey (2010)提出利用多目標蟻群最佳化演算法來求解多目標的單機台的排程問題。其特點在於結合兩種目標值來建構費洛蒙路徑及加入類似禁忌搜尋法的區域搜尋概念。

2.4 本章小結

本章結論如下，過去進化演算法都是著重在應用與改善進化機制，甚少文章提到採用不同解表達法對解品質的影響。本研究所提出的解表達法(S_{new})，戴邦豪(2010)、林耿漢(2011)與李奕勳(2011)已經分別應用 S_{new} 於基因演算法、禁忌搜尋法和蟻群最佳化演算法，實驗結果都發現採用 S_{new} 比採用 S_{old} 為佳，但卻不知道原因為何。在本論文，吾人先針對這三篇論文做研究，分析出 S_{new} 比 S_{old} 為佳的原因。後續再討論一個新的演算法（混用基因演算法和禁忌搜尋法），介紹其演算流程並分析實驗結果；同時也討論使用混合式解表達法對解品質的影響。

第三章　排程問題與新舊表達法

本章節首先介紹本研究所探討的排程問題並介紹此排程問題的兩種解表達法(S_{new} 和 S_{old})。

3.1 排程問題

本研究所討論的排程問題稱為「順序相依整備時間且固定序列之流線型單元製造系統排程」，此排程問題過去已有很多文獻發表 (Schaller *et al.* 2000)。此排程問題的結構如圖 3.1，茲將其諸排程特性列述如下。

第一、此排程問題具有流線型生產(flow shop)的特性；亦即所有工件都有相同的製造流程。在此製造流程中的每一個階段都只有一台機台，而且每個階段之前都有一個無限產能的緩衝區(buffer)來存放等待加工的工件。

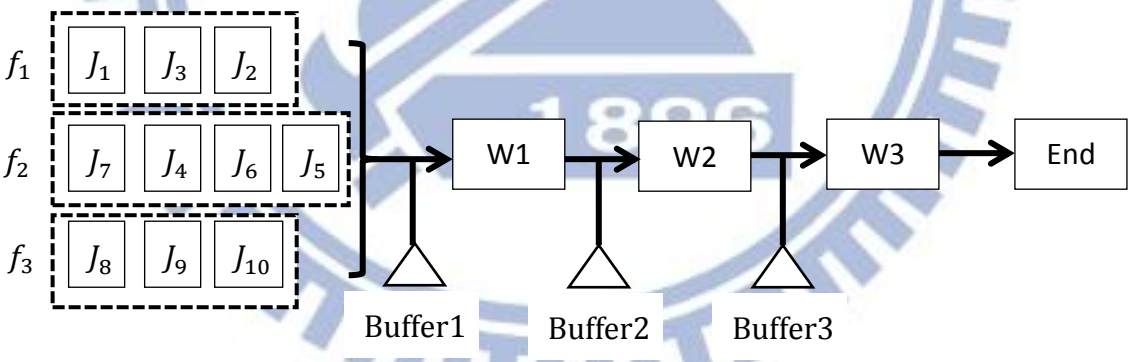


圖 3.1 固定序列之流線型單元製造系統.

第二、此排程問題的環境是一個製造單元系統，因此具有家族導向排程(family-based scheduling)的特性。製造單元系統是依據群組技術(group technology)的原理而產生。在這樣的應用下，會先將所有工件依工件的製造性質進行分類，將有相似製造性質的工件歸類成同一個工件族；亦即轉換生產相同工件族(same

family)的工件時不需要整備時間(no setup)。相反的，當加工不同工件族時就會需要明顯的整備時間。所以加工會以工件族(job family)為單位進行加工，以節省整備時間。亦即當某一工件族開始加工後，會等待該工件族都加工完成才會加工另一個工件族中的工件。再者，每個工件具有獨立運輸的特性；也就是說當每個工件加工完畢後會立即獨立的運輸到下一個階段的緩衝區，並不需要等待全部的工件族完成才運送。

第三、家族之間的整備時間具有順序相依的特性。如前所述，不同工件族轉換的時候需要整備時間，此整備時間具有順序相依(sequence dependent)；也就是說整備時間會依據不同的工件族順序而不一樣。會造成這樣的現象，主要是因為每個工件族之間的相似性不同；若前後兩個工件族的加工性質較為相近會比部相近的整備時間來的短。例如：工件族F_1和F_2比較相近，F_1和F_3比較不相近，$F_2 \rightarrow F_1$的整備時間就會比$F_3 \rightarrow F_1$來的短。

第四、此情境具有固定序列的特性；固定序列是指工件在工件族中的排序在每一個加工階段都相同，不會因為加工階段不同而有不同的加工排序。此外，此問題情境還包含不當機的特性，因為機台非常的穩健所以不會有當機的問題。

在上述的特性之下，此排程問題有兩個重大的排程決策：工件在工件族中的排序決策(job sequencing within each family)與工件族之間的排序決策(family sequence)。因為此問題情境有獨立運輸的特性才會導致工件在工件族中的排序有其重要性；相反地，若沒有此特性，工件在工件族中的排序(job sequencing within each family)並不會影響整體的排程績效，也就不需要考慮此決策。因此，獨立運輸(each job is independently transported)是本問題一個很重要的特性。

此排程問題的真實案例是應用於印刷電路板 PCB(printed circuit board)製造業中的表面黏著技術 SMT(surface mounting technology)製造流程上(Schaller *et al.* 2000)。此製造流程是為了在電路板表面上插入許多的電子零件(components)，因

此製造流程中包含了一系列的 SMT 機台來執行這樣的製造流程。每一個 SMT 機台可視為一個工作站或加工階段，將特定的電子零件群組插入電路板表面，而透過整備可以更改電子零件群組的組合內容。而每一個電路板(printed circuit board) 可視為一個獨立的工件(job)來經過這一系列的加工階段進行加工。在這樣的生產情境下，如果兩個電路板所需要的電子零件很相近，在轉換加工這兩個電路板時就不需要更換電子零件群組的內容就可以進行加工，也就不需要整備時間(setup)。因此使用類似電子零件的電路板就會歸類成同一個加工族(job family)，也就造成了家族導向排程(family-based scheduling)的特性。而因為電子零件群組的重複性，會導致順序相依整備時間的特性。也就是說工件族之間電子零件群組較為接近時，所需要更換的電子零件較少整備時間就會比較短；相反地，若相差很多時整備時間就會較長因為要更換的電子零件多。

3.2 兩種解表達法

本研究的排程問題具有雙決策向量：(1) 工件在工件族中的排序，(2)工件族之間的排序。本研究的排程問題過去文獻都使用多區段設計(multiple segment design)，本研究提出一單區段設計(single segment design)的染色體表達法。為了本研究之後解釋的方便性，本研究稱舊表達法（多區段設計）為 S_{old}，稱新表達法（單區段設計）為 S_{new}；其區別與特性解釋如下。

S_{old} 為了獲得兩個決策向量，會有兩個特性：分群與多區段設計。考慮一個排程問題有 n 個工件$(J_1, J_2, \dots, J_n)$需要歸類成 k 個工件族$(f_1, f_2, \dots, f_k)$。根據多區段設計的特性，此染色體會切割成 $k+1$ 個區段，而分群是指此 $k+1$ 個區段會歸類成兩群體。第一個群體只有一個區段，表示這 k 個工件族的排序。而第二群包含 k 個區段，每個區段表示工件在工件族中的排序。舉例來說，一個排程問題有 10 個工件分成 3 個工件族。如圖 3.2 所示，第一個群體表示工件族的排序為

$f_3 \rightarrow f_2 \rightarrow f_1$；第二個群體分成三個區段，第一個區段表示工件族$f_1$包含了 3 個工件，然後它們的排序為$J_1 \rightarrow J_3 \rightarrow J_2$。而第二個和第三個區段分別表示$f_2$和$f_3$的工件在該工件族的排序。

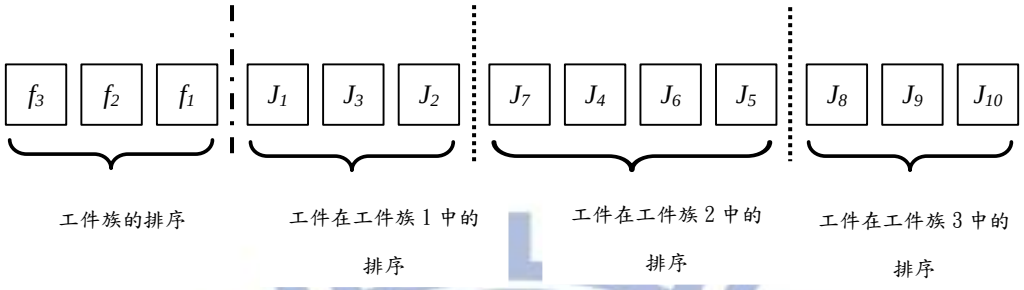


圖 3.2 傳統的染色體表達法 (S_{old}).

如前所述，新表達法只有單一個區段且還需要特定的解碼機制。因此 S_{new} 的元素只有工件，其代表工件的排序；而透過解碼機制，我們可以獲得工件族的排序。考慮一個排程問題有 n 個工件$(J_1, J_2, \ldots, J_n)$需要歸類成 k 個工件族$(f_1, f_2, \ldots, f_k)$；由染色體可以獲得這 n 個工件之間的排序。在解碼機制中，首先我們由左到右一個一個的讀取每一個工件相對應的工件族，並記錄每一個工件族第一次出現的順序，直到所有工件族都出現後停止；因此我們可以獲得工件族的排序。而工件在工件族中的排序則可以根據其在染色體相對的排序來決定。舉例來說，一個排程問題有 10 個工件分成 3 個工件族(如圖 3.3)。染色體的前四個工件分別是$J_8 \rightarrow J_7 \rightarrow J_4 \rightarrow J_1$其所對應的工件族為$f_3 \rightarrow f_2 \rightarrow f_2 \rightarrow f_1$；由此我可以得到工件族排序為$f_3 \rightarrow f_2 \rightarrow f_1$。由工件在染色體中的相對排序，我們可以很容易地獲得工件在工件族中的排序；因此，工件在工件族f_3的排序為$J_8 \rightarrow J_9 \rightarrow J_{10}$，在工件族$f_2$的排序為$J_7 \rightarrow J_4 \rightarrow J_6 \rightarrow J_5$，在工件族$f_1$的排序為$J_1 \rightarrow J_3 \rightarrow J_2$。

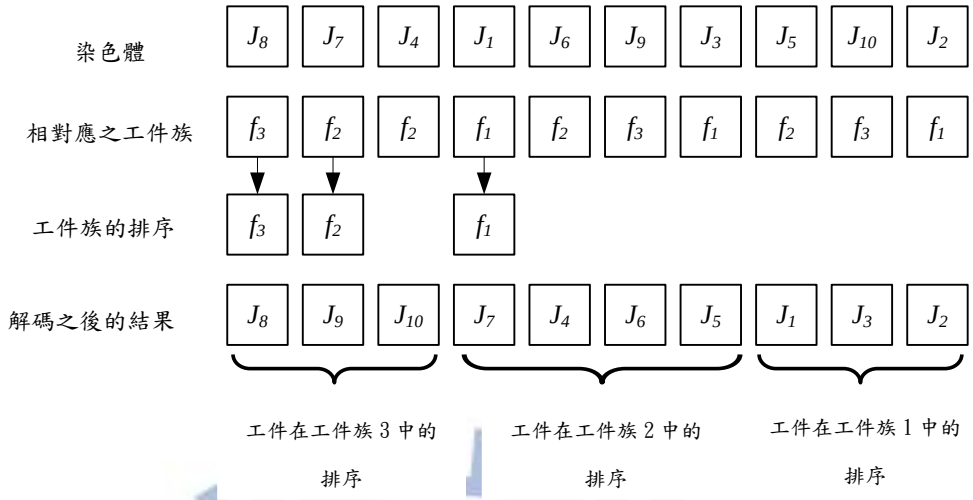


圖 3.3 新的染色體表達法 (S_{new})

第四章 禁忌搜尋法與結果分析

本章將討論禁忌搜尋法的演算流程、實驗結果與分析。而本章內容主要分成三部分：第一部分介紹禁忌搜尋法的演算流程；第二部分介紹實驗情境與結果；第三部分針對其結果作分析。

4.1 禁忌搜尋法的流程與結果

本章節可以分為三小節；第一節描述禁忌搜尋法的進化流程；第二節介紹如何使用禁忌搜尋法搭配新舊表達法(S_{new}和 S_{old})。為了方便描述，本研究將禁忌搜尋法(tabu search)搭配舊表達法(S_{old})命名為 TS_S_{old}，將禁忌搜尋法(tabu search)搭配新表達法(S_{new})命名為 TS_S_{new}。

4.1.1 禁忌搜尋法的進化流程

在禁忌搜尋法中，我們命名解為染色體，且其包含元素為基因，而這些基因都為正整數(因為其代表工件)。舉例來說，一個染色體有五個基因可以表示成 $h = (1, 5, 3, 2, 4)$。在禁忌搜尋法的進化流程中，期會重複使用兩大模組：(1) **產生鄰近解**；(2)**禁忌列表**。我們接來會先介紹此兩模組然後描述整個禁忌搜尋法的流程。

產生鄰近解：給定一個染色體 h，本研究有系統的使用交換的方式來產生一組新的鄰近解群體 $N(h)$。假設有一個染色體$h = (a_1, a_2, a_3, a_4, a_5)$；本研究有順序的將每一個$a_i$和其右邊的所有基因做交換。因此，我們可以經由將$a_1$與其他基因交換獲得 4 個新的染色體組合。依照這樣的模式，交換a_2可以得到 3 個，針對a_3有 2 個，交換a_4產生 1 個。最後 $N(h)$的總數目會是$C_2^5 = 10$個新的染色體。

禁忌列表：在交換步驟之中，交換的兩個基因(a_i和a_j)稱為移步(move 或 swap pair)，本研究以(a_i, a_j)來表示。禁忌列表是一個包含最多q_{size}移步的集合，其建置的過程是有順序性的。假設給定一個禁忌列表的大小為$q_{size} = 3$，而其內容為$\{(a_1, a_3), (a_2, a_3), (a_1, a_5)\}$。如果一個新的移步$(a_i, a_k)$要加入現在的禁忌列表之中，要遵守下列兩個規則。

規則 1：如果此移步(a_i, a_k)已經存在原本的禁忌列表名單之中時，將此移步從原本的順序移動到最後的位置來產生新的禁忌列表。舉例來說，原本的禁忌列表是$\{(a_1, a_3), (a_2, a_3), (a_1, a_5)\}$；而我們要加入的移步是$(a_2, a_3)$。我們可以獲得新的禁忌列表為$\{(a_1, a_3), (a_1, a_5), (a_2, a_3)\}$。

規則 2：如果此移步(a_i, a_k)已經不存在原本的禁忌列表名單之中時，將此移步放置在名單的最後面。如果名單總數超過q_{size}，則將名單中最前面的移步刪除，然後依序往前遞補。舉例來說，如果原本的禁忌列表是$\{(a_1, a_3), (a_2, a_3), (a_1, a_5)\}$，我們要加入的移步為$(a_5, a_3)$，我們可以獲得新的禁忌列表為$\{(a_2, a_3), (a_1, a_5), (a_5, a_3)\}$。但是如果原本的禁忌列表之中只有兩個移步$\{(a_1, a_3), (a_1, a_5)\}$時，我們可直接將要加入的移步放在列表的最後即可，因此可獲得新的禁忌列表$\{(a_1, a_3), (a_1, a_5), (a_5, a_3)\}$。

在描述完兩個重要模組之後，接下來我們要描述整個禁忌搜尋法的流程如下：

Procedure Tabu_Search

Step 1: Initialization

- Generate an initial chromosome h_0;
- Evaluate h_0;
- Set $h^* = h_0$; /* h^* is the local best solution*/
- Set $h^+ = h_0$; /* h^+ is the global best solution*/
- Set $T = 0$; /*age of h^+, global best solution*/

- Set *Tabu_list* = ϕ; /* Tabu_list is initially an empty set*/

Step 2: Generate and Sort $N(h^*)$

- Generate $N(h^*)$; set $T = T+1$;
- Evaluate each chromosome in $N(h^*)$;
- Sort all chromosomes in $N(h^*)$ according to their performances;
- Name the sorted results as $\{h_1, h_2, \ldots, h_w\}$, in which h_i is better than h_{i+1} in terms of performance;
- Name the swap pair that generates h_i from h^* as ω_i;

Step 3: Update h^*, h^+, and Tabu_list

- Set $i = 1$;
- While ($i \leq w$) /*w is the total number of chromosomes in $N(h^*)$*/

  If $\omega_i \notin Tabu\_list$,

  then place ω_i in the Tabu_list; set $h^* = h_i$;

  If h_i is better than h^+, then set $h^+ = h_i$ and $T = 1$;

  Go to Step 4;

  Else $(\omega_i \in Tabu\_list)$,

  If h_i is better than h^*,

  then place ω_i into Tabu_list; set $h^* = h_i$;

  If h_i is better than h^+,

  then set $h^+ = h_i$ and $T = 1$;

  Go to Step 4;

  Else $i = i + 1$

  If $i = w + 1$, then place ω_1 in the Tabu_list; set $h^* = h_1$;

  Go to Step 4;

  Endwhile

Step 4: Termination Check

- If $T > T_f$, output h^+ and STOP; /*T_f is a predefined large number*/
- Else, Go to Step 2.

4.1.2 TS_S_{new} 和 TS_S_{old}

TS_S_{new} 的進化流程跟原本的禁忌搜尋法幾乎完全一樣，然而在評估染色體的績效時會有些許的修正。也就是在評估績效前要先對 S_{new} 染色體做解碼的動作，經過解碼機制之後才會產生完整的解資訊。因此，禁忌搜尋法只需要修正在每次要評估染色體之前對其先進行解碼的動作即可，其他相關流程與步驟都完全相同。

然而在 TS_S_{old} 之中，禁忌搜尋法必須要有兩個修正：**(1) 產生鄰近解；(2) 多個禁忌列表**。

產生鄰近解：給定一個參考解 h^*，我們想要產生一組鄰近解 $N(h^*)$。在 TS_S_{old} 之中，我們採用「**單一區段改變**」的方式，也就是說當我們在交換某個區段的基因時，其他區段的基因保持不變。舉例來說，如果有一個染色體有三個區段如：$h^* = s_1\_s_2\_s_3$，當我們交換s_3之中的基因時，s_1和s_2之中的基因組合都不會做更動。假設每個區段都有 5 個基因時，我們可以針對每個s_i得到$C_2^5 = 10$個新的染色體。因此所有 $N(h^*)$的總數為$3 \times C_2^5 = 30$。

多個禁忌列表：在 TS_S_{old} 之中，染色體屬於多區段設計，所以每一個區段都有自己唯一的禁忌列表，而且都獨立各自更新互不影響。由於移步的產生是只交換某一個區段的基因所產生的，因此更新禁忌列表就變成各自獨立更新而不會有同時兩個禁忌列表同時更新的現象發生。舉例來說，給定一個有 3 個區段的染色體$h^* = s_1_s_2_s_3$，而每個區段相對應的禁忌列表為T_1, T_2, T_3。假設h_i為一個從 $N(h^*)$挑出的新染色體，而此新染色體是交換第二個區段s_2裡的基因所產生的。因此我們只會新禁忌列表T_2，而不會變動其他兩個禁忌列表。

4.2 禁忌搜尋法的實驗情境與結果

此小節將描述實驗情境與結果，而其研究的目標值是最大完工時間(makespan)，比較 TS_S_{old} 和 TS_S_{new} 的績效表現。而其禁忌搜尋法的相關參數設定為$T_f = 2{,}000$且$q_{size} = C_2^{n_s} \times p$；其中$q_{size}$為各區段相對應的禁忌列表儲存容量，而機率$p = 0.3$，$n_s$表示該區段的所有基因數目總和。此兩種演算法流程都是使用 C++語言撰寫，而實驗的電腦配備為 AMD Athlon(tm) Ⅱ*4640 3.0Ghz CPU 以及 4G Ram。

實驗資料的來源是跟據 Schaller *et al.* (2000)的文獻而來，其分成 30 個不同的情境，而每個情境包含 30 個實例。因此會產生總共 900 個實驗實例；而每一個實例代表唯一的排程問題。在這 30 個情境中，我們以 *X-F-m* 來表示其所代表的不同情境。其中，*X* 表示不同的整備時間(LSU,MSU,SSU)，*F* 表示工件族的數目，*m* 表示機台的數目—有多少加工階段。SSU 表示整備時間最短，MSU 表示整備時間為中間值而 LSU 表示整備時間最大。舉例來說，表 3.1 中的 LSU33 表示一個大整備時間有 3 個工件族與 3 個機台的情境。

在每個情境的 30 個實例中，我們必須隨機產生以下的相關參數：每個工件族中的工件數目(n_f)、生產時間與每個工件族之間的整備時間。其中，n_f是採用間斷的單一分配(uniform)函數產生 U[1,10]；而工件在每一個加工階段的生產時間也是採用間斷的單一分配函數產生 U[1,10]。而整備時間則根據不同的整備時間大小採用不同的方案，在 SSU 的時候是採用 U[1,20]、在 MSU 的時候是 U[1,50]，而大整備時間 LSU 則是 U[1,100]。

由於初始解會受到亂數的影響而產生不一樣的初始解，進而影響結果，造成使用不同的亂數跑一樣的實驗會有不一樣的結果。為了避免實驗受到亂數的影響，因此每一個實例都需要用 15 個不同的亂數種子(seed)來進行實驗。將此 15 個結果取平均做為此實例的績效，然後再將每個情境的 30 個實例平均起來做為最後

評估的績效值。所以此研究為了比較此兩個方法需要做 27,000 次的實驗。

表 4.1 禁忌搜尋法的實驗結果

Scenario	*Makespan*							Computation Time	
	N	***Nw+Ne***	***Ne***	***Nw***	***Cn***	***Ct***	***γ (%)***	***Tn***	***Tt***
SSU33	30	30	3	27	134.48	137.27	2.04	0.94	0.33
SSU34	30	30	1	29	151.10	153.98	1.92	1.40	0.42
SSU44	30	30	0	30	186.55	191.57	2.65	2.64	0.61
SSU55	30	30	0	30	243.25	250.31	2.81	7.01	1.05
SSU56	30	30	0	30	254.87	262.76	3.04	7.30	1.17
SSU65	30	30	0	30	288.18	296.44	2.79	12.35	1.50
SSU66	30	30	0	30	296.83	305.63	2.88	13.43	1.66
SSU88	30	30	0	30	407.41	419.07	2.77	43.40	3.50
SSU108	30	30	0	30	489.27	504.36	2.99	95.58	5.30
SSU1010	30	30	0	30	530.67	546.58	2.90	128.91	6.83
MSU33	30	30	6	24	160.00	162.41	1.41	0.73	0.28
MSU34	30	30	3	27	184.71	187.60	1.53	1.09	0.38
MSU44	30	30	0	30	238.34	245.51	2.95	2.71	0.60
MSU55	30	30	0	30	309.52	318.11	2.70	6.09	1.03
MSU56	30	30	0	30	319.91	329.14	2.85	6.02	1.08
MSU65	30	30	0	30	366.90	378.18	3.00	10.96	1.43
MSU66	30	30	0	30	385.34	398.26	3.23	12.25	1.64
MSU88	30	30	0	30	521.90	540.04	3.36	36.51	3.30
MSU108	30	30	0	30	640.46	661.39	3.16	86.43	5.42
MSU1010	30	30	0	30	672.82	692.26	2.80	94.50	6.12
LSU33	30	30	4	26	228.09	233.21	2.24	1.06	0.37
LSU34	30	30	4	26	239.06	243.85	2.05	0.88	0.31
LSU44	30	30	0	30	324.89	333.18	2.62	2.41	0.55
LSU55	30	30	0	30	421.59	434.62	3.03	5.70	0.96
LSU56	30	30	0	30	442.82	455.67	2.84	6.84	1.13
LSU65	30	30	0	30	500.06	518.11	3.48	10.63	1.47
LSU66	30	30	0	30	524.49	540.73	3.02	10.93	1.50
LSU88	30	30	0	30	722.17	745.71	3.18	33.00	3.18
LSU108	30	30	0	30	880.50	910.78	3.31	71.70	5.02
LSU1010	30	30	0	30	935.79	968.47	3.37	85.92	6.10
Average	**30**	**30.00**	**0.70**	**29.30**	**400.07**	**412.17**	**2.76**	**26.65**	**2.14**

表 4.1 顯示了這 30 個情境的結果。而其參數解釋如下：*TS_S*$_{old}$ 所獲的的績效為C_t，*TS_S*$_{new}$ 所獲的的績效為C_n；而其對應的計算時間分別為T_t和T_n。而為了比較此兩種演算法的績效，此研究設定一個參數$\gamma = (C_t - C_n)/C_t$。除此之外，此研究還比較每個情境之中，新舊表達法在實例上的表現。$N = 30$表示每個情境有 30 個實例，而N_e表示$\gamma = 0$的實例個數，而N_w代表$\gamma > 0$的實例個數。$N_w + N_e$為 *TS_S*$_{new}$ 與 *TS_S*$_{old}$ 比較勝過或平手的總數量。如果$N_w + N_e$越大表示 *TS_S*$_{new}$ 有明顯的贏過 *TS_S*$_{old}$。

由表 4.1 可得到在這 30 個情境下$N_w + N_e = N$，而γ最好為 3.37%最差也有 1.41%，平均為 2.76%；表示 *TS_S*$_{new}$ 明顯的比 *TS_S*$_{old}$ 績效來的好。而此研究也進行了統計上的檢定，其檢定方式是對 900 個實驗實例進行成對 *t* 檢定。首先計算每一個實例的績效差異$d = \frac{\overline{C_t} - \overline{C_n}}{\overline{C_t}}$，其中$\overline{C_t}$和$\overline{C_n}$分別表示這兩種演算法在此實例中的 15 實驗平均值。而 t 值可以根據公式 $t_0 = \frac{\bar{d}}{S_d/\sqrt{n}}$ 獲得，其中$\bar{d}$與S_d分別表示這 900 個實例的平均與變異數。因此此研究得到的 t 值為$t_0 = 50.05 > t_{0.025,899} = 1.96$，這表示 *TS_S*$_{new}$ 在 95%的信心水準下明顯的比 *TS_S*$_{old}$ 績效好。

除此之外，在表 4.1 中，T_t和T_n分別表示此兩種演算法的計算時間。然而我們可以發現其計算時間都只需要幾分鐘，在實務上是可以接受的；因此在此研究中就不討論計算時間的議題。

4.3 禁忌搜尋法的結果分析

為了詳細的分析 *TS_S*$_{new}$ 比 *TS_S*$_{old}$ 好的原因，本研究首先針對某一個情境下的某個實例進行分析，仔細觀察禁忌搜尋法的進化過程。發現在其進化過程最後會掉入一個迴圈(loop)的特性之中，而無法跳脫。此迴圈的特性導致當此演算法進化掉入迴圈之中時，其最佳解h^+無法再進步。這樣的發現讓我們想要驗證這樣的迴圈特性是否都存在這兩種演算法之中，結果發現 *TS_S*$_{old}$ 比 *TS_S*$_{new}$ 有較高的機率陷入迴圈之中。在接下來的小節中，本研究首先將禁忌搜尋法流程做簡單的整理；之後設計一個驗證方法來驗證此迴圈是否存在此兩個演算法中，並討論其結果；最後分析為何 *TS_S*$_{old}$ 容易陷入迴圈的原因。

4.3.1 禁忌搜尋法的進化流程整理

為了便於解釋迴圈的特性，我們首先整理在第三章提到的禁忌搜尋法的流程。此流程是一個解進化的過程，因為其解會受前面累計的經驗而產生，而其進化方式是採用階段式的進化方法；所謂階段式進化就是解的進化是一個步驟接著一步驟，在本研究中我們稱之為「**進化步驟**」。對第 i 步進化步驟，為了方便本研究之後的解釋，我們命名其輸入參數為 $S_i = \left(h_i^*, \{T_i^1, \dots, T_i^k\}\right)$ 而其產出為 $S_{i+1} = \left(h_{i+1}^*, \{T_{i+1}^1, \dots, T_{i+1}^k\}\right)$。其中，$h_i^*$ 是用來產生一組新鄰近解 $N(h_i^*)$ 的參考解；T_i^q 是第 q 個區段禁忌列表而 $\{T_i^1, \dots, T_i^k\}$ 是此多區段染色體的禁忌列表集合。

而接下來本研究將描述針對每一個進化步驟 i，輸入與輸出是如何運作。首先，此流程會以 h_i^* 做為參考解產生 $N(h_i^*)$；接著從 $N(h_i^*)$ 選取合適的染色體做為下一個參考解 h_{i+1}^*，同時考慮是否需要更新最佳解。由於 h_{i+1}^* 執行交換 h_i^* 的某一個區段中的基因產生的，所以在獲得下一個參考解的時候也會同時更新相關的禁忌列表，因此可以獲得新的禁忌列表組合 $\{T_{i+1}^1, \dots, T_{i+1}^k\}$。

本研究將禁忌搜尋法流程的每一個步驟都記錄下來，產生了一連串的紀錄；並定義從進化步驟 i 到 m 的所有輸入狀態集合為 $Track(i, i+m) = \{S_i, S_{i+1}, \dots, S_{i+m}\}$。根據這樣的定義，也就是說整個進化的流程可以表示成 $Track(1, N_f) = \{S_1, S_2, \dots, S_{N_f}\}$，$N_f$ 表示整個流程達到終止條件之前產生了多少組的鄰近解。因此本研究可以根據圖 4.1 的概念來定義何謂迴圈特性：如果 $S_i = S_{i+n}$ 且 $S_i \neq S_k$ 對所有的 $1 < k < n$，則我們稱 $Track(i, i+n)$ 為一個迴圈；而 n 為迴圈的大小。

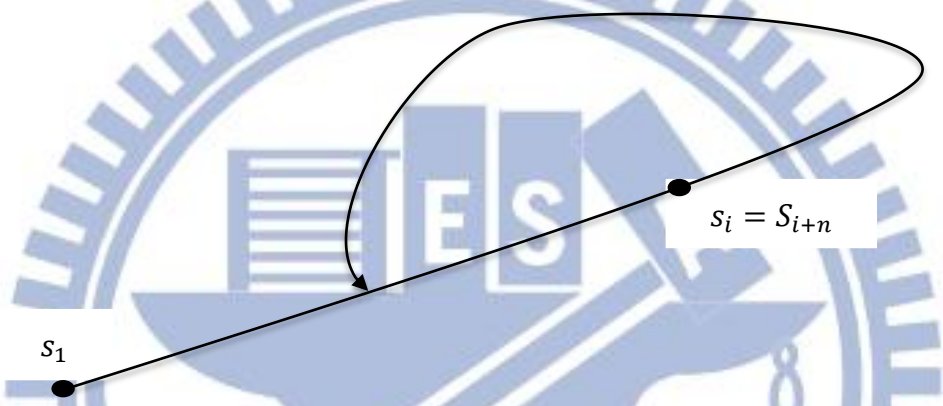


圖 4.1 迴圈特性的示意圖

4.3.2 迴圈特性的實驗

為了實驗在 *TS_S*$_{new}$ 與 *TS_S*$_{old}$ 之中是否有迴圈特性的存在，本研究發展了一個檢驗的流程，命名為 *Loop_Check*。而其流程將描述如下，而其中的 ***Tabu_Search*** 將參閱我們想要驗證的是 *TS_S*$_{new}$ 還是 *TS_S*$_{old}$ 來用 *TS_S*$_{new}$ 或 *TS_S*$_{old}$ 做鑲入取代。

Procedure Loop_Check

Step 1: Carry out the ***Tabu_Search*** process for determining N_f, S_{N_f}

- While the ***Tabu_Search*** process terminates (i.e., $h^+_{N_f}$ is obtained)
  - Record N_f, S_{N_f};
  - Set $k = 0, i_1^* = N_f, i_2^* = N_f$;

Step 2: Repeat the ***Tabu_Search*** process for checking loop existence

For each evolution-step $1 \le i \le N_f$

If $(S_i = S_{N_f})$, then /*while a loop seems appear*/

Set $k = k + 1$, and $i_k^* = i$; /*record the evolution-step*/

Endfor

$i_{Loop}^* = i_1^*$;

if $(i_1^* = N_f)$ then $\kappa = 0$; /*no loop is found*/

if $(i_1^* < N_f)$ then $\kappa = 1$; /*a loop is found*/

Compute *Loop_Size* = $i_2^* - i_1^*$;

Output κ, i_1^*, *Loop_Size*; Stop.

在上述的流程中，如果在 ***Tabu_Search*** 有迴圈$(i_1^* < N_f)$的存在，則$\kappa = 1$。如果有迴圈在$Track(1, N_f)$中被發現，則代表之後的進化過程都會陷入此迴圈中而無法在進化。因此，當 ***Tabu_Search*** 在進化步驟i_1^*之後的進化都會陷入迴圈中，而迴圈的大小為 *Loop_Size*。這也就表示說，此禁忌搜尋法的有效搜尋路徑最長為$Track(1, i_1^* + Loop_Size)$；而之後的搜尋進化都會重複進化步驟$i_1^* + Loop_Size$之中，造成此禁忌搜尋法已無法找到比$h_{i_1^*+Loop_Size}^+$更好的答案了。

因為有迴圈的特性，讓本研究想了解禁忌搜尋法的搜尋的有效性，因此我們訂定一個新的定義：**搜尋有效率**—其計算方式為 $\eta = (i_1^* + Loop\_Size)/N_f$。如果$\eta$值越大，表示搜尋的有效性會好，對整個流程會有較好的效果。此外，如果我們沒有找到迴圈的存在時$(i_1^* = N_f)$，$\eta = 1$且 *Loop_Size* = 0。相反地，如果有迴圈特性的存在時$(i_1^* < N_f)$，大部分的結果會是 *Loop_Size* > 0 and $0 < \eta < 1$。但是會有少數的結果會是$\eta = 1$且 *Loop_Size* > 0 因為其$i_2^* = N_f$。

這裡的實驗情境跟 4.2 節所介紹的實驗資料一樣，有 30 個情境，每個情境有 30 個實例，每個實例要執行 15 次。也就是說每個情境要跑 450 次，而每次會產生一個κ值和η值；本研究將這 450 次做平均產生$\bar{\kappa}$值和$\bar{\eta}$值來代表每一個情境

的數據。而且$\bar{\kappa}$值和$\bar{\eta}$值需要介於 0 跟 1 之間。其中，$\bar{\kappa}$表示迴圈特性的參數，如果$\bar{\kappa}$越大表示在此情境中相對應的演算法越容易產生迴圈，造成搜尋進化的有效性越低。而$\bar{\eta}$表示平均的搜尋有效率，如果$\bar{\eta}$越大表示在此情境中相對應的演算法有越長的搜尋有效性。

表 4.2 呈現了兩種演算法 *TS_S*$_{new}$ 與 *TS_S*$_{old}$ 的$\bar{\kappa}$值和$\bar{\eta}$值結果。以$\bar{\kappa}$值得結果來說，TS_S_{old}有較高的$\bar{\kappa}$值(總平均 85.46%)；因此大約有 85%的 *TS_S*$_{old}$ 實驗都陷入迴圈之中。相反地，*TS_S*$_{new}$ 的$\bar{\kappa}$值相當的低(總平均 0.38%)；表示 *TS_S*$_{new}$ 不容易陷入迴圈之中。以$\bar{\eta}$值結果來說，*TS_S*$_{old}$ 的$\bar{\kappa}$值相當低，總平均只有 24.93%表示有 75.07%都是無效的搜尋步驟。而 *TS_S*$_{new}$ 的$\bar{\eta}$值卻相對地高，其總平均達到 99.75%，表示 *TS_S*$_{new}$ 的有效搜尋比例很高。

經過我們的分析，發現 *TS_S*$_{new}$ 比 *TS_S*$_{old}$ 績效好是因為其較沒有迴圈的特性，而評估的方法就是計算出$\bar{\kappa}$值和$\bar{\eta}$值。而 *TS_S*$_{old}$ 的$\bar{\kappa}$值也表示禁忌搜尋法搭配舊表達法的搜尋過程有很高的機率會陷入迴圈之中；而其$\bar{\eta}$值顯示了它的搜尋有效性很差，也就是說其有效的進化步驟很短。而在下一小節，我們將更詳細的分析為何 *TS_S*$_{new}$ 與 *TS_S*$_{old}$ 會有不同的迴圈特性。

4.3.3 *TS_S*$_{new}$ 與 *TS_S*$_{old}$ 的迴圈特性

在這個小節將解釋為何 *TS_S*$_{old}$ 會有較高的機率迴圈之中。如前所述，*TS_S*$_{new}$ 和 *TS_S*$_{old}$ 都是採用階段式的進化方法；而對第 i 步進化步驟，我們命名其輸入參數為$S_i = \left(h_i^*, \{T_i^1, \dots, T_i^k\}\right)$而產出為$S_{i+1} = \left(h_{i+1}^*, \{T_{i+1}^1, \dots, T_{i+1}^k\}\right)$。其中，$h_i^*$是用來產生一組新鄰近解$N(h_i^*)$的參考解；$T_i^q$是第 q 個區段禁忌列表而$\{T_i^1, \dots, T_i^k\}$是此 k 個區段染色體的禁忌列表集合。為了方便解釋，本研究定義所有S_i的組合空間為$\Psi(S)$，其表示S_i內元素組合的所有可能。

表 4.2 迴圈特性實驗的結果

Scenario	$\bar{\kappa}$ (%)		$\bar{\eta}$ (%)	
	S_{new}	S_{old}	S_{new}	S_{old}
SSU0	4.14	86.00	96.40	24.92
SSU1	1.61	80.92	99.46	30.11
SSU2	0.00	85.06	100.00	25.05
SSU3	0.00	81.38	100.00	28.78
SSU4	0.00	87.59	100.00	23.92
SSU5	0.00	82.99	100.00	27.79
SSU6	0.00	72.87	100.00	36.93
SSU7	0.00	71.03	100.00	41.50
SSU8	0.00	83.45	100.00	26.96
SSU9	0.00	78.39	100.00	32.61
MSU0	0.46	87.11	99.94	21.36
MSU1	0.00	81.84	99.65	26.19
MSU2	0.00	93.10	100.00	18.67
MSU3	0.00	87.13	100.00	22.50
MSU4	0.00	85.98	100.00	25.33
MSU5	0.00	94.94	100.00	15.41
MSU6	0.00	81.61	100.00	28.76
MSU7	0.00	84.37	100.00	27.51
MSU8	0.00	79.31	100.00	29.58
MSU9	0.00	86.67	100.00	24.98
LSU0	2.99	88.05	98.78	21.41
LSU1	0.22	91.49	99.80	18.57
LSU2	2.07	91.26	98.53	17.52
LSU3	0.00	88.51	100.00	21.17
LSU4	0.00	84.83	100.00	26.18
LSU5	0.00	90.80	100.00	20.26
LSU6	0.00	94.02	100.00	14.18
LSU7	0.00	91.95	100.00	19.53
LSU8	0.00	82.99	100.00	27.04
LSU9	0.00	88.05	100.00	23.02
Average	**0.38**	**85.46**	**99.75**	**24.93**

如果演算法陷入迴圈時，表示$S_i = S_j$，其中$i \neq j$。這也隱含如果演算法的S_i組合空間($\Psi(S)$)越大，則越不容易達成$S_i = S_j$，也就是說陷入迴圈的機率越低。所以本研究接下來需要探討 *TS_S*$_{new}$ 和 *TS_S*$_{old}$ 的$\Psi(S)$是否有差別；為了方解釋，我們定義 *TS_S*$_{new}$ 的S_i組合空間為$\Psi_{new}(S)$，而 *TS_S*$_{old}$ 的為$\Psi_{old}(S)$。

以圖 2.2 的內容為案例來分析$\Psi_{old}(S)$與$\Psi_{new}(S)$的複雜度。以$\Psi_{old}(S)$來說，其染色體有 4 個區段(s_1-s_2-s_3-s_4)；區段 s_3 包含四個基因元素，而其所對應的禁忌列表大小為 2 (q_{size} = $\lceil C_2^4 * 0.3 \rceil$= $\lceil 1.8 \rceil$= 2)。其中，此區段可能的所有基因組合為 $4! = 24$；而可能的禁忌列表元素組合數量為 37 個，其計算方式解釋如下：在區段 s_3 中可能的移步有 6 個(C_2^4 = 6)，而其所對應的禁忌列表的儲存容量(空格數量)為 2 個(q_{size} = 2)。禁忌列表的空格可以填入移步或空缺，因此禁忌列表的所有可能元素組合數量為(6×5) + (6×1) + (1×1) = $P_2^6 + P_1^6 + P_0^6$，其中第一個區塊表示兩個空格都填滿了，第二個表示只填一個空格，最後則是都沒有填入任何移步。

根據上述的釋例，我們發展出一個通式。給定一個有 n_i 個基因的區段 s_i 且其對應的禁忌列表儲存容量為 q_i，我們可以計算其組合空間大小如下。此區段的可能基因組合為$n_i!$，可能的移步為$C_2^{n_i}$，而其對應的禁忌列表空格數量為 q_i。則我們計算禁忌列表的所有可能組合空間總數量如導出的下列公式：$P_{q_i}^{C_2^{n_i}} + P_{q_i-1}^{C_2^{n_i}} + \cdots + P_0^{C_2^{n_i}} = \sum_{j=0}^{q_i} P_j^{C_2^{n_i}}$；最後我們可以算出此區段的所有組合空間數量為 $n_i! \times \sum_{j=0}^{q_i} P_j^{C_2^{n_i}}$

考慮一個有 k 個區段的染色體，我們可以根據上述的推算方式獲得此染色體的組合空間$\Psi(S) = \prod_{i=1}^{k} \left(n_i! \times \sum_{j=0}^{q_i} P_j^{C_2^{n_i}} \right)$。根據此公式，本研究可以計算圖 3.2 中排程問題的$\Psi_{old}(S)$總數量為5.6×10^6而$\Psi_{new}(S)$的總數量為5.4×10^{28}。由此結果可以發現$\Psi_{old}(S)$的組合空間遠小於$\Psi_{new}(S)$的組合空間，也造成 *TS_S*$_{old}$ 有較高的機率會陷入迴圈之中。

由上述的分析結果，可以顯示 TS_S_{new} 比 TS_S_{old} 績效好可能是因為其有較高的自由度(組合空間大)，這也表示說如果提高其自由度(使$\Psi(S)$變大)可能會使其解的品質更好。因此，我們利用增加禁忌列表的儲存容量來驗證這樣的假說。本研究將禁忌列表的儲存容量($q_{size} = C_2^{n_i} \times p$)經由設定不同的 p 值來做實驗，因此我們分別用 p = 0.3, 0.5, 0.7, 0.9 來做相同的實驗情境，其結果如表 4.3。由表中可得到兩個重要的發現，首先 TS_S_{new} 在任何的 p 值之下都比 TS_S_{old} 績效來的好。另一個發現是當我們增加 p 值時，TS_S_{new} 與 TS_S_{old} 都有更好的績效表現。由此兩個發現可以驗證我們的假說：組合空間($\Psi(S)$)有較高的自由度會使得演算法得到更好的解品質。

表 4.3 不同 p 值的實驗結果

Average	*Makespan*							Computation Time	
***p* value**	***N***	***Nw+Ne***	***Ne***	***Nw***	***Cn***	***Ct***	***γ (%)***	***Tn***	***Tt***
***p* = 0.3**	**30**	**30.00**	**0.70**	**29.30**	**400.07**	**412.17**	**2.76**	**26.65**	**2.14**
***p* = 0.5**	**30**	**30.00**	**1.17**	**28.83**	**400.00**	**411.51**	**2.60**	**26.91**	**2.18**
***p* = 0.7**	**30**	**29.97**	**1.37**	**28.60**	**399.86**	**410.49**	**2.37**	**27.77**	**2.26**
***p* = 0.9**	**30**	**29.97**	**1.90**	**28.07**	**398.37**	**408.98**	**2.27**	**28.82**	**2.27**

第五章 基因演算法與結果分析

本章節將介紹另一個進化演算法—基因演算法；討論新表達法搭配基因演算法時，解的品質有怎樣的影響。因此本章節安排如下，首先介紹基因演算法的流程以；第二部分介紹實驗情境與結果；接下來第三部分討論 GA_S_{new} 比較好的原因分析。

5.1 基因演算法的流程

本章節可以分為兩小節；第一節描述基因演算法的搭配 S_{old} 的流程，命名為 GA_S_{old}；而基因演算法搭配 S_{new} 命名為 GA_S_{new}，此種演算法流程基本上與 GA_S_{old} 相同，所以 GA_S_{new} 可以視為是 GA_S_{old} 的特例；因此，第二節介紹 GA_S_{new} 的流程。

5.1.1 GA_S_{old} 的進化流程

GA_S_{old} 的進化流程是根據 Lin *et al.* (2009a)所發展的基因演算法而來，本研究將其分成兩個部份來做描述；首先是先描述其主要的三種運算模組，接下來是介紹其主要的流程。

此三種運算模組主要是用來從母代染色體產生新的子代染色體。而為了挑選到良好的母代染色體(parent)，因此其還使用二擇一的競賽法則(binary tournament selection; Brindle 1981)來挑選母代，其方法是先隨機從母代染色體群中挑選兩條染色體，然後選擇其績效較佳的一條作為進化用的母代染色體。

此三個運算模組分別屬於基因演算法中的交配與突變兩種類別。首先為 Syswerda (1989)所提出的 **PBX**(position-based crossover)交配方法，屬於二對一的

運算模組，也就是用兩條母代染色體只會產生一條子代染色體(offspring)。而其他兩種運算模組分別是**交換**(swap)與**插入**(insert)兩種運算模組屬於一對一的運算模組；也就是一個母代染色體會產生單一個子代染色體。而這兩種運算模組都是屬於突變方法。

如圖 5.1，本研究將以實際例子來解釋 PBX 方法。圖中有兩個母代染色體區段(Parent 1 和 Parent 2)，而每個母代染色體區段中各有 10 個工件，而每個工件各表示一個基因。而 PBX 的三個步驟解釋如下：首先，對 Parent 1 的每個基因隨機產生相對應的二元位數(0 或 1)，而這一系列的二元位數稱為遮罩(mask)。接下來依序將此遮罩中數值為「1」的相對應基因(工件)填入相對應的子代染色體位置中。最後，將子代還沒填入的基因由 Parent 2 填入，其填入順序依照其在 Parent 2 由左到右的順序依序填入，就可以得到完整的子代染色體。

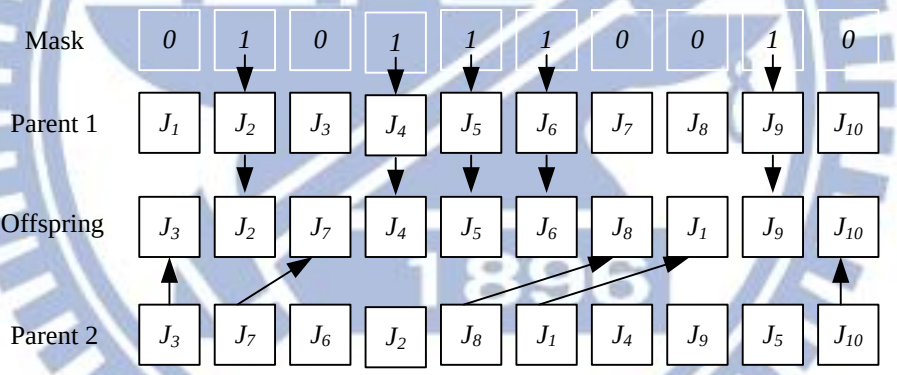


圖 5.1 PBX 交配運算模組【資料來源：戴邦豪(2010)】

而**交換**的運算模組實例如圖 5.2，針對某個染色體區段，隨機選取兩個基因(例如：J_4 和 J_7)來進行交換的步驟。而**插入**的運算模組是採用兩步驟的方法，如圖 5.3。首先隨機選兩個基因(例如：J_4 和 J_7)；然後將排序較後面的基因(J_7)插入到排序比較前面的基因(J_4)之前。

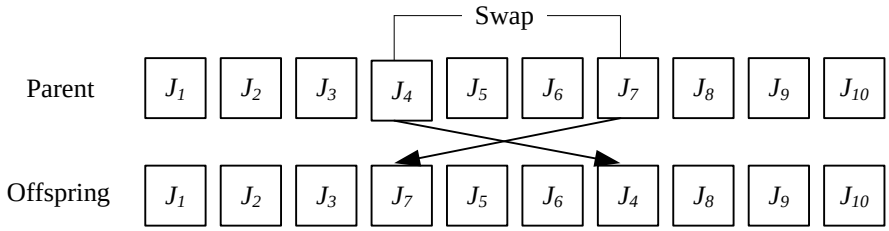


圖 5.2 交換運算模組【資料來源：戴邦豪(2010)】

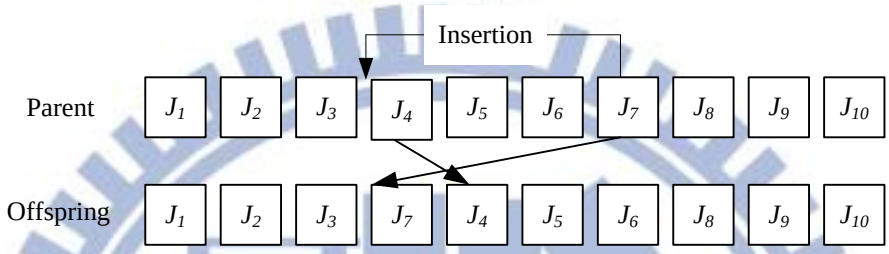


圖 5.3 插入運算模組【資料來源：戴邦豪(2010)】

描述完三個重要的運算模組之後，本研究將介紹 GA_S_{old} 的流程。如之前所描述，一個舊染色體表達法(S_{old})包含了 k+1 個區段；而為了產生一個子代染色體，這 k+1 個區段是各自應用這三個模組來產生新的子代區段。然後再將這 k+1 個新的子代區段組合起來成為一個完整的子代染色體。而整個 GA_S_{old} 的流程如下：

Procedure *GA_S*$_{old}$

Step 1: *Initial Setting*

- Input parameters T_f, P_{size}, $0 \le p_c \le 1$, $0 \le p_m \le 1$.
- Set $t = 0$; (t denotes the age of population P_t)
- Form a population P_0 by randomly generating P_{size} chromosomes.

Step 2: *Record the Best Solution in* P_t

- Find $\omega_{best}(t)$ (i.e., the best solution in P_t)

- Denote $L_{best}(t)$ as the makespan of $\omega_{best}(t)$.

Step 3: *Apply Crossover Operation to Update P_t*

- *Skip Check*: Generate a random number r, if $r > p_c$ then skip the crossover operation and go to Step 4;
- *Select Parents:* Carry out the binary tournament section method twice to pick two chromosomes from P_t as a pair of parents, which are respectively called X_{p1} and X_{p2} (X_{p2} denotes the inferior one in terms of fitness values). Each segment i $(1 \le i \le k+1)$ in X_{p1} and X_{p2} is respectively called $S_{i,p1}$ and $S_{i,p2}$.
- *Create New Chromosome*
  - Create offspring for each segment *i*

    For segment $i = 1, \ldots, k+1$

    Apply the position-based crossover operator on $S_{i,p1}$ and $S_{i,p2}$ to create one new segment S_i.

    Endfor
  - Combine $S_i\,(1 \le i \le k+1)$ to form a new chromosome $X_{offspring}$
- *Update Population P_t:*
  - If X_{p2} is inferior to $X_{offspring}$, then X_{p2} is removed from P_t and $X_{offspring}$ is placed into P_t; else go to Step 4

Step 4: *Apply Mutation Operation to Update* P_t

- *Skip Check*: Generate a random number r, if $r > p_m$ then skip the mutation operation and go to Step 5;
- *Select Parent*: Use the binary tournament selection method to pick one parent, denoted by X_{parent} and each of its segment is denoted by $S_{i,p}$.
- *Select Operator:* Of the swap and insertion operators, randomly determine which one to use—each one is with the same probability (0.5).
- *Create New Chromosome*
  - Create offspring for each segment *i*

    For segment $i = 1, \ldots, k+1$

    Apply the swap/insertion operator on $S_{i,p}$ to create one new segment S_i.

    Endfor
  - Combine $S_i\,(1 \le i \le k+1)$ to form a new chromosome $X_{offspring}$
- *Update Population* P_t*:* If X_{parent} is inferior to $X_{offspring}$, then X_{parent} is removed from P_t and $X_{offspring}$ is placed into P_t; else go to Step 4. .

Step 5: Termination Check

- If $L_{best}(t) = L_{best}(t+1) = \cdots = L_{best}(t+T_f)$, Then STOP;

Else, go to Step 2.

5.1.2 GA_S_{new} 的進化流程

而 GA_S_{new} 的進化流程其實是 GA_S_{old} 進化流程的一個特例。因為在 GA_S_{new} 中染色體表達法為一個區段的表達的方式，而表達法在 GA_S_{old} 中為 k+1 個區段；因此只要將流程中的多個區段改為單一區段就是 GA_S_{new} 的流程了。

5.2 基因演算法的實驗情境與結果

本研究的目標值是最大完工時間(makespan)，比較 GA_S_{old} 和 GA_S_{new} 的績效表現。基因演算法的相關參數設定如下：P_{size} = 1,000、p_c = 0.95、p_m = 0.10 和 T_f = 4,000,000。本研究使用 C++語言撰寫，而實驗的電腦配備為 AMD Athlon(tm) Ⅱ *4640 3.0Ghz CPU 以及 4G Ram。

實驗資料的來源是跟據 Schaller *et al.* (2000)的文獻，分成 30 個情境，而每個情境包含 30 個實例。在這 30 個情境中，我們以 *X-F-m* 來表示其所代表的不同情境。*X* 表示不同的整備時間(LSU,MSU,SSU)，*F* 表示工件族的數目，*m* 表示機台的數目—有多少加工階段。SSU 表示整備時間最短，MSU 表示整備時間為中間值而 LSU 表示整備時間最大。

在每個情境的 30 個實例中，我們必須隨機產生以下的相關參數：每個工件族中的工件數目(n_f)、生產時間與每個工件族之間的整備時間。其中，n_f 是採用間斷的單一分配(uniform)函數產生 U[1,10]；而工件在每一個加工階段的生產時間也是採用間斷的單一分配函數產生 U[1,10]。而整備時間則根據不同的整備時間大小採用不同的方案，在 SSU 的時候是採用 U[1,20]、在 MSU 的時候是 U[1,50]，而大整備時間 LSU 則是 U[1,100]。為了避免實驗受到亂數的影響，因此每一個實例都需要用 15 個不同的亂數種子(seed)來進行實驗。將此 15 個結果取平均做為此實例的績效，然後再將每個情境的 30 個實例平均起來做為最後評估的績效值。

表 5.1 顯示了實驗的結果。而其參數解釋如下：*GA_S*$_{old}$ 所獲的的績效為C_t，*GA_S*$_{new}$ 所獲的的績效為C_n；而其對應的計算時間分別為T_t和T_n。而為了比較此兩種演算法的績效，此研究設定一個參數$\gamma = (C_t - C_n)/C_t$。此研究還比較每個情境之中，新舊表達法在不同實例的績效表現。$N = 30$表示每個情境有 30 個實例，而N_e表示$\gamma = 0$的實例個數，而N_w代表$\gamma > 0$的實例個數。$N_w + N_e$為 *GA_S*$_{new}$ 與 *GA_S*$_{old}$ 比較勝過或平手的總數量。如果$N_w + N_e$越大表示 *GA_S*$_{new}$ 有明顯的贏過 *GA_S*$_{old}$。

在 30 個情境之下，γ最好為 0.17%最差則為-0.08%，平均為 0.03%，表示 *GA_S*$_{new}$ 有比 *GA_S*$_{old}$ 績效好。而且在 30 個情境下的 30 個實例的總平均$N_w + N_e = 26.27$，表示 *GA_S*$_{new}$ 有比 *GA_S*$_{old}$ 差的實例個數偏低。本研究也將此實驗做統計分析，其 *t* 值為$t_0 = 3.16 > t_{0.025,899} = 1.96$，這表示 *GA_S*$_{new}$ 在 95% 的信心水準下明顯的比 *GA_S*$_{old}$ 績效好。而其運算時間都非常短，因此本研究不討論運算時間。

表 5.1 基因演算法的實驗結果

Scenario	*Makespan*							Computation Time	
	N	***Nw+Ne***	***Ne***	***Nw***	***Cn***	***Ct***	***γ (%)***	***Tnew***	***Told***
SSU33	30	30	30	0	134.47	134.47	0.00	14.88	18.54
SSU34	30	28	25	3	150.98	150.99	0.01	16.84	20.50
SSU44	30	28	24	4	185.64	185.64	0.00	21.17	24.56
SSU55	30	29	19	10	241.94	242.11	0.07	30.63	32.30
SSU56	30	28	12	16	253.36	253.71	0.14	31.95	34.81
SSU65	30	28	15	13	285.78	285.96	0.06	36.41	37.27
SSU66	30	28	13	15	294.90	295.14	0.08	40.16	40.03
SSU88	30	22	7	15	402.60	403.14	0.12	67.49	58.19
SSU108	30	16	0	16	481.76	481.94	0.04	95.61	72.24
SSU1010	30	23	1	22	521.90	522.45	0.10	107.94	83.84
MSU33	30	30	30	0	160.00	160.00	0.00	13.75	17.91
MSU34	30	30	29	1	184.68	184.69	0.01	15.70	19.59
MSU44	30	30	28	2	237.60	237.61	0.00	21.01	24.23
MSU55	30	29	24	5	306.09	306.16	0.02	29.64	31.42
MSU56	30	30	24	6	317.47	317.68	0.06	30.04	32.98
MSU65	30	28	23	5	362.63	362.66	0.01	35.92	36.68
MSU66	30	28	19	9	380.02	380.07	0.02	38.55	39.33
MSU88	30	21	7	14	510.39	510.44	0.01	65.15	56.55
MSU108	30	14	2	12	623.95	623.51	(0.08)	89.16	70.74
MSU1010	30	21	1	20	655.17	655.52	0.05	99.95	79.66
LSU33	30	30	30	0	228.03	228.03	0.00	15.44	18.79
LSU34	30	30	30	0	239.00	239.00	0.00	14.94	18.96
LSU44	30	29	29	0	323.44	323.43	(0.00)	19.99	23.81
LSU55	30	29	28	1	415.97	415.97	(0.00)	28.00	30.95
LSU56	30	30	24	6	436.97	437.17	0.04	30.72	33.24
LSU65	30	30	27	3	491.92	492.02	0.02	35.95	36.48
LSU66	30	28	24	4	514.32	514.34	0.00	36.37	38.44
LSU88	30	23	10	13	701.63	701.11	(0.08)	60.78	54.90
LSU108	30	16	3	13	847.62	847.36	(0.04)	85.85	68.94
LSU1010	30	22	1	21	907.13	908.67	0.17	101.14	78.04
Average	**30**	**26.27**	**17.97**	**8.30**	**393.25**	**393.37**	**0.03**	**44.37**	**41.13**

5.3 基因演算法的結果分析

在本章節中我們將嘗試解釋為何 GA_S_{new} 比 GA_S_{old} 好的原因。經過詳細的分析，本研究發現了兩個主要的推論(Chen *et al.* 2012)。第一個推論是「**主導區段特性**」，在傳統表達法中，第一個區段(家族間的排序)就是一個主導區段。亦即，第一個區段是工件族間排序比其他工件族內工件排序區段對解品質更具有影響。在之前的 5.1.1 章節中，可以發現染色體群體會持續地被較好解品質的染色體所取代。這樣的取代機制會使主導區段的染色體逐漸趨向「**同質化特性**」。

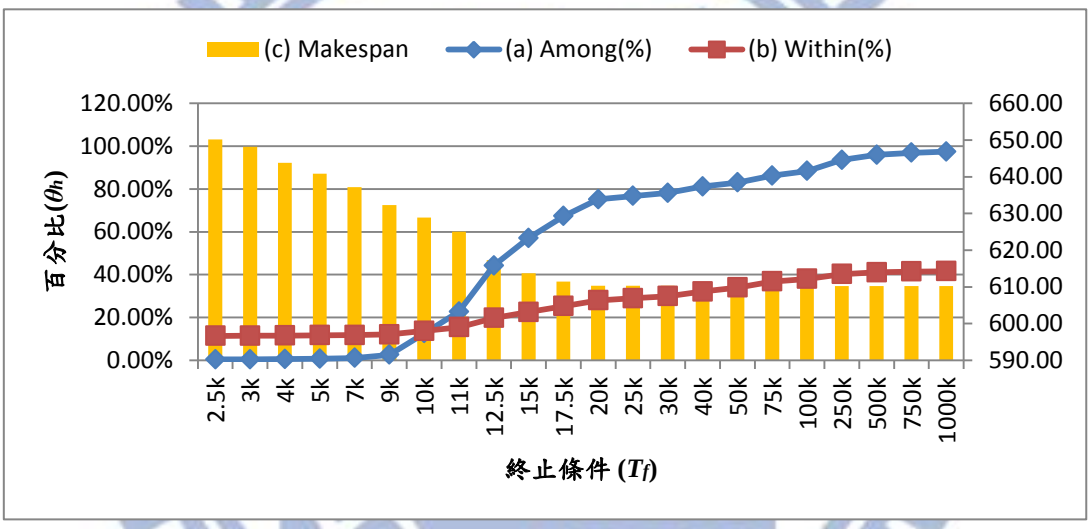


圖 5.4 GA_S_{old} 在 MSU9 不同終止條件(T_f)下的結果: (a) 在主導區段的同質化百分比(θ_h)接近 100%, (b) 在其他區段的同質化百分比(θ_h)平均接近 40%, (c) 垂直軸為目標值(最大完工時間)

如圖 5.4 所示，本研究以 MSU9 中的一個實例為解釋案例。在圖 5.4 中，水平軸 T_f 表示演算法的終止條件；右邊的垂直軸表示解品質而左邊的垂直軸表示母體染色群中統治區段(排序完全相同)相同的比例；在本研究中命名其百分比為 θ_h。在此圖中，當終止條件增加時，在一開始解品質會進步，但之後會趨於固定。除此之外，當終止條件增加時，θ_h 會逐漸地增加最後幾乎快達到 100%；這也表示主導區段不容易繼續進化。表 5.2 中可以發現所有情境的主導區段同質化百分

比平均為 98.83%，而其他不重要的區段其同質化百分比就很低，其平均為 29.32%；亦即表示主導區段對解品質有很大的影響性。

表 5.2 GA_S_{old} 的同質化結果

Scenario	主導區段	其他區段
LSU0	98.58%	17.98%
LSU1	99.86%	27.46%
LSU2	96.89%	29.56%
LSU3	99.63%	24.85%
LSU4	98.62%	25.31%
LSU5	99.48%	23.32%
LSU6	99.50%	24.56%
LSU7	99.06%	32.24%
LSU8	98.61%	30.42%
LSU9	98.63%	32.31%
MSU0	99.93%	29.26%
MSU1	97.67%	27.59%
MSU2	99.22%	26.41%
MSU3	98.39%	26.47%
MSU4	99.10%	32.32%
MSU5	98.40%	27.76%
MSU6	98.83%	29.99%
MSU7	98.83%	32.85%
MSU8	98.19%	29.83%
MSU9	98.31%	36.95%
SSU0	99.88%	21.16%
SSU1	98.33%	22.65%
SSU2	99.33%	27.00%
SSU3	99.13%	28.04%
SSU4	98.98%	36.25%
SSU5	99.30%	31.45%
SSU6	98.84%	32.51%
SSU7	98.57%	38.73%
SSU8	98.05%	34.22%
SSU9	98.75%	40.21%
Average	**98.83%**	**29.32%**

第二個推論是新表達法是重複的；亦即，一個解可以用不同的 S_{new} 來表示。這種「一對多」的特性有好處也有壞處。壞處是此種表達法會比較沒有效率，因為重複表達的關係。而其好處是染色體比較有彈性，不容易有同質化的現象。在圖 5.5 中，本研究比較 GA_S_{new} 比 GA_S_{old} 的績效表現。在終止條件較小的時候，GA_S_{new} 的解品質比 GA_S_{old} 差；但是當終止條件變大時，GA_S_{new} 的解品質就比 GA_S_{old} 好。這表示我們如果有足夠的計算時間，GA_S_{new} 會持續的進步，然而 GA_S_{old} 並不會進步。

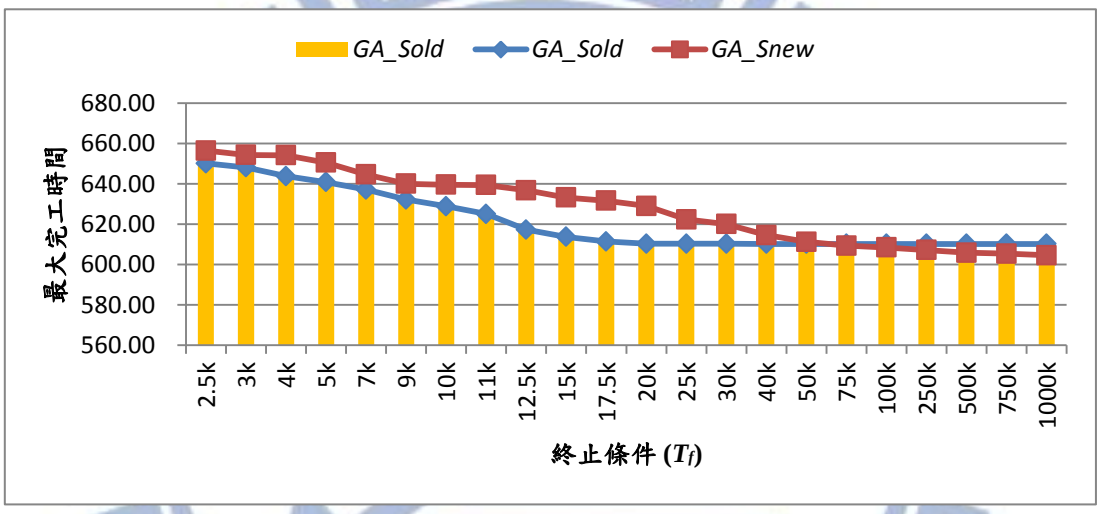


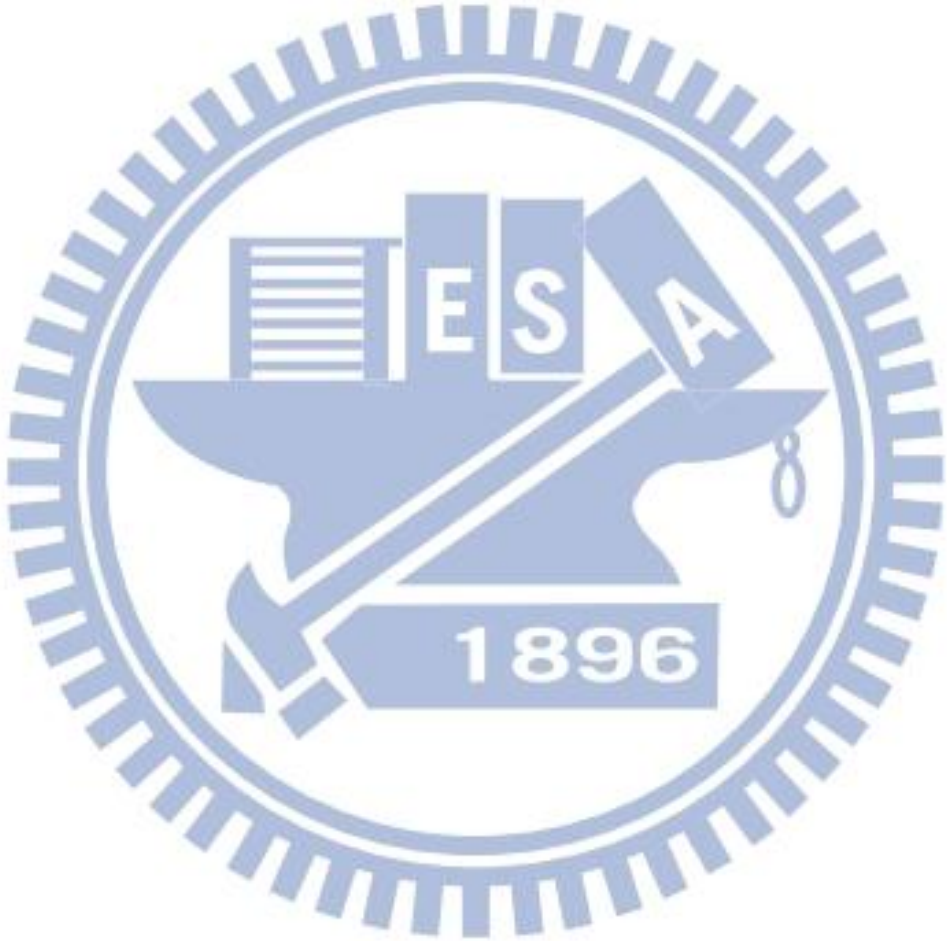
圖 5.5 MSU9 的問題中比較 GA_S_{old} 和 GA_S_{new} 在不同終止條件下的解品質

由上面的推論可以推導出如果先使用 GA_S_{old} 再用 GA_S_{new} 來求解，可以避免一開始因為重複而沒效率的搜尋更有效率，也可以避免舊染色體表達法無法進化的缺點；而此推論在戴邦豪(2010)的文章中可以獲得驗證。

在表 5.3 中，此三種方法 GA_S_{new}、$GA\_S_{new\text{-}old}$ 和 $GA\_S_{old\text{-}new}$ 都跟 GA_S_{old} 做比較，雖然其終止條件和本研究中的終止條件不同；亦即，在戴邦豪的論文中，其終止條件較長，但不影響實驗結果。而 $GA\_S_{new\text{-}old}$ 表示演算法先使用 S_{new} 之後再使用 S_{old}；相反地，$GA\_S_{old\text{-}new}$ 表示演算法先使用 S_{old} 之後再使用 S_{new}。其結果顯示 $GA\_S_{old\text{-}new}$ 有較好的表現，γ 為 0.03%比其他的 0.02%還要好；也驗證了本研究的推論。

表 5.3 GA_S_{new}、$GA\_S_{new\text{-}old}$ 和 $GA\_S_{old\text{-}new}$ 的實驗結果【資料來源：戴邦豪(2010)】

Methods	Average of all Scenarios						
	N_e	N_w+N_e	N_e	N_w	γ (%)	T_{new}	T_{old}
GA_S_{new}	30	25.18	16.12	9.06	0.02	119.43	145.47
$GA\_S_{new\text{-}old}$	30	25.47	17.27	8.20	0.02	117.99	128.85
$GA\_S_{old\text{-}new}$	30	29.83	18.23	11.60	0.03	119.81	128.85



第六章 蟻群最佳化演算法與結果分析

本章節分成三大部分：首先介紹蟻群最佳化演算法的演算流程；之後介紹實驗情境與結果；最後針對實驗的結果作分析。

6.1 蟻群最佳化演算法的流程

為了方便解釋蟻群最佳化演算法，本研究將蟻群最佳化演算法搭配 S_{old} 命名為 ACO_S_{new}；而搭配 S_{new} 命名為 ACO_S_{old}。而兩種搭配其演化流程基本上都相同。因此本研究將先介紹 ACO_S_{new}，然後再介紹 ACO_S_{old}。

6.1.1 ACO_S_{new} 的進化流程

考慮一個有 N 個工件的排程問題。S_{new} 是一個單一區段且代表整個工件的排序；而經過解碼機制，此排序可以得到兩個決策：(1) 工件在工件族中的排序，(2) 工件族之間的排序。一個好的染色體表示其結果是最小最大完工時間。而 ACO_S_{new} 得目的是得到一個好的染色體排序；而為了達成這樣的目的，本研究將此排程問題看成是一個旅行者問題(TSP)。給定一個虛擬起始節點(Node_0)，此演算流程必須經過 N 個節點。演算法流程得目的是找到一個好的旅行者途徑(也就是這 N 個節點的經過排序)，而此途徑也就是一個好的染色體(工件排序)。

為了解釋一個世代的旅行途徑，本研究假設有一個 N+1 個節點的網絡(包含了 Node_0)。在此網絡中，每兩個節點之間都有一個路徑，如此會產生總共$\mathrm{C}(N,2)$條路徑，如圖 6.1 所示。而每條路徑都有一個「**總和特性**」(也稱作**績效指標**)；其代表會選擇經過該條路徑的偏好程度。在本研究中，我們定義績效指標為λ_{ij}來表示從節點 i 到節點 j 路徑上的指標。如果λ_{ij}越大，表示選擇經過此條路徑的機率越高。而λ_{ij}包含了兩個重要的特性指標—τ_{ij}和η_{ij}；而這兩個指標將在之後做詳細地描述。

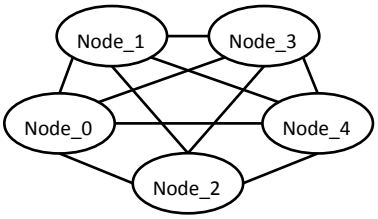


圖 6.1 蟻群最佳化演算法的例子，其所有的路徑總數為$\mathrm{C}(N,2)$

給定一個蟻群最佳化網絡，每一個世代產生一個旅程路徑的方式將介紹如下。假設一個銷售員目前在節點 i，且還有 m 個節點尚未經過；也就是說此銷售員有 m 條路徑(也就是 m 個節點)可以選擇。在 ACO_S_{new} 中採用輪盤法(Dorigo & Gambardella 2000)來選擇下一個節點；因此，選擇節點 k 的機率為$p_{ik} = \lambda_{ik}/(\sum_{j=1}^{j=m}\lambda_{ij})$。而每個銷售員都是從同一個虛擬起始節點開始進行旅程，不斷重複上述的方式來選擇下一個節點，最後可以獲得一個旅行途徑。如果有 S 位不同的銷售員進行旅程，我們經由輪盤法的機率性原因，使我們可能獲得 S 條不同的旅行途徑。而在過去文獻中，銷售員通常又稱為螞蟻，這也是為何此種演算法稱為蟻群最佳化演算法的原因。

在 ACO_S_{new} 網絡中的績效指標是會不斷地更新；本研究定義被更新的網絡狀態為 W_t，所以 W_0 為最初始的網絡狀態而 t 代表網絡被更新過的次數。此外，W_t 和 W_{t+k} 的網絡組成是相同的，但其每條路徑上的績效指標(λ_{ij})卻不相同。而 W_{t+1} 的產生是以每隻螞蟻在 W_t 所產生的回饋為依據來產生。亦即，本研究會先在 W_t 世代送出 S 隻螞蟻(銷售員)來進行旅程，然後產生 S 條旅行途徑；而每一條旅行途徑可能會有不同的績效(最大完工時間)。然後將這 S 條不同旅行途徑的績效綜合起來，用來改變λ_{ij}以獲得 W_{t+1}；而改變λ_{ij}的方式介紹如下。

如前所述，λ_{ij}包含了兩個重要的特性指標—τ_{ij}和η_{ij}。其中，η_{ij}是一個穩態的特性指標，其表示路徑從節點 i 到節點 j 的相對重要性，且在不同的 W_t 其數值都是相同的。在 ACO_S_{new} 中，本研究定義$\eta_{ij} = 1/(s_{ij} + p_j)$，其中$p_j$代表工件 j

的加工時間，s_{ij}為製造工件 i 換成工件 j 所需要的設置時間。也就是說，如果工件 i 和工件 j 是同一個工件族的時候，$s_{ij} = 0$。

相反地，τ_{ij}是一個動態特性指標，在不同的 W_t 其數值會不斷的變動。假設第 s 條旅行途徑的績效為 L_s；總和所有的 L_s 就可以更改τ_{ij}(在文獻中稱為費洛蒙濃度)。而其公式如下：

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{s=1}^{S} \Delta\tau_{ij}^{s},$$

where

$$\Delta\tau_{ij}^{s} = \begin{cases} \frac{Q}{L_s} & \text{when } s\text{-th traveler passes through path } ij \\ 0 & \text{otherwise} \end{cases}$$

Q: a constant (parameter, $Q > 0$)

ρ: evaporation rate (parameter, $1 > \rho > 0$)

在上述的公式中，$\tau_{ij}(t+1)$表示在 W_{t+1} 世代時，從節點 i 到節點 j 的路徑上的費洛蒙濃度。而路徑上的費洛蒙濃度從 W_t 世代到 W_{t+1} 世代時，必須要有一定的比例(ρ)揮發；因此，$(1-\rho)\tau_{ij}(t)$表示殘留在每條路徑上的費洛蒙濃度。而$\Delta\tau_{ij}^{s}$為第 s 隻螞蟻針對從節點 i 到節點 j 的路徑上的績效回饋。當第 s 隻螞蟻有經過節點 i 到節點 j 的路徑時，其對該路徑增加的費洛蒙濃度為$\Delta\tau_{ij}^{s} = Q/L_s$。亦即，$L_s$ 值越小(越少的最大完工時間)，$\Delta\tau_{ij}^{s}$就越大(殘留的費洛蒙越多)。相反地，如果第 s 隻螞蟻未經過節點 i 到節點 j 的路徑時，則不會殘留任何的費洛蒙濃度($\Delta\tau_{ij}^{s} = 0$)。

而上述的更新$\tau_{ij}(t)$是一個遞迴的模式，也就是會不斷的重複上述的步驟。因此，為了方便本研究做解釋，我們定義一開始在節點 i 到節點 j 的路徑上的費洛蒙濃度為$\tau_{ij}(0)$，而其計算的方式如下。首先在 *ACO_S*$_{new}$ 中，本研究依據最短加工時間(SPT)為準則，產生一條初始的工件排序。亦即，在排序 N 個工件時，時間越短的工件會優先排列。而此染色體我們稱為是初始染色體(ω_0)，而其所產生的最大完工時間為L_0。依據此目標值，我們可以產生節點 i 到節點 j 的路徑上的初始費洛蒙濃度$\tau_{ij}(0) = 1/(S \cdot L_0)$，其中 S 表示所有的螞蟻(銷售員)總和。

在求得兩個特性指標：τ_{ij}和η_{ij}之後，我可以根據下列公式算出績效指標λ_{ij}。

$$\lambda_{ij}(t) = \left[\tau_{ij}(t)\right]^{\alpha}[\eta_{ij}]^{\beta}, \text{ where } \alpha,\ \beta \text{ are positive integers.}$$

而在世代 W_t，λ_{ij}經由計算可以求得；而每位銷售員的旅行途徑具有機率性。如前所述，當第 s 隻螞蟻目前在節點 i 要選擇下一個節點 k 的機率公式如下：

$$p_{ik}^{s}(t) = \begin{cases} \dfrac{\lambda_{ik}(t)}{\sum_{j \in J_s} \lambda_{ij}(t)} & if\ node\ k \in J_s \\ 0 & if\ node\ k \notin J_s \end{cases}$$

其中，J_s表示網絡中第 s 隻螞蟻還未經過的所有節點集合。而由於輪盤法的機率性原因，不同的兩隻螞蟻 s 和 q，在大部分的情形下$J_s \neq J_q$；這也導致 $p_{ik}^{s}(t) \neq p_{ik}^{q}(t)$。而$p_{ik}^{s}$也稱為是第 s 隻螞蟻的旅行機率。

每當在蟻群最佳化演算法更新的過程中獲得新的 W_t，每當有最好的染色體出現時都會被記錄起來，本研究將其命名為$\omega_{best}(t)$，而其最大完工時間為 $L_{best}(t)$。而蟻群最佳化演算經過 T_f 個世代，最佳解還是一樣的話，就終止蟻群最佳化演算法的流程。亦即，$L_{best}(t_n) = L_{best}(t_n + 1) = \cdots = L_{best}\left(t_n + T_f\right)$。

而整個 *ACO_S*$_{new}$ 的組成都已經介紹如上述段落。為了方便讀者可以更了解整個演算法的流程，本研究將 *ACO_S*$_{new}$ 的流程整理如下：

Procedure ACO_S$_{new}$

Step 1: Initialization

- Input parameters $\rho, \alpha, \beta, S, T_f$;
- Set t = 0.

Step 2: Compute $\tau_{ij}(0)$

- Generate *initial chromosome* ω_0;
- Compute its makespan L_0; Set $\tau_{ij}(0) = \frac{1}{S \cdot L_0}$;
- Create ACO network W_0.

Step 3: Update ACO network W_t

- Send *S* ants to travel through network W_t;
- Obtain *S* traveling routes (chromosomes) and their makespans;
- Obtain W_{t+1} by using the ACO network updating method;
- Record the best chromosome $\omega_{best}(t)$ and its makespan $L_{best}(t)$.

Step 4: Termination Check

- If $L_{best}(t_n) = L_{best}(t_n + 1) = \cdots = L_{best}(t_n + T_f)$, STOP;
- Else, Go to Step 3.

6.1.2 ACO_S_{old} 的進化流程

如圖 3.2 所示，舊表達法有 *F+1* 個區段。第一個區段是工件族之間的排序，而其他 *F* 個區段為每個工件族中工件的排序。而在 ACO_S_{old} 中，其染色體是一個組合式的蟻群最佳化演算法網路，其包含 *F+1* 個子網絡。亦即，本研究將每個區段變成一個子網絡，因此將這 *F+1* 個子網絡連結起來就可以產生一個組合式的蟻群最佳化演算法網路。以圖 3.2 中的染色體為例，可以產生四個子網絡，如圖 6.2。每一個子網絡可以使用上一節 ACO_S_{new} 所介紹的方法來產生旅行途徑。亦即，每個子網絡都有一個初始虛擬節點；然後螞蟻從此節點出發經過每一個其他的節點來完成此子網絡的旅行途徑，而此途徑也就代表了此區段的節點排序。

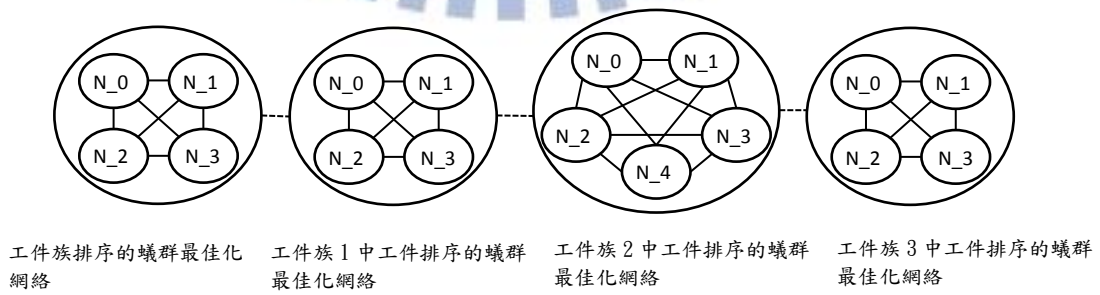


圖 6.2 組合式的蟻群最佳化演算法網絡

在 ACO_S_{old} 中，染色體是透過一隻螞蟻旅行整個組合式的蟻群最佳化演算法網路來獲得，因此每一隻螞蟻都必須依序的旅行完每個子網絡。每當螞蟻走完一個子網絡，它會自動的轉移到下一個子網絡的虛擬起始節點。透過螞蟻旅行完整個組合式的蟻群最佳化演算法網路，我們就可以獲得其染色體。

而如前所述，ACO_S_{old} 和 ACO_S_{new} 基本上是相同的進化流程，但是套用不同的解表達法。所以 Procedure ACO_S_{old} 和 Procedure ACO_S_{new} 幾乎是一樣的，除了 W_t 在 ACO_S_{old} 是一個組合式的蟻群最佳化演算法網路。在 ACO_S_{old} 中，我們也定義$\eta_{ij} = 1/(s_{ij} + p_j)$且針對每一個子網絡都套用最短加工時間的法則來計算$\tau_{ij}(0)$。比較特殊的是在第一個區段中，其加工時間是以每一個工件族的所有工件加工時間總和為基準。

6.2 蟻群最佳化演算法的實驗情境與結果

本研究的目標值是最大完工時間(makespan)，比較 ACO_S_{old} 和 ACO_S_{new} 的績效表現。而蟻群最佳化演算法的相關參數設定如下：$\rho = 0.8$和$S = \mathrm{C}(N, 2)$，其中 N 是所有工件的總數目；其他參數$\alpha = 1$、$\beta = 2$和 $T_f = 10{,}000$。本研究使用 C++語言撰寫，而實驗的電腦配備為 AMD Athlon(tm) Ⅱ*4640 3.0Ghz CPU 以及 4G Ram。

實驗資料的來源是跟據 Schaller *et al.* (2000)的文獻，分成 30 個情境，而每個情境包含 30 個實例。在這 30 個情境中，我們以 *X-F-m* 來表示其所代表的不同情境。*X* 表示不同的整備時間(LSU,MSU,SSU)，*F* 表示工件族的數目，*m* 表示機台的數目—有多少加工階段。SSU 表示整備時間最短，MSU 表示整備時間為中間值而 LSU 表示整備時間最大。

每個工件族中的工件數目(n_f)、生產時間與每個工件族之間的整備時間。其中，n_f是採用間斷的單一分配(uniform)函數產生 U[1,10]；而工件在每一個加工

階段的生產時間也是採用間斷的單一分配函數產生 U[1,10]。而整備時間則根據不同的整備時間大小採用不同的方案，在 SSU 的時候是採用 U[1,20]、在 MSU 的時候是 U[1,50]，而大整備時間 LSU 則是 U[1,100]。為了避免實驗受到亂數的影響，因此每一個實例都需要用 15 個不同的亂數種子(seed)來進行實驗。將此 15 個結果取平均做為此實例的績效，然後再將每個情境的 30 個實例平均起來做為最後評估的績效值。

表 6.1 顯示了實驗的結果。而其參數解釋如下：*ACO_S*$_{old}$ 所獲的的績效為C_t，*ACO_S*$_{new}$ 所獲的的績效為C_n；而其對應的計算時間分別為T_t和T_n。而為了比較此兩種演算法的績效，此研究設定一個參數$\gamma = (C_t - C_n)/C_t$。此研究還比較每個情境之中，新舊表達法在不同實例的績效表現。$N = 30$表示每個情境有 30 個實例，而N_e表示$\gamma = 0$的實例個數，而N_w代表$\gamma > 0$的實例個數。$N_w + N_e$為 *ACO_S*$_{new}$與 *ACO_S*$_{old}$比較勝過或平手的總數量。如果$N_w + N_e$越大表示 *ACO_S*$_{new}$ 有明顯的贏過 *ACO_S*$_{old}$。

在 30 個情境之下，γ最好為 3.08%最差則為 0.56%，平均為 1.70%，表示 *ACO_S*$_{new}$ 有比 *ACO_S*$_{old}$ 績效好。而且在 30 個情境下的 30 個實例的總平均 $N_w + N_e = 26.80$，表示 *ACO_S*$_{new}$ 有比 *ACO_S*$_{old}$ 差的實例個數偏低。本研究也將此實驗做統計分析，其 t 值為$t_0 = 30.88 > t_{0.025,899} = 1.96$，這表示 *ACO_S*$_{new}$ 在 95%的信心水準下明顯的比 *ACO_S*$_{old}$ 績效好。而其運算時間都在可接受範圍，因此本研究不討論運算時間。

表 6.1 蟻群最佳化演算法的實驗結果

Scenario	*Makespan*							Computation Time	
	N	***Nw+Ne***	***Ne***	***Nw***	***Cn***	***Ct***	***γ (%)***	***Tnew***	***Told***
SSU33	30	25	4	21	134.96	136.27	0.97	3.35	0.64
SSU34	30	28	1	27	152.56	154.46	1.26	5.15	1.28
SSU44	30	28	0	28	187.63	190.84	1.71	9.77	2.14
SSU55	30	24	0	24	246.79	250.53	1.45	19.49	4.69
SSU56	30	25	1	24	259.57	262.97	1.32	21.03	6.70
SSU65	30	27	0	27	292.51	297.35	1.64	31.09	7.36
SSU66	30	27	0	27	302.18	308.71	2.14	31.67	7.72
SSU88	30	24	0	24	418.97	424.28	1.24	57.37	18.29
SSU108	30	25	0	25	506.84	512.10	1.01	100.11	25.48
SSU1010	30	22	0	22	551.92	554.87	0.56	98.01	29.51
MSU33	30	26	6	20	160.76	162.74	1.23	3.38	0.51
MSU34	30	28	4	24	185.23	187.42	1.18	5.13	0.97
MSU44	30	26	0	26	239.03	242.31	1.32	12.76	2.16
MSU55	30	28	0	28	309.46	315.96	2.07	31.04	5.66
MSU56	30	29	0	29	321.72	329.32	2.32	30.41	4.90
MSU65	30	28	0	28	367.47	375.24	2.07	49.92	7.51
MSU66	30	28	0	28	386.52	396.05	2.42	49.08	8.39
MSU88	30	29	0	29	529.09	546.14	3.08	102.58	20.49
MSU108	30	29	0	29	655.03	667.53	1.88	175.06	39.82
MSU1010	30	25	1	24	687.61	698.20	1.52	164.46	40.30
LSU33	30	22	5	17	228.67	229.98	0.56	4.98	0.80
LSU34	30	27	5	22	239.38	241.19	0.76	5.50	0.83
LSU44	30	28	4	24	324.13	327.27	0.93	16.36	1.95
LSU55	30	28	0	28	419.19	426.50	1.70	48.32	4.75
LSU56	30	27	0	27	440.70	449.42	1.95	50.98	5.80
LSU65	30	29	0	29	496.02	507.07	2.17	92.34	6.75
LSU66	30	27	0	27	518.88	531.96	2.48	87.37	8.96
LSU88	30	29	0	29	715.14	736.53	2.90	180.73	22.54
LSU108	30	27	0	27	883.67	905.50	2.40	328.03	64.14
LSU1010	30	29	0	29	947.02	972.93	2.66	295.34	61.76
Average	**30**	**26.80**	**1.03**	**25.77**	**403.62**	**411.39**	**1.70**	**70.36**	**13.76**

6.3 蟻群最佳化演算法的結果分析

在本章節中我們將嘗試解釋為何 ACO_S_{new} 比 ACO_S_{old} 好的原因。經過詳細的分析，本研究同樣發現了兩個主要的推論(Chen *et al.* 2012)。第一個推論為工件族間的排序為「**主導區段特性**」，而這個假設也可以從圖 5.5 得到證實。本研究沿用這樣的結果來分析為何 ACO_S_{new} 會比 ACO_S_{old} 績效好的原因。因此，本研究將著重在分析主導區段。

而第二個推論是 ACO_S_{old} 的網絡有「**高濃度運輸路徑特性**」，亦即，ACO_S_{old} 的平均運輸路徑費洛蒙濃度高於 ACO_S_{new} 的網絡。在 ACO_S_{old} 中，我們考慮一個簡單的排程問題，其中包含 4 個工件族和 20 個工件。因此，可以規劃出 5 個子網絡而第一個子網絡是統治區段，如圖 6.1。在這個子網絡中，所有的旅行路徑為 C (5, 2) = 10 條；而每個世代的總螞蟻數目為 C (20, 2) = 190 隻。因此，ACO_S_{old} 網絡中的平均運輸集中率為平均每條路徑有 19 隻螞蟻經過(190/10=19)。相反地，在 ACO_S_{new} 中只有一個大的網絡，其中包含 21 個節點。在每個進化世代，其旅行路徑的總數為 C (21, 2) = 210，而總螞蟻數目為 C (20, 2) = 190 隻。因此，其網絡中的平均運輸集中率為平均每條路徑有 0.9 隻螞蟻經過(190/210=0.90)，遠低於在 ACO_S_{old} 中的數值。

為了方便解釋高濃度運輸路徑特性的特性，本研究先定義蟻群最佳化演算法的兩個符號：$\hat{\tau}_{ij}$和$\bar{\tau}_{ij}$。當蟻群最佳化演算法終止時，每條路徑都有最後的費洛蒙濃度τ_{ij}(從節點 i 到節點 j 的路徑)；本研究定義$\hat{\tau}_{ij} = \tau_{ij} / \sum_{k \neq i, k \in S} \tau_{ik}$，其中 S 表示網絡中所有的節點集合，$\hat{\tau}_{ij}$值越高表示螞蟻選到此路徑的機會越高。以問題 MSU9 中的一個實例為例子，我們可以得到一個從至矩陣如表 6.2。根據此表，本研究可以根據費洛蒙濃度依序推論出**重要旅行途徑**；所以在這 C (N, 2)條路徑中，我可以選擇 N 條路徑組合成重要旅行途徑如表 6.2 中被標記的路徑。而根據這條重要旅行途徑，本研究可以定義一個加總參數：$\bar{\tau}_{ij} = \sum_{\hat{\tau}_{ij} \in D} \hat{\tau}_{ij} / N$，其中 D 表示在這重要旅行途徑上的所有路徑集合，而$\bar{\tau}_{ij}$表示重要旅行途徑上的平均運輸

路徑費洛蒙濃度。

表 6.2 MSU9 情境下一個實例的 ACO_S_{old} 從至圖

From-to Chart		To									
		F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
From	Starting Node	99.97%	0.00%	0.00%	0.00%	0.03%	0.00%	0.00%	0.00%	0.00%	0.00%
	F1	-	0.07%	0.00%	0.00%	68.94%	12.22%	11.22%	7.25%	0.25%	0.04%
	F2	0.09%	-	89.74%	1.98%	3.01%	0.40%	4.49%	0.02%	0.00%	0.27%
	F3	0.00%	49.79%	-	50.21%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	F4	0.00%	1.10%	50.12%	-	48.79%	0.00%	0.00%	0.00%	0.00%	0.00%
	F5	49.97%	1.66%	0.00%	48.37%	-	0.01%	0.00%	0.00%	0.00%	0.00%
	F6	12.60%	0.31%	0.00%	0.00%	0.01%	-	38.30%	48.51%	0.23%	0.04%
	F7	10.50%	3.18%	0.00%	0.00%	0.00%	34.77%	-	50.34%	1.01%	0.19%
	F8	6.62%	0.02%	0.00%	0.00%	0.00%	42.96%	49.11%	-	0.28%	1.02%
	F9	0.51%	0.00%	0.00%	0.00%	0.00%	0.44%	2.17%	0.62%	-	96.25%
	F10	0.08%	0.41%	0.00%	0.00%	0.00%	0.08%	0.42%	2.26%	96.74%	-

本研究將針利用第二個推論來分析為何 ACO_S_{new} 和 ACO_S_{old} 的網絡。首先，本研究將比較 ACO_S_{new} 和 ACO_S_{old} 的 $\bar{\tau}_{ij}$，以 MSU9 情境中的一個實例為例子，其結果如圖 6.3 所示。由圖中可知，ACO_S_{old} 的 $\bar{\tau}_{ij}$ 值都在 50%左右，然而 ACO_S_{new} 的 $\bar{\tau}_{ij}$ 值只有 20%左右。而本研究進一步對所有實驗情境作測試，其結果如表 6.3；結果發現在 ACO_S_{old} 中主要旅行途徑上的 $\bar{\tau}_{ij}$ 明顯高於 ACO_S_{new}；ACO_S_{old} 的 $\bar{\tau}_{ij}$ 平均值為 62.37%左右，然而 ACO_S_{new} 的 $\bar{\tau}_{ij}$ 平均值只有 18.71%左右。亦即，ACO_S_{old} 有比較高的機率會選擇該旅行途徑；而相對而言，ACO_S_{new} 比較發散，因此也比 ACO_S_{new} 容易產生不同的旅行途徑。假設主導區段陷入區域最佳解時，ACO_S_{old} 比 ACO_S_{new} 有較低的機率可以產生更好的解。而此區域最佳解的現象可以由表 5.1 和 6.1 的比較可以驗證；基因演算法的績效都比蟻群最佳化演算法的績效好，亦即，蟻群最佳化演算法容易陷入區域最佳解。

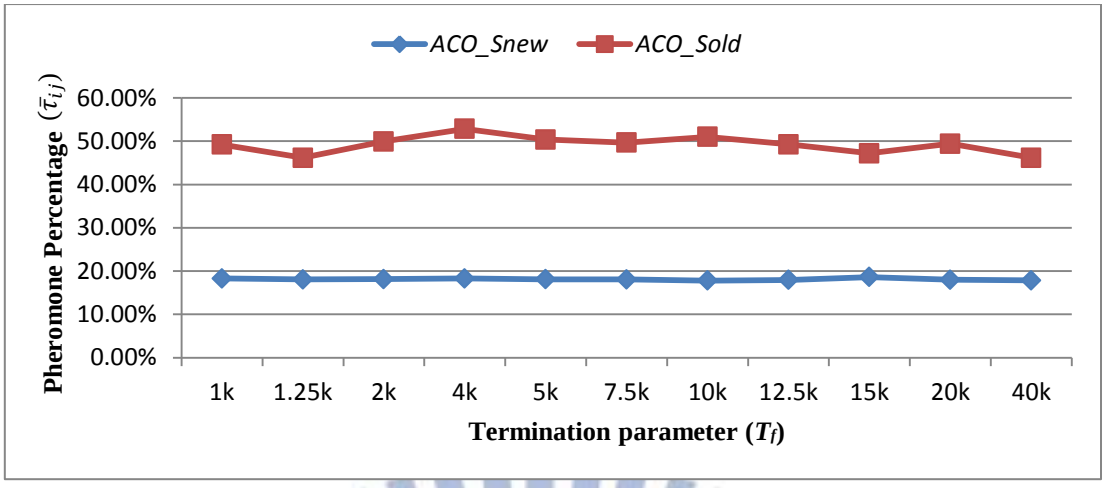


圖 6.3 比較 ACO_S_{new} 和 ACO_S_{old} 的 $\bar{\tau}_{ij}$ 在不同的 T_f

我們進一步分析發現支持 ACO_S_{new} 比較發散的現象。如圖 6.4 所示，當終止條件增加時，ACO_S_{new} 不斷的進步產生更好的解。然而 ACO_S_{old} 很快地趨於穩態，亦即，當終止條件為 $T_f = 1k$ 時的最佳解與終止條件為 $T_f = 40k$ 時的最佳解很接近；也就是說 ACO_S_{old} 很快地就陷入區域最佳解。

圖 6.4 比較 ACO_S_{new} 和 ACO_S_{old} 的解品質在不同的 T_f

表 6.3 蟻群最佳化演算法的實驗結果

Scenario	ACO_S_{new}	ACO_S_{old}
SSU0	21.70%	79.45%
SSU1	20.89%	77.88%
SSU2	19.55%	68.91%
SSU3	18.64%	63.17%
SSU4	18.58%	62.16%
SSU5	18.16%	60.02%
SSU6	18.18%	59.39%
SSU7	17.62%	50.70%
SSU8	17.36%	46.35%
SSU9	17.18%	45.60%
MSU0	22.11%	79.19%
MSU1	21.33%	79.51%
MSU2	19.74%	70.54%
MSU3	18.28%	64.02%
MSU4	18.73%	63.75%
MSU5	17.81%	59.33%
MSU6	17.46%	59.88%
MSU7	17.15%	53.49%
MSU8	16.64%	47.83%
MSU9	16.84%	47.53%
LSU0	20.71%	79.65%
LSU1	22.20%	80.18%
LSU2	20.86%	71.23%
LSU3	18.69%	64.32%
LSU4	18.64%	64.51%
LSU5	17.52%	61.14%
LSU6	18.19%	60.61%
LSU7	17.30%	54.96%
LSU8	16.87%	47.56%
LSU9	16.43%	48.40%
Average	**18.71%**	**62.37%**

第七章 *GA_TS* 進化演算法與結果分析

本研究的第一個重點發現新表達法搭配不同進化演算法可以改善解的品質；然而過去研究卻沒有討論如果同時改善進化演算法之後，表達法的改善是否還具有相同的結論。因此，本研究將改善進化演算法的演化流程，搭配兩種表達法來討論改善表達法的績效；本研究混合使用基因演算法與禁忌搜尋法 (*GA_TS*)來搭配新舊表達法。本章節接下來要介紹整個 *GA_TS* 演算法的流程，之後再介紹其結果與結論。

7.1 *GA_TS* 演算法的流程

過去發展混合進化演算法的演算法搭配有很多種，但是搭配通常是一種全域搜尋法搭配一種區域搜尋法，此搭配能不陷入區域最佳解，同時也有微調的功能。因此本研究用基因演算法(全域搜尋法)搭配禁忌搜尋法(區域搜尋法)來改善進化演算法的進化機制。然後就算使用此兩種演算法搭配的方式也有區分；有一種演算法包含另一種演算法的方式，也有一種演算法搜尋完轉換另外一種演算法繼續進行搜尋。邊搜尋邊交互使用的例如：瀰集進化演算法，就是一種這樣的混合式進化演算法，其作法是將禁忌搜尋法包含在基因演算法之中，也就是在取得新子代後再用禁忌搜尋法去做微調，然後在找下一個世代。而此種方法通常需要大量的時間，因此本研究使用另外一種方法，先執行基因演算法之後，將其最佳解當作參考解，在進行禁忌搜尋法做微調。

而本研究的混合進化演算法的流程如圖 7.1，先使用基因演算法執行，而其流程如 5.1.1 節所介紹的流程，但修改其終止條件設為$T_f = T_f^G$。之後再利用禁忌搜尋法繼續進化，其流程如 4.1.2 節介紹的流程；而不同的是其初始解為基因演算法所產生的最佳解，而其終止條件設為$T_f = T_f^{TS}$。而此混合進化演算法所搭配 S_{old} 和 S_{new} 的流程，是根據對應的相關演算法流程做搭配，為了方便解釋，本研

究將其搭配 S_{old} 命名為 *GA_TS_*S_{old} 而 S_{new} 為 *GA_TS_*S_{new}。

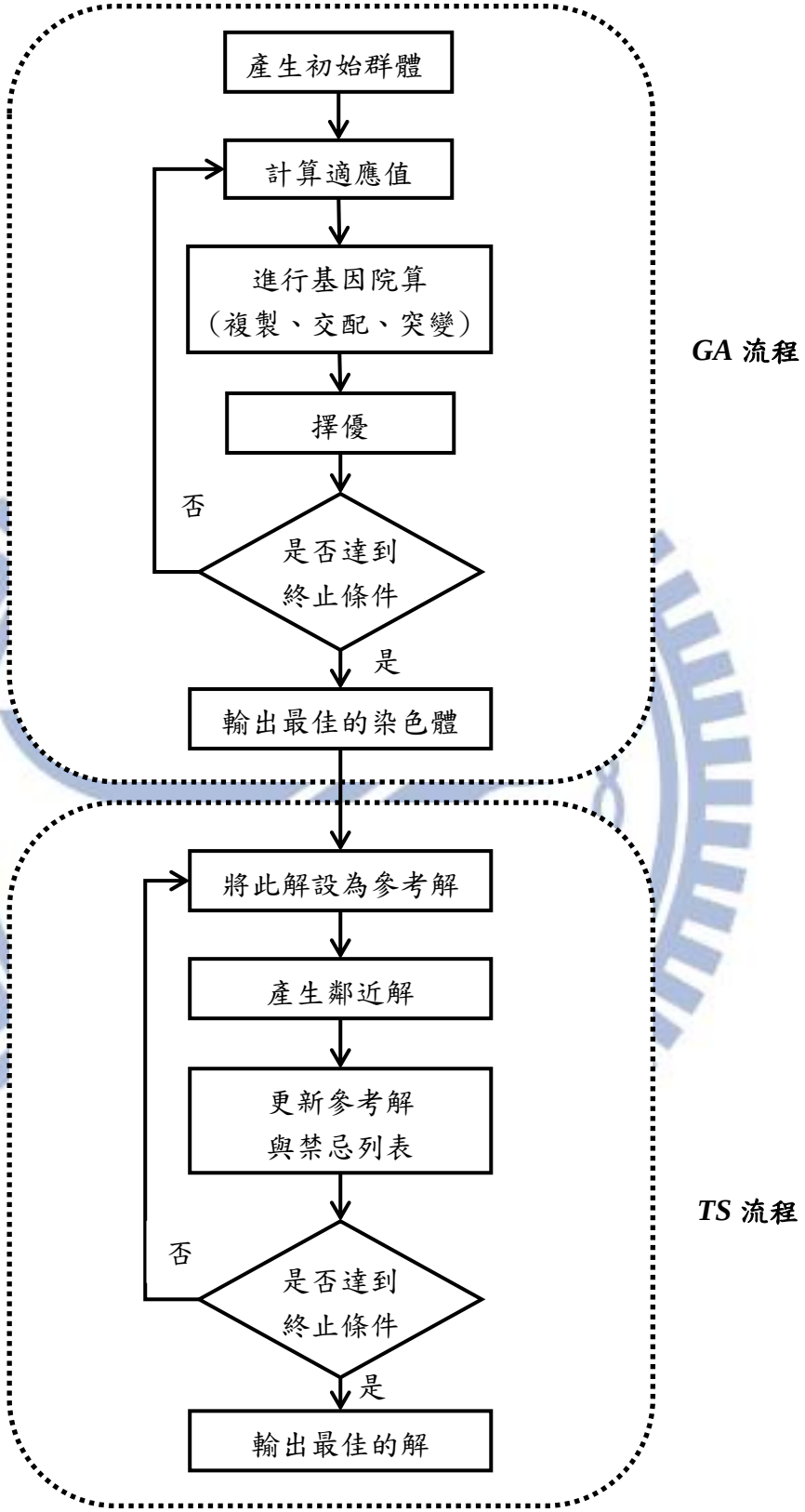


圖 7.1 *GA_TS* 進化演算法的流程

7.2 *GA_TS* 進化演算法的實驗情境與結果

本研究的目標值是最大完工時間(makespan)，比較 *GA_TS_S*$_{old}$ 和 *GA_TS_S*$_{new}$ 的績效表現。其基因演算法流程相關參數設定如下：P_{size} = 1,000、p_c = 0.95、p_m = 0.10 和T_f^G= 3,000,000。而禁忌搜尋法流程的相關參數設定如下：$T_f^{TS} = 2{,}000$且 $q_{size} = C_2^{n_s} \times 0.3$。本研究使用 C++語言撰寫，而實驗的電腦配備為 AMD Athlon(tm) Ⅱ*4640 3.0Ghz CPU 以及 4G Ram。

實驗資料的來源是跟據 Schaller *et al.* (2000)的文獻，分成 30 個情境，而每個情境包含 30 個實例。在這 30 個情境中，我們以 *X-F-m* 來表示其所代表的不同情境。*X* 表示不同的整備時間(LSU,MSU,SSU)，*F* 表示工件族的數目，*m* 表示機台的數目—有多少加工階段。SSU 表示整備時間最短，MSU 表示整備時間為中間值而 LSU 表示整備時間最大。

在每個情境的 30 個實例中，本研究隨機產生以下的相關參數：每個工件族中的工件數目(n_f)、生產時間與每個工件族之間的整備時間。其中，n_f是採用間斷的單一分配(uniform)函數產生 U[1,10]；而工件在每一個加工階段的生產時間也是採用間斷的單一分配函數產生 U[1,10]。而整備時間則根據不同的整備時間大小採用不同的方案，在 SSU 的時候是採用 U[1,20]、在 MSU 的時候是 U[1,50]，而大整備時間 LSU 則是 U[1,100]。為了避免實驗受到亂數的影響，因此每一個實例都需要用 15 個不同的亂數種子(seed)來進行實驗。將此 15 個結果取平均做為此實例的績效，然後再將每個情境的 30 個實例平均起來做為最後評估的績效值。

表 7.1 顯示了實驗的結果。而其參數解釋如下：*GA_TS_S*$_{old}$ 所獲的的績效為C_t，*GA_TS_S*$_{new}$ 所獲的的績效為C_n；而其對應的計算時間分別為T_t和T_n。而為了比較此兩種演算法的績效，此研究設定一個參數$\gamma = (C_t - C_n)/C_t$。此研究還比較每個情境之中，新舊表達法在不同實例的績效表現。$N = 30$表示每個情境有

30 個實例，而N_e表示$\gamma = 0$的實例個數，而N_w代表$\gamma > 0$的實例個數。$N_w + N_e$為 *GA_TS_S*$_{new}$ 與 *GA_TS_S*$_{old}$ 比較勝過或平手的總數量。如果$N_w + N_e$越大表示 *GA_TS_S*$_{new}$ 有明顯的贏過 *GA_TS_S*$_{old}$。

由表 7.1 了解在 30 個情境下，γ 值最好為 0.16%，在其中有四個情境下，*GA_TS_S*$_{old}$ 有較好績效表現。但 γ 值總平均還是有 0.03%，表示大部分情況下，*GA_TS_S*$_{new}$ 有比 *GA_TS_S*$_{old}$ 績效好。且在 30 個情境下的 30 個實例的總平均 $N_w + N_e = 26.40$，表示 *GA_TS_S*$_{new}$ 有比 *GA_TS_S*$_{old}$ 差的實例個數不多。本研究也將此實驗做統計分析，其 t 值為$t_0 = 3.66 > t_{0.025,899} = 1.96$，這表 *GA_TS_S*$_{new}$ 在 95%的信心水準下明顯的比 *GA_TS_S*$_{old}$ 績效好。而 *GA_TS_S*$_{new}$ 和 *GA_TS_S*$_{old}$ 運算時間都非常短，因此本研究不討論運算時間。

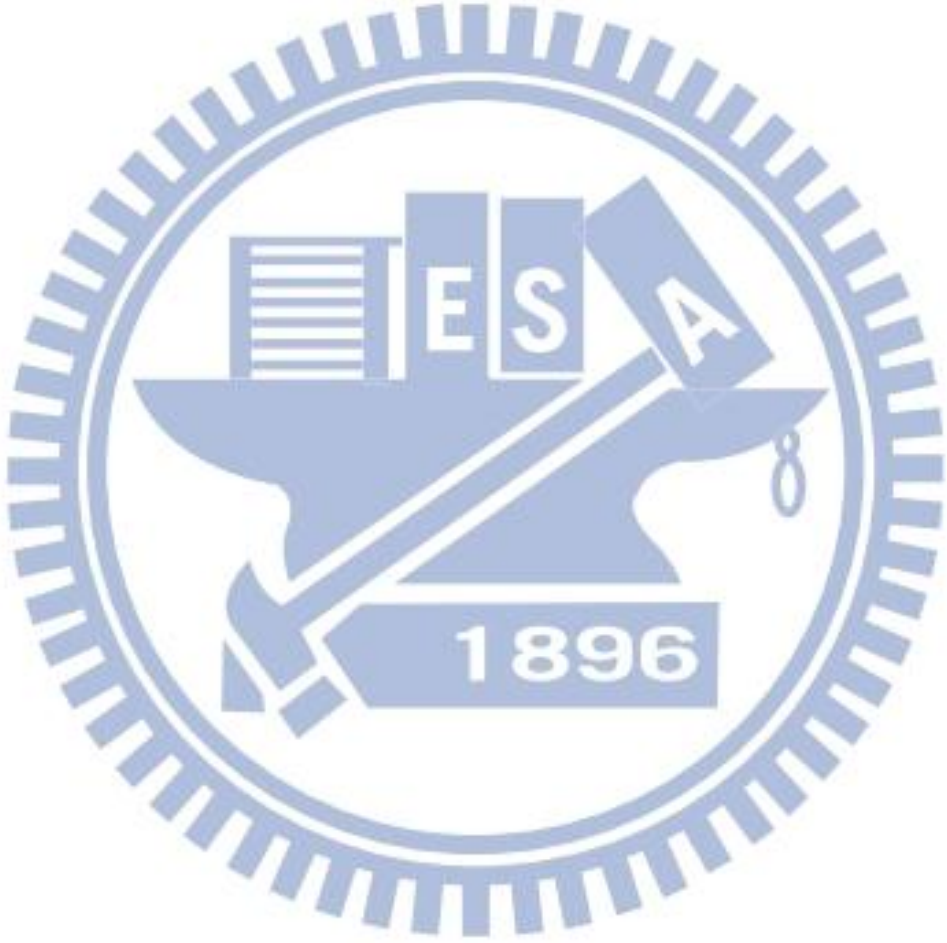
本研究也討論 *GA_TS* 進化演算法和本研究其他進化演算法的比較，如表 7.2 所示。由此表可以了解 *GA_TS_S*$_{new}$ 有最好的績效表現，表示同時改善進化演算法與解表達法有最好的績效。而進化演算法搭配 *S*$_{new}$ 的績效都比搭配 *S*$_{old}$ 的績效好。由上述的結果可知，改善表達法會改善績效品質。因此，未來的研究應該著重於同時改善進化演算法的流程與表達法，經由這樣的設計可以獲得更好的解品質。而過去大部分研究都著重在進化演算法的流程改善，很少文獻做表達法的改善。此外，過去文獻也沒有討論同時改善兩個面相時的績效，這也是本研究的貢獻。

表 7.1 *GA_TS* 演算法的實驗結果

Scenario	*Makespan*							Computation Time	
	N	***Nw+Ne***	***Ne***	***Nw***	***Cn***	***Ct***	***γ (%)***	***Tn***	***Tt***
SSU33	30	30	30	0	134.47	134.47	0.00	11.75	14.67
SSU34	30	28	25	3	150.98	151.00	0.01	13.72	16.35
SSU44	30	29	25	4	185.63	185.64	0.01	17.97	19.68
SSU55	30	28	18	10	241.90	242.09	0.07	28.42	26.49
SSU56	30	28	12	16	253.36	253.69	0.13	29.58	28.55
SSU65	30	28	14	14	285.77	285.95	0.06	36.80	31.49
SSU66	30	28	11	17	294.87	295.12	0.09	40.68	33.57
SSU88	30	22	6	16	402.46	403.12	0.15	80.40	50.85
SSU108	30	17	1	16	481.61	481.92	0.06	134.05	65.27
SSU1010	30	23	1	22	521.73	522.42	0.13	161.38	76.64
MSU33	30	30	30	0	160.00	160.00	0.00	10.83	14.09
MSU34	30	30	29	1	184.67	184.70	0.01	12.69	15.66
MSU44	30	30	29	1	237.60	237.60	0.00	17.87	19.54
MSU55	30	29	25	4	306.08	306.15	0.03	27.37	25.83
MSU56	30	30	24	6	317.47	317.68	0.06	27.61	27.08
MSU65	30	28	24	4	362.63	362.66	0.01	35.84	30.79
MSU66	30	28	19	9	380.02	380.06	0.01	38.76	33.26
MSU88	30	23	10	13	510.21	510.45	0.04	78.20	50.20
MSU108	30	15	2	13	623.88	623.49	-0.07	128.69	64.35
MSU1010	30	22	2	20	655.00	655.48	0.07	143.50	72.21
LSU33	30	30	30	0	228.03	228.03	0.00	12.39	15.10
LSU34	30	30	30	0	239.00	239.00	0.00	11.91	15.05
LSU44	30	29	29	0	323.44	323.43	-0.00	17.02	19.06
LSU55	30	29	28	1	415.97	415.97	0.00	25.62	25.28
LSU56	30	30	24	6	436.98	437.16	0.04	28.61	27.44
LSU65	30	30	27	3	491.92	492.02	0.02	35.94	30.70
LSU66	30	28	24	4	514.32	514.34	0.00	36.47	32.50
LSU88	30	22	9	13	701.62	701.11	-0.07	72.55	48.51
LSU108	30	16	2	14	847.61	847.33	-0.04	121.70	62.46
LSU1010	30	22	1	21	907.14	908.63	0.16	138.82	71.05
Average	**30**	**26.40**	**18.03**	**8.37**	**393.21**	**393.36**	**0.03**	**52.57**	**35.46**

表 7.2 各種進化演算法的實驗結果比較

Algorithm	*Makespan*							Computation Time	
	N	*Nw+Ne*	*Ne*	*Nw*	*Cn*	*Ct*	*γ (%)*	*Tn*	*Tt*
TS	30	30.00	0.70	29.30	400.07	412.17	2.76	26.65	2.14
GA	30	26.27	17.97	8.30	393.25	393.37	0.03	44.37	41.13
ACO	30	26.80	1.03	25.77	403.62	411.39	1.70	70.36	13.76
GA_TS	30	26.40	18.03	8.37	393.21	393.36	0.03	52.57	35.46



第八章 結論與未來研究方向

在經過一連串的實驗與驗證，本章節將對本研究的問題與研究重點做相關的整理，並討論的未來可研究的方向。

8.1 結論

本研究想要研究的問題有兩個：(1) 討論為何新表達法在禁忌搜尋法之中會比舊表達法績效來的好；(2) 討論改善進化機制與改善解表達法的相關性。而討論這些問題都是以一個順序相依整備時間且固定序列之流線型單元製造系統的排程規劃問題為案例做研究。

由林耿漢(2011)的研究實驗中，我們可以發現 TS_S_{new} 比 TS_S_{old} 績效好，但此研究並沒有討論其原因，因此本研究想深入的分析其可能的原因。經過了詳細的步驟檢查，本研究發現了迴圈的現象，進而設計了一個驗證迴圈的方法，成功的驗證了迴圈的影響性。除此之外，本研究更仔細的分析找出其發生迴圈的原因—組合模組的自由度；再利用放大禁忌列表的儲存容量來驗證本研究的假說，發現本研究的假說是正確的。因而成功的解釋為何 TS_S_{new} 比 TS_S_{old} 績效好的原因。

而在戴邦豪(2010)的研究中，經由分析可以發現 S_{old} 有兩個特性：(1) 主導區段特性；(2) 同質化現象的特性。這兩項特性的結合導致 GA_S_{new} 比 GA_S_{old} 好的原因。同時也發現 GA_S_{new} 的計算較沒有效率，但可以找到更好的解；由此可以推論出若先使用基因演算法搭配 S_{old} 之後再換成 S_{new} 會有最好的效果，也就是 $GA\_S_{old\text{-}new}$ 會有最好的績效，而此結果也由戴邦豪的實驗結果中可以驗證此推論。

在李奕勳(2011)的研究中，由於使用不同的進化演算法，可以發現在 S_{old} 中和基因演算法有同一個特性：主導區段特性；和另外一個特性：高濃度運輸路徑

特性。這兩項特性的結合導致 ACO_S_{new} 比 ACO_S_{old} 較容易跳脫區域最佳解，進而推論出為何 ACO_S_{new} 的解品質較好的原因。

除了原因的解釋，本研究還想討論如何改善進化機制與改善解表達法之間會有怎樣的關聯性。整理了過去相關研究，了解在不同演算法之中，S_{new} 績效都比 S_{old} 好。但如果改善了其進化機制後，表達法是否也有其影響性。因此本研究結合了基因演算法和禁忌搜尋法，搭配兩種不同的表達法，結果顯示 S_{new} 一樣比 S_{old} 有較好的績效。

本研究除了解釋為何在各種不同進化演算法中，S_{new} 比 S_{old} 績效好之外，還發現如果改善進化機制之後，改善解的表達法一樣有效果，如此就能跳脫過去傳統的思維。過去大部分研究都在改善進化機制，發展了許多很複雜的搜尋方法，然而本研究提供的一個新的思維，除了改善進化機制，同時改善解表達法一樣會有相當好的效果。

8.2 未來研究方向

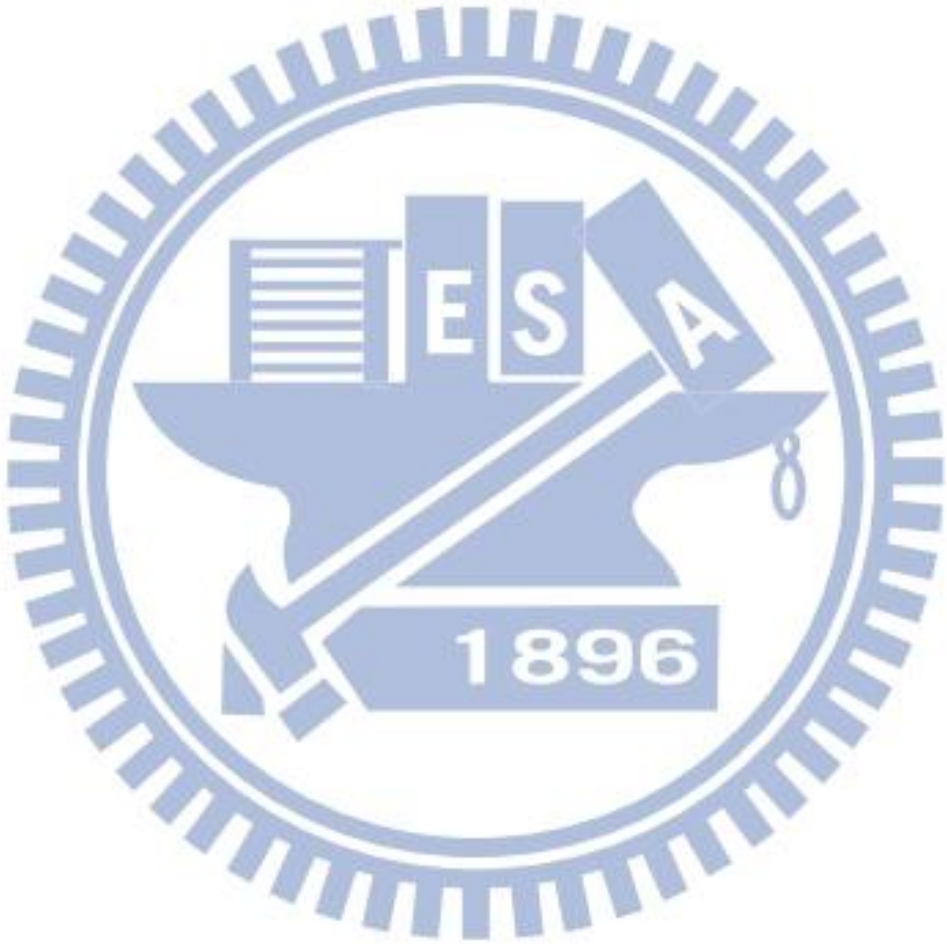
而最直接可以延伸的未來研究方向就是在搭配不同的進化演算法或混合進化演算法，進而了解此新舊表達法的現象是不是在所有的進化演算法之中都有一樣的結果。

未來研究也可以經由本研究的分析來針對本研究的排程問題做相關的調整。在本研究的分析中，可以發現此問題有「**主導區段**」的現象發生；因此可以著重在先搜尋主導區段之後，在搜尋其他區段的排序。然而在此流程上必須考慮搜尋主導區段時，其他區段該如何調整，而這部分可做為未來研究的重要議題。

而針對不同的進化演算法搭配 S_{old} 時，也有不同的研究方向。在禁忌搜尋法中，可以設計避免迴圈發生的演算流程，來使其解品質可以獲得改善。而在基因演算法中，可考慮如何設計流程可以避免同質化的產生，進而改善解品質。最後

在蟻群最佳化進化演算法中，如何以避免掉入區域最佳解為研究的重點。

而本研究以一個順序相依整備時間且固定序列之流線型單元製造系統的排程規劃問題為案例做研究。未來可以針對不同的問題，設計解的許多不同種新表達法，將可能會發現較好的績效表現，而不需要使用更複雜的進化演算法進化機制。



參考文獻

英文文獻

Brindle, A., Genetic algorithms for function optimization. Doctoral dissertation, University of Alberta, Edmonton, Canada. Unpublished doctoral dissertation; 1981.

Chan, F.T.S., Choy, K.L., Bibhushan, A genetic algorithm-based scheduler for multiproduct parallel machine sheet metal job shop. *Expert Systems with Applications*; 2011; 38: 8703-8715.

Chang, P.C., Chen, S.H., Fan, C.Y., Mani, V., Generating artificial chromosomes with probability control in genetic algorithm for machine scheduling problems. *Annals of Operations Research*; 2008: 1-15.

Chen, C.F., Wu M.C., Li Y.H., Tai P.H., Chiou C.W., A comparison of two chromosome representation schemes used in solving a family-based scheduling problem. *Robotics and Computer-Integrated Manufacturing*; 2012: doi:10.1016/j.rcim.2012.04.009.

Cordeau, J.F., Maischberger M., A parallel iterated tabu search heuristic for vehicle routing problems. *Computers & Operations Research*; 2012; 39: 2033-2055.

Demir, L., Tunalı, S., Eliiyi, D.T., An adaptive tabu search approach for buffer allocation problem in unreliable non-homogenous production lines. *Computers & Operations Research*; 2012; 39: 1477-1486.

Dorigo , M., Gambardella, L.M., Ant colony system: A cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*; 2002; 1(1): 53-66.

Dorigo , M., Maniezzo, V., Colorni, A., Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*; 2002; 26(1): 29-41.

Franca, P.M., Gupta, J.N.D, Mendes, A.S., Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups. *Computers and Industrial Engineering*; 2005; 48 (3): 491–506.

Gajpal, Y., Rajendran, C., An ant-colony optimization algorithm for minimizing the completion-time variance of jobs in flowshops. *International Journal of Production Economics*; 2006; 101(2): 259-272.

Glover, F., Tabu search Part I. *ORSA Journal of Computing*; 1989; 1: 190-206.

Glover, F., Tabu search Part II. *ORSA Journal of Computing*; 1990; 2: 4-32.

Hendizadeh, S.H., Faramarzi, H., Mansouric, S.A., Gupta J.N.D., ElMekkawy T.Y., Meta-heuristics for scheduling a flowline manufacturing cell with sequence dependent family setup times. *International Journal of Production Economics*; 2008; 111 (2): 593–605.

Holland J.H., Adaptation in Neural and Artificial Systems. University of Michigan Press, Ann Arbor, Michigan; 1975.

Kim, Y.K., Kim, J.Y., Kang, S.S., A Tabu search Approach for Designing a Non-Hierarchical Video-on-Demand Network Architecture. *Computers industrial Engineering*; 1997; 33 (3-4): 837-840.

Lin, S.W., Ying, K.C., Lee, Z.J., Metaheuristics for scheduling a non-permutation flowline manufacturing cell with sequence dependent family setup times. *Computers & Operations Research*; 2009a; 36: 1110-1121.

Lin, S.W., Gupta, J.N.D., Ying, K.C., Lee, Z. J., Using simulated annealing to schedule a flowshop manufacturing cell with sequence dependent family setup times. *International Journal of Production Research;* 2009b; 47 (12): 3205-3217.

Logendran, R., deSzoeke, P., Barnard, F., Sequence-dependent group scheduling problems in flexible flow shops. *International Journal of Production Economics*; 2006; 102 (1): 66–86.

Merkle, D., Middendorf, M., On solving permutation scheduling problems with ant colony optimization. *International Journal of Systems Science*; 2005; 36(5): 255-266.

M'Hallah, R., Al-Khamis, T., Minimising total weighted earliness and tardiness on parallel machines using a hybrid heuristic. *International Journal of Production*

Research; 2011; 1: 1-26.

Reddy, V., Narendran, T.T., Heuristics for scheduling sequence-dependent set-up jobs in flow line cells. *International Journal of Production Research*; 2003; 41 (1): 193–206.

Schaller, J.E., Gupta, J.N.D., Vakharia, A.J., Scheduling a flowline manufacturing cell with sequence dependent family setup times. *European Journal of Operational Research*; 2000; 125 (2): 324–339.

Syswerda, G., Uniform crossover in genetic algorithms, in: J.D. Schaffer (Ed.), Proceeding of the 3rd International Conference on Genetic Algorithms; 1989: 2–9.

Tavakkoli-Moghaddam, R., Taheri, F., Bazzazi, M., Izadi, M., Sassani, F., Design of a genetic algorithm for bi-objective unrelated parallel machines scheduling with sequence-dependent setup times and precedence constraints. *Computers &Operations Research*; 2009; 36: 3224-3230.

Wen, U.P., Huang, A.D., A simple Tabu Search method to solve the mixed-integer linear bilevel programming problem. *European Journal of Operational Research*; 1996; 88: 563-571.

Yagmahan, B., Yenisey, M. M. A multi-objective ant colony system algorithm for flow shop scheduling problem. Expert Systems with Applications; 2010; 37(2): 1361-1368.

中文文獻

戴邦豪，「應用混合式染色體表達法漁具順序相依家族整備時間之流線型製造單元排程」，國立交通大學工業工程與管理學系，碩士論文，2010。

林耿漢，「以塔布搜尋法求解流線型製造單元排程」，國立交通大學工業工程與管理學系，碩士論文，2011。

李奕勳，「以蟻群最佳化演算法求解流線型製造單元排程」，國立交通大學工業工程與管理學系，碩士論文，2011。

巫木誠，「以進化搜尋演算法求解流線群組排程問題」，國科會專題研究計畫申請書，計畫編號：99-2221-E-009-110-MY3，2010。