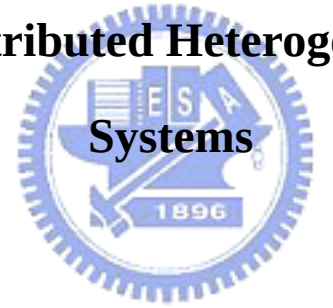
國立交通大學

工業工程與管理學系

碩士論文

DHC 系統之任務指派與排程的新型基因演算法

A Novel Genetic Algorithm for Task Matching and Scheduling in Distributed Heterogeneous Computing Systems



研究生：吳政翰

指導教授：巫木誠 博士

中華民國九十五年六月

DHC 系統之任務指派與排程的新型基因演算法

A Novel Genetic Algorithm for Task Matching and Scheduling in Distributed Heterogeneous Computing Systems

研 究 生：吳政翰　　Student : Chun-Hsung Wu

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 2006

Hsin-Chu, Taiwan, Republic of China

中 華 民 國 九 十 五 年 六 月

DHC 系統之任務指派與排程的新型基因演算法

研究生：吳政翰　　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

摘要

分散式異質計算系統(distributed heterogeneous computing, DHC)是一個將大型應用程式解構成多個子程式分別計算的架構，異質是指計算機器種類不同，分散是指機器的位置不同，因此互通的資料需要傳輸時間。DHC 系統有兩個重要的決策：指派(matching)、排程(scheduling)。指派決策是決定將子程式分給哪一個機器執行，排程決策是訂定子程式的執行優序。指派與排程決策的解空間很大，過去多用基因演算法來求解。本研究提出一個新的基因演算法，此演算法產生新染色體是採用本研究所獨創的共識因子(consensus operator)，共識因子是一個導引(guided)機制，可據以產生品質好的染色體，因而加速基因演算法的求解品質和時間。實驗結果顯示，在求解 DHC 的指派與排程決策，此新的基因演算法，求解品質和求解時間都較過去的基因演算法為佳。

關鍵辭:異質系統、指派與排程、基因演算法、共識因子

A Novel Genetic Algorithm for Task Matching and Scheduling in Distributed Heterogeneous Computing Systems

Student：Chung-Haung Wu　　Advisor：Dr. Muh-Cherng Wu

Institute of Industrial Engineering
National Chiao Tung University

ABSTRACT

A DHC (distributed heterogeneous computing) system is a computing architecture, where computing machines are heterogeneous, located sparsely and thereby need data transfer among them. A large-scale application problem can be executed on a DHC by decomposing the program into several sub-programs. This needs to make two decisions—matching and scheduling. The matching decision assigns a machine for executing each subprogram, while the scheduling decision prioritizes the execution of subprograms. The matching and scheduling problem in DHC has a huge solution space. Various genetic algorithms have been proposed to solve the problem. This research proposes a novel genetic algorithm (GA), which is distinguished in generating new chromosomes by a so-called *consensus operator* developed by us—a mechanism that attempts to efficiently generate quality chromosomes. Experiment results indicate that the proposed GA, while solving the DHC matching and scheduling problem, outperforms the prior GAs both in solution quality and time.

Keywords: Genetic algorithm, DHC (distributed heterogeneous computing), matching, scheduling, consensus operator

誌謝

本論文得以順利完成，首先要感謝恩師 巫木誠教授悉心的指導，使我在研究過程中，不論是研究領域的相關知識或是做事的態度，讓學生在研究所兩年中得到最大的啟發，在此致上最崇高的敬意。同時感謝許錫美教授、彭德保教授，在論文口試時所給予的寶貴建議，讓本論文更臻完備。

感謝之前學長孫士雄的經驗傳承，博士班學長蘇泰盛、施昌甫的引領，學弟妹們，因為你們的出現，使得我的研究所生活更加多采多姿。並感謝我的摯友兼室友的豪君、啟峰、詠進，不管是學業中的討教學習，或是生活的相互陪伴，甚至研究中的問題幫助，都讓我能夠表現的更為出色。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、兄弟，由於你們的支持與鼓勵，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。

政翰　于風城交大

中華民國九十五年六月

目錄

中文摘要..... I

英文摘要..... II

誌謝..... III

目錄..... IV

圖目錄..... VI

表目錄..... VII

第一章 緒論..... 1

1.1 研究動機..... 1

1.2 研究目的..... 2

1.3 章節安排..... 2

第二章 文獻回顧..... 3

2.1 同步/逐步解題架構簡介..... 3

2.2 評估機制的演算法..... 4

2.3 搜尋機制的演算法..... 4

第三章 問題描述..... 6

第四章 指派的解法..... 9

4.1 DS 法則指派決策..... 9

4.2 DS 演算法的釋例 10
第五章 排序的解法 12
5.1 染色體表達法 12
5.2 適應函數 12
5.3 初始母體產生法 13
5.4 產生新染色體的共識因子 16
5.5 選擇染色體形成新母體的方法 23
5.6 終止條件 25
5.7 基因演算法流程 25
第六章 實做與驗證 27
6.1 模擬情境 27
6.2 實驗比較 29
6.3 交配因子與共識因子比較 32
第七章 結論與未來研究 33
參考文獻 34
附錄 37

圖目錄

圖 1.1 DAG 任務..... 1

圖 1.2 3 個機器的異質系統..... 1

圖 2.1 Algorithm = Evaluator + Search Engine..... 3

圖 3.1 有傳輸時間的 DAG 任務..... 6

圖 4.1 一個簡單的 DS 例子..... 11

圖 5.1 6 個任務的 DAG 與 machine list..... 15

圖 5.2 由任務的優先值轉換為染色體..... 16

圖 5.3 PWP_matrix:*M* 流程..... 18

圖 5.4 任務優先關係矩陣 *A*..... 18

圖 5.5 標準化後的 *M*..... 19

圖 5.6 PWP_matrix 的硬性限制與柔性限制..... 20

圖 5.7 PWP 任務優先值去排序:*q2*→*q3* →*q1* →*q4*..... 20

圖 5.8 選擇參考任務 q_r..... 23

圖 5.9 基因演算法架構圖..... 26

圖 6.1 實驗結果..... 30

圖 6.2 Proposed GA 比 PSGA 績效改善率..... 31

表目錄

表 3.1 3 個機器 6 個任務的預估執行時間表(6×3矩陣)..... 8

表 5.1 染色體 t_1→t_6→t_5→t_2→t_3→t_4..... 12

表 5.2 各任務的 *AvgE*、*t-level* 和 *b-level*..... 15

表 6.1 測試情境:100 任務 20 個機器 DHC 系統..... 29

表 6.2 95%信心水準 proposed GA 平均績效勝過比較方法..... 30

表 6.3 各方法在各情境的平均時間..... 32

第一章 緒論

1.1 研究背景

有限資源的專案排程(project scheduling)問題[1-4]可能會非常複雜，因為具有下列二個特性。第一、任務(task)之間具有相依關（dependent relationship）， 此等相依關係通常可用一個有向非循環圖來描述 (directed acyclic graph，簡稱DAG)，如圖 1.1 所示，該專案具有 6 個任務 (task)，任務間有執行順序上的限制，譬如說，任務 t_2 需要任務 t_1 與任務 t_6 都執行完畢，才能執行。第二、任務的執行具有共用資源的特性，如圖 1.2 所示，該專案所有的資源是 3 部機器，每個任務都可用任一部機器加工。

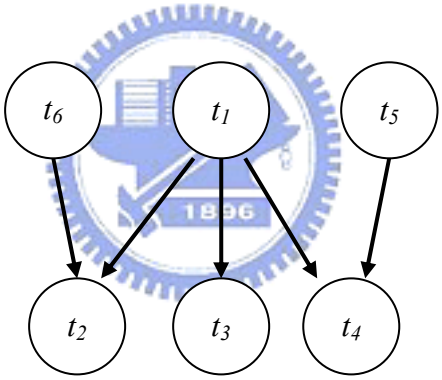


圖 1.1　DAG 任務

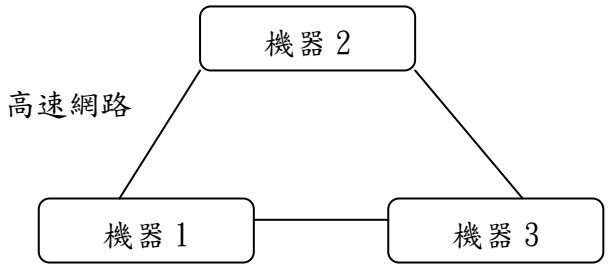


圖 1.2　3 個機器的異質系統

此種排程問題可分成兩個子問題：排序(sequencing)問題、指派問題

(matching)。排序問題是決定任務執行的先後順序，指派問題是將任務指派給機器。這類排程問題的複雜度(complexity)很高，因為 t 個任務排序最多有 $t!$ 組合，若該專案有 m 個資源，則有 m^t 個指派組合；結合排序與指派的決策，一專案的排程方案最多有 $t! \cdot m^t$ 種，解空間非常龐大。

上述專案排程問題可能發生在各種領域，譬如研發專案排程[5]、生產系統排程[6-9]、計算機系統排程[10-12]。在計算機領域，一個典型的案例是異質運算系統(distributed heterogeneous computing; DHC)的排程。所謂 DHC 系統是指由多部異質機器(具有不同運算能力的計算機) 、高速網路、通訊協定、作業系統，和程式環境所組成的一強大運算系統。此種計算系統執行一個程式時，先將程式切割成多個子程式，排定子程式的執行順序，並指派各子程式給不同機器處理，因為子程式間有相依關係，而且共有資源，因此 DHC 排程問題具有上述專案排程問題的特性，以下本研究稱此專案排程問題為 DHC 排程問題。

1.2 研究目的

DHC 排程問題過去已有一些文獻發表，但是在複雜的情境時，DHC 的解空間變得非常巨大，無論在求解時間和求解品質，過去的方法都尚有改善的空間。本研究的目的是發展一 DHC 排程的演算法，期能有效改善求解時間和品質。

1.3 章節安排

本論文其他章節安排如下：第二章是文獻回顧；第三章描述此問題的數學模型；第四章說明指派的演算法；第五章說明排序的演算法；第六章是實例驗證；第七章是結論與建議。

第二章　文獻回顧

本章回顧過去 DHC 排程與指派文獻的主要求解方法。DHC 排程與指派的特性是任務之間具有順序限制(sequence constraint)。根據 Kolisch[13]的研究，此等排程問題，用逐步求解法(serial method)會比用同步求解法(parallel method)好。以下分析同步與逐步兩種解題架構，並介紹在各解題架構下所發展的演算法。

2.1 同步/逐步解題架構簡介

同步求解法是同步求解指派問題和排序問題。此種求解法譬如 Wang[14]所提的基因演算法，此演算法的染色體是由兩個字串(string)組成，第一個為排程字串，第二個為指派字串，排程字串不得違反任務的順序限制。由於逐步求解法績效較佳，目前已成為求解順序限制(sequence constraint)排程問題的主流。

逐步求解法則是先求解指派問題，再求解排序問題；亦即給定一個任務順序，經過指派決策，可求得此任務順序的績效，因此指派決策基本上一個績效評估機制(performance evaluator)。根據此績效評估機制，排序問題就是在龐大解空間中，找出最佳的任務順序(task sequence)，所以排序決策基本上是一個搜尋機制(search mechanism)。因此一個逐步求解架構的演算法，如圖 2.1 所示是由兩個機制所構成：績效評估機制（以下簡稱 evaluator）、搜尋機制(以下簡稱 Search Engine)。

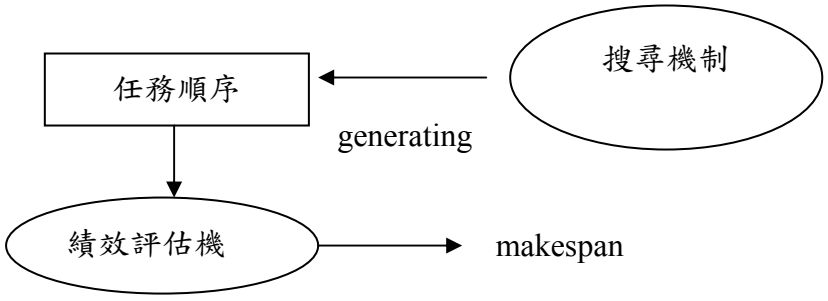


圖 2.1　Algorithm = Evaluator + Search Engine

2.2 評估機制的演算法

DHC 排程問題常見的評估機制有下列四種:LIT[16-17,26](Longest Idle Time), EST [23](earliest start time), EFT [23](earliest finish time), DS [15](dependent task scheduler)。

LIT 法把任務優先指派給閒置時間最長(longest idle machine)的機器，此評估機制相當著名而且簡單的。EST 是將任務指派給可以最早開始加工的機器。EFT 是將任務指派給可以最早完工的機器。DS 是 EFT 方法的修正，若能不增加目前總完工時間(make span)，優先將任務指派給加工時間(processing time)與傳輸時間相加最短的機器。若會增加目前總完工時間(make span)，用 EFT 指派。根據過去研究，上述的指派方法 DS 相對較佳。

2.3 搜尋機制的演算法

搜尋機制的各種方法基本尚可分為：啟發式(Heuristic approach)和演化式(evolutionary approach)和非演化式 (non-evolutionary approach)三大類。

啟發式一般皆為 LS(list scheduling)的延伸[17]，LS 為先對任務給定優先值，根據優先值排程。Sih and Lee 學者提出一個 DLS(dynamic Level scheduling)演算法 [11]，此方法根據任務先後關係和各機器執行時間的平均組合等級，以任務根據等級去排程。Topcuoglu 等學者也提出類似概念的 CPOP (Critical-Path-on-a-Processor)啟發法[25]去排程，根據為關鍵路徑的任務為優先排程。

Boyer 學者提出一個非演化式方法稱為 RS 演算法(random search algorithm) [15]，此方法以隨機方式產生一條「符合拓樸排序」(topological sort) [18]，所謂

「符合拓樸排序」是指排序符合 DAG 的拓樸結構，亦即該順序不違反任務的順序限制。RS 法不斷遞迴去比較新的解和目前的最佳解，如果找到更好的解，就取代目前的最佳解。此法所需的計算時間非常快速，而記憶體的利用也比一般基因演算法少很多。

演化式一般多用基因演算法[19-21]來求解，如 Shroff 等學者採用 genetic simulated annealing(GSA)，利用標準的 GA 去搜尋但用 SA 模擬退火法[22]來選擇染色體去更新母體。Dhodhi 等人提出 PSGA (problem-space genetic algorithm) 演算法[23]，他們提出 *b-value* 和 *t-value* 的概念，使起始母體(initial Population) 解的品質較佳。然後利用交配與突變等因子來更新母體。

過去基因演算法，都是利用交配運算子(crossover operator)[24] 和突變因子(mutation operator) 來產生新染色體。在 DHC 的排程指派問題，因為任務有順序限制，交配因子和突變因子可能產生兩個問題：(1)很容易產生不合理解，(2) 產生品質不佳的解。因此過去的方法在搜尋的效率上仍有改善的空間。

第三章 問題描述

在 DHC 系統中，應用程式通常分割為許多任務(task)，這些任務的關係可以被描述為有向非循環圖(DAG; directed acyclic graph)，表示為 $G = (T, <, E)$，$T=\{t_i, i=1,...,n\}$代表所有任務的集合，t_i 為第 i 個任務，n 為任務的個數。< 代表任兩節點的順序關係，例如: 任兩個任務 $t_i, t_k \in T$ ，當 $t_i < t_k$ 意指，t_i 為 t_k 的先行者(predecessor)，t_k 為 t_i 的後續者(successor)。所以 t_i 執行完畢後，t_k 才可以被執行。E 代表整個圖形所有的邊(edge)或弧(arc)的集合，而每個邊或弧都附加傳輸資料量 $D_{i,k}$，代表從任務 t_i 到任務 t_k 所需的傳輸資料量(以位元 byte 為單位)。圖 3.1 是一個 DAG 的釋例，例如 (1 < 3)代表任務 t_1 與任務 t_3 之間相連，且任務 t_1 執行完成之後，才可以執行任務 t_3，而任務 t_1 到任務 t_3 的傳輸資料量 $D_{1,3}$ 為 5。該圖中也明顯說明任兩任務並不需要完全連接，例如任務 t_5 為一個獨立的任務。

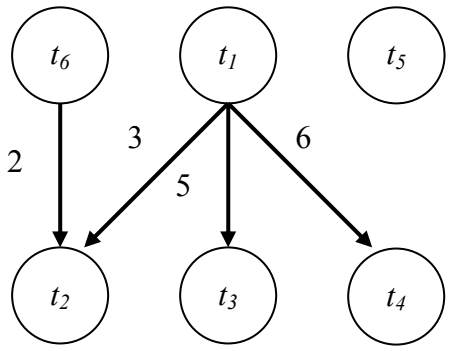


圖 3.1 有傳輸時間的 DAG 任務

DHC 運算系統是由一組獨立且不同型的機器由高速網路所整合的運算系統，M 為所有機器的集合，描述為 $M = \{m_j, j=1,...,m\}$ ，m 為機器的個數，m_j 為第 j 個機器。在 DHC 系統中，不同機器之間的頻寬 (bandwidth)跟據其網路(network)而有所不同，通常描述為 $m \times m$ 矩陣 $R_{m \times m}$。任兩任務之間的傳輸成本

(communication time)如下所示：

$$CommTime(t_i, t_k) = \frac{D_{i,k}}{R[H(i), H(k)]}$$

$$if \quad H(i) = H(k), then \qquad CommTime(t_i, t_k) = 0$$

上式中，$H(i)$為任務 i 被指派的機器，$R[H(i),H(k)]$為頻寬，$D_{i,k}$為從任務 t_i 到任務 t_k所需的傳輸資料量，當任兩任務 t_i, t_k 指派給同一個機器時，傳輸成本為0，代表並不需要傳輸資料。為了簡化問題，我們假設任兩機器的頻寬皆為 1.0，因此任兩任務之間的傳輸成本相等於其資料傳輸量，指派給不同機器不會影響傳輸時間。

因為是異質機器，同一任務在不同機器的期望執行時間(expected execution time)會不同。再者任務不能再拆解，亦即設每一個任務只能指派給一部機器執行。

每一個任務根據其所執行機器可以精確的估計其期望執行時間，期望執行時間可以一個 $n \times m$ 的矩陣 E 表達，其中 E_{ij} 代表任務 i 在機器 j 的期望執行時間（釋例見表 3.1）。

表 3.1　3 個機器 6 個任務的預估執行時間表(6×3 矩陣)

機器

任務	m_1	m_2	m_3
t_1	2	3	1
t_2	3	4	5
t_3	3	2	2
t_4	3	1	2
t_5	3	3	3
t_6	1	2	3

期望執行時間矩陣 E_{ij}

本排程部問題績效指標是總完工時間(makespan of the program)，目標函數如下所示：

$$T = \max(F_1,\ldots,F_m)$$

上式中，F_j 為機器 j 的全部完成時間(makespan)，$j=1,\ldots,m$。完成時間包括執行時間(processing time)和因有優先關係而產生的等待時間(waiting time)，其中等待時間包括了機器實際閒置時間(idle time)和資料傳輸時間(communication time)。

第四章　指派的解法

本研究求解 DHC 的排程與指派問題，也是採用逐步求解(serial methodology)的解題架構。亦即先求解指派問題，再求解排序問題。如前所述，指派的決策是一績效評估機制，排序問題是一最佳解的搜尋機制。本研究是採用 Boyer 所提出的 DS 法則[15]當指派的決策，本章擬說明此 DS 法則，以下先說明符號和演算法流程，其次舉例說明此演算法。

4.1 DS 法則指派決策

指標與變數:

任務指標: i　$(1 \le i \le n)$

機器指標: j　$(1 \le j \le m)$

F_j : 就目前已經指派在機器 j 上的任務，預估機器 j 的總完成時間

$F_{ij} = T_j$: 若將新任務 i 指派在機器 j 上，預估機器 j 的總完成時間

$T = \max(F_1,\ldots..,F_m)$ ：指派新任務後，預估程式總完工時間

Δ_{ij} : F_{ij}- F_j，若新任務 i 指派給機器 j，機器 j 預估完成時間的增量

茲將 Boyer 文獻中 DS 法則的演算法敘述於下。此演算法輸入參數包括一給定的任務順序(ordered list of tasks)、計算機器執行各任務的時間(machine list)、任務的 DAG 圖。輸出結果是將每個任務指派給機器(matching decision)，並計算出總完工時間(makespan)。首先初始化 T=0，目前排程解為空集合。依序把任務預估計算每個機器 j 的$\triangle_{ij}$和 T_j，根據是否有無超過目前的 makespan 為判定。大於目前的 makespan，把任務指派給最小 T_j 的機器 y，更新目前的 makespan 和 F_j ；小於等於目前的 makespan，把任務指派給最小$\triangle_{ij}$的機器 x，更新 F_j。直到全部任務指派完停止。流程如下:

Procedure DS

Input: ordered Task_list, Machine_list, DAG

Initialization: $T = 0$, *Solution_schedule* = {empty}

While(Task_list not empty)

Remove task *i* from head of Task_List

For each machine *j* in Machine_List

Compute $\triangle_{ij}$, T_j

$x = \arg\min_{j}(\Delta_{ij})$

$y = \arg\min_{j}(T_j)$

if ($T_y > T$) then ,

$T = T_y$, and $F_y = T_y$

Machine_Assign(i) = *y*

else

$F_x = T_x$

Machine_Assign(i) = *x*

Endfor

Endwhile;

Return Solution_Schedule = {*Assign(i), for each task in Task_List*}

4.2 DS 演算法的釋例

茲以一釋例說明 DS 的演算法。此例的輸入 Task_List, Machine_List 和 DAG 如圖 4.1 所示為 5 個任務 3 個機器所組成的問題。輸入的 Task_List 的 $L=t_1 \rightarrow t_5 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4$，根據 DS 運算之後，如圖左已指派完 t_1、t_5、t_2、t_3，選擇 t_4 指派，預估計算$\triangle_{ij}$最小為 m_2，T_j最小為 m_1，但因 T_j 未超過目前 makespan，選擇指派

給 m_2，計算出此 Task_List 的總完工時間為 6，t_5 指派給機器 1，t_3 與 t_4 指派給機器 2， t_1 與 t_2 指派給機器 3。

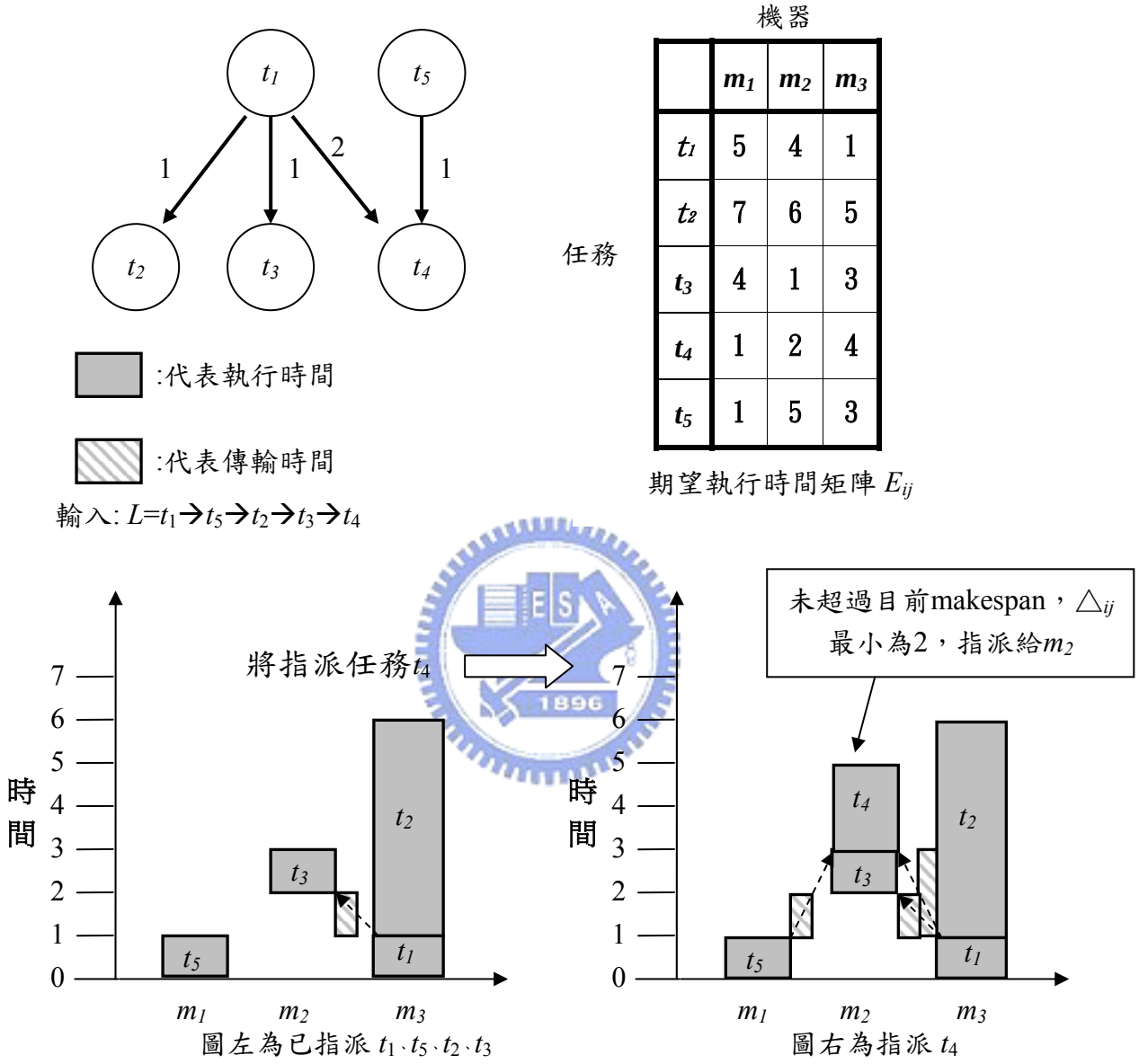


圖 4.1　一個簡單的 DS 例子

第五章　排序的解法

如前所述，本研究求解 DHC 的排程與指派問題，是採用逐步求解(serial methodology)的解題架構。指派的決策是一績效評估機制，排序問題是一最佳解的搜尋機制。本研究提出一以共識因子為基礎的基因演算法為搜尋機制。此基因演算法分為六個模組：染色體表達法、適應函數、初始母體產生法、產生新染色體的共識因子、選擇染色體形成新母體的方法、終止條件。茲將各模組分別敘述如下：

5.1 染色體表達法

每條染色體代表為任務執行順序(task sequencing)，可表示為 $X=\{x_1, x_2,\ldots, x_n\}$，$X$ 代表一條染色體，n 代表所有任務的個數。染色體基因值(gene value)表示為執行的任務 t_i，染色體的索引(index)代表任務 t_i 的先後。而一個有效的染色體不可以違反任務之間的執行順序限制。例如在 DAG 中，6 個任務可能產生如表 5.1 的染色體 $t_1 \rightarrow t_6 \rightarrow t_5 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4$，其中 t_6 在 t_2 之前，因此 t_6 的 index 小於 t_2 的 index。

表 5.1 染色體 $t_1 \rightarrow t_6 \rightarrow t_5 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4$

染色體 chromosome :

索引(index)	1	2	3	4	5	6
基因(gene)	t_1	t_6	t_5	t_2	t_4	t_3

5.2 適應函數

適應函數是用於表達各染色體解的品質，以決定其下一世代的存活率。解的品質越好，染色體存活率越高。產生的染色體經由派工方法 DS 計算出的程式總完工時間 T，作為適應值(fitness value)。最小化目標式，其適應函數如下：

適應函數: $T = \max(F_1,\ldots,F_m)$

5.3 初始母體產生法

因為任務之間有執行順序限制，染色體基因不可以隨機的指派。為了產生好的染色體，我們採用 PSGA 的優點，根據問題特徵去求得 *b-level* 和 *t-level* 等數值，以產生任務的優先值(task priority value)，敘述如下。

(A)計算 *b-level*:

*b-level*為任務t_i到最後一個任務的最長關鍵路徑長度，因此每個任務t_i的*b-level*被DAG的所限制，因此越前面的任務所需的旅行路徑越長，其*b-level*通常較高。每個任務t_i的*b-level*計算為在各機器平均執行時間*Avg*E(t_i)加上與最長(longest)的子任務t_j (child task)的*b-level*和傳輸時間*CommTime*(t_i, t_j)。根據DAG由下往上計算每個任務t_i的*b-level*，若任務t_i沒有子任務t_j其*b-level*為其平均執行時間*Avg*E(t_i)。*b-level*的計算流程如下:

Procedure *b-levels*( ):

Construct a reverse topological order list (*RevTopOrdList*) of tasks.

for each task t_i in the *RevTopOrdList* do begin

max = 0;

for each child task t_j of task t_i do begin

if (*CommTime* (t_i, t_j) + *b-levels*(t_j) > *max*) then

max *CommTime* (t_i, t_j) + *b-levels* (t_j)

endif;

endfor;

b-levels (t_i)= *Avg*E(t_i) + *max*

endfor;

(B)計算 *t-level*:

另外我們還要計算每個任務的 *t-level*，*t-leve*(t_i)為 entry node 到達任務 t_i 的最長距離，因此可以判斷任務 t_i 的最早開始時間。*t-level*(t_i)為父任務(parent task)的 *t-level*(t_k)與父任務的傳輸時間 *CommTime*(t_k, t_i)加上父任務的執行時間 *Avg*E(t_k)。根據 DAG 由上往下計算每個任務 t_i 的 *t-level*，若任務沒有父任務的話，其 *t-level* 為 0。*t-level* 的計算流程如下:

Procedure *t-levels*( ):

Construct a topological order list (*TopOrdList*) of tasks.

for each task t_i in the *TopOrdList* **do begin**

max 0;

for each parent task t_k of task t_i do begin

if *t-level* (t_k) + *Avg*E (t_k) + *CommTime*(t_k, t_i) > max then

max *t-level* (t_k) + *Avg*E(t_k) + *CommTime*(t_k, t_i);

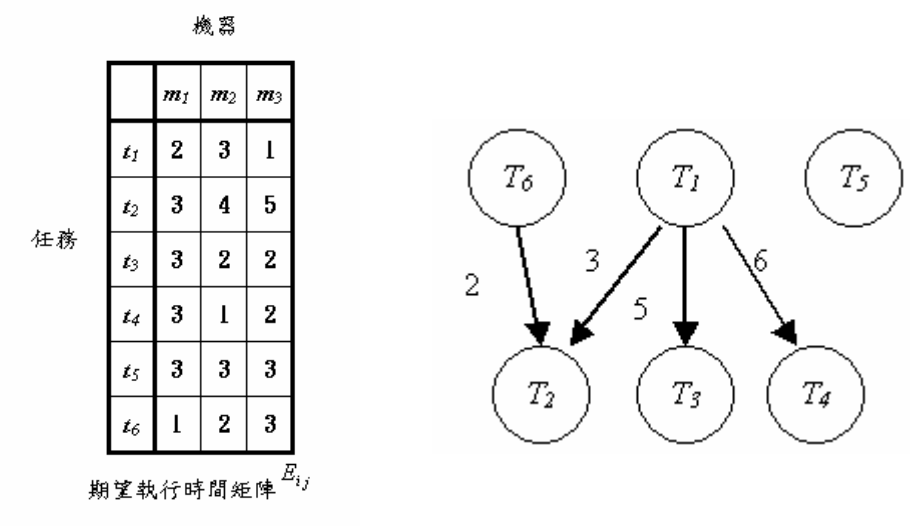
endif;

endfor;

t-level (t_i)= *max;*

endfor;

茲以一釋例如下圖 5.1 所示，計算出如表 5.2 所示各任務的 *AvgE*、*t-level* 和 *b-level* 的數值。



	m_1	m_2	m_3
t_1	2	3	1
t_2	3	4	5
t_3	3	2	2
t_4	3	1	2
t_5	3	3	3
t_6	1	2	3

圖 5.1　　6 個任務的 DAG 與 machine list

表 5.2 各任務的 *AvgE*、*t-level* 和 *b-level*

task	*AvgE*	*b-level*	*t-level*
t_1	2	12	0
t_2	4	4	5
t_3	7/3	7/3	7
t_4	2	2	8
t_5	3	3	0
t_6	2	8	0

(C)計算任務的優先值:

任務優先值 *Task_priority*(t_i)以 *b-level* 當作任務的優先基準，*t-level* 當作變異，每個任務的優先值 *Task_priority* 如下所示:

$$Task\_priority(ti) = b\text{-}level\ (t_i) + uniform\ (\ t\text{-}level(t_i)/2\ ,\ -t\text{-}level(t_i)/2)$$

(D)產生染色體:

任務的優先值 *Task_priority* 越大，代表其優先順序越前面，因此可以根據每個任務的優先值排出任務的執行先後順序(task sequencing)，產生的染色體，而且

根據 *t-level* 變異影響，每次產生的任務優先值皆為不同，因此可以產生不同的染色體，如圖 5.2 所示任務的優先值轉換為染色體。因為經過問題結構去分析過而產生，染色體比隨機產生的染色體，有較佳的求解品質。

任務(task)	t_1	t_2	t_3	t_4	t_5	t_6
任務優先值	13	4	7/3	2	3	8

解碼➔t_1→t_6→t_2→t_5→t_3→t_4

任務 t_i

染色體:

索引(index)	1	2	3	4	5	6
基因(gene)	t_1	t_6	t_2	t_5	t_3	t_4

圖 5.2 由任務的優先值轉換為染色體

(E)初始母體:我們利用 PSGA 的方式產生了 *Ni*:1000 個染色體，根據其適應值，抽取前 N_p:60 條染色體當作我們初始染色體 *P(0)*。

5.4 產生新染色體的共識因子

傳統的基因演算法對於產生新的染色體一般採用的演化技術(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，在這有相依關係的問題中，有時反而會產生不好的解，甚至會有不合理解，交配的過程中並不考慮上一代好的基因，而盲目搜尋(blind search)。我們主要提出這共識因子(consensus operator)，希望可以根據上一代好的染色體中尋找資訊，建立一個導引(guided)機制，根據這引導機制讓好基因順序保留，不好的基因順序則自然淘汰，以產生好的染色體，敘述如下。

(A)共識因子產生方式

共識因子產生方式主要分為三個步驟:

Step 1: 建立一個共識池(consensus pool) S

為了讓取得的資訊不會陷入局部解(local trap),我們從母體 $p(t)$隨機抽取一半的染色體放進我們建立的共識池 S 裡。

Step 2: 建立引導機制:對偶優先矩陣 PWP_matrix(pair-wise priority_matrix)

本研究提出這引導機制:對偶優先矩陣 PWP_matrix; M,初始為一個 0 矩陣(n × n matrix ,n 為任務的個數),從共識池 S 中每條染色體,藉由 PWP_Chromosome 流程去計算任務之間的優先關係矩陣 A,形成新矩陣 M (M=M+A),如圖 5.3 所示。其中 PWP_Chromosome(X_k)流程,X_k 代表輸入的染色體,我們初始一個任務優先關係矩陣 A=[a_{ij}],(1≤ $i, j \leq n$)也為 0 矩陣(n × n matrix)只要染色體 X_i 中,任兩任務 t_i,t_j 任務 t_i 在任務 t_j 之前,a_{ij} 就加 1,流程如下:

Procedure PWP_Chromosome(X):

Initialization: matrix A =[a_{ij}] = **0** (n × n matrix)

For

any two tasks t_i and t_j in chromosome X , $t_i \neq t_j$

If rank(t_i) < rank(t_j) , *then*

$a_{ij} = a_{ij} + 1$

endif

Endfor;

其中 rank(t_i)為染色體的索引(index),也為任務的優先順序。

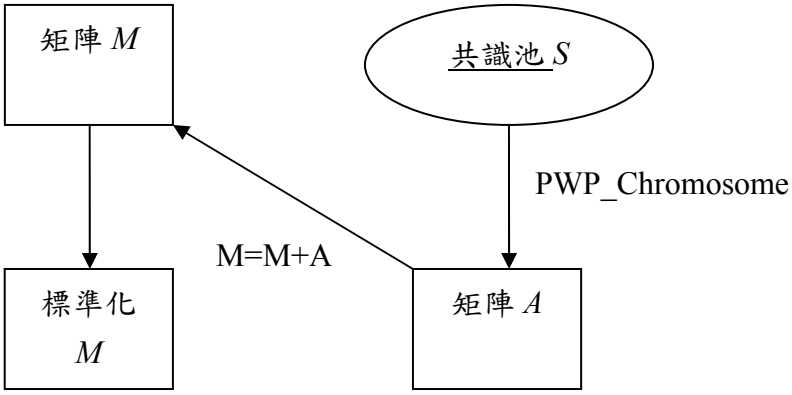


圖 5.3　PWP_matrix:*M* 流程

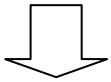
釋例，給定一染色體 *t1*→*t6*→*t5*→*t2*→ *t4*→ *t3*，藉由 PWP_Chromosome 流程，產生如圖 5.4 的任務優先關係矩陣 *A*。

任務	1	2	3	4	5	6
1		1	1	1	1	1
2	0		1	1	0	0
3	0	0		0	0	0
4	0	0	1		0	0
5	0	1	1	1		0
6	0	1	1	1	1	

圖 5.4 任務優先關係矩陣 *A*

我們還需要把對偶優先矩陣 *M* 標準化為 $M = M/\mathrm{N}(S)$，N(*S*)代表共識池 *S* 所有的染色體個數。標準化的對偶優先矩陣 *M* 特性為互補性 $M_{ij} + M_{ji} = 1$，如圖 5.5 所示，$M_{31}+M_{21}=1$，這矩陣提供資訊，說明在 S 集合中，百分 80 的染色體，認為任務 t_3 在任務 t_1 前執行比較好。因此我們可以知道任兩任務之間 t_i 應該在 t_j 之前執行的**共識比率**，因此對偶優先矩陣 *M* 我們也稱為共識矩陣(consensus Matrix)。

任務	1	2	3	4	5	6
1	0	30	6	30	15	21
2	0	0	15	18	3	0
3	24	15	0	21	0	12
4	0	12	9	0	0	6
5	15	27	30	30	0	18
6	9	30	18	24	12	0



標準化:除以 N(*S*)=30

任務	1	2	3	4	5	6
1	0	1	0.2	1	0.5	0.7
2	0	0	0.5	0.6	0.1	0
3	0.8	0.5	0	0.7	0	0.4
4	0	0.4	0.3	0	0	0.2
5	0.5	0.9	1	1	0	0.6
6	0.3	1	0.6	0.8	0.4	0

圖 5.5 模擬 6 個任務 從 PWP_matrix: *M* 標準化後的 *M*

根據 PWP_matrix 的共識建議，我們可以很清楚任務 t_i 在任務 t_j 之前執行的**共識比率**。然而機率 1 或 0 為必然任務 t_i 在任務 t_j 之前執行或之後執行的比例，而在母體 *p(t)*不斷的演化之下，如圖 5.6，我們很清楚可以看出有些 1 或 0 的比例，經過世代的演化，還是沒有影響，有些卻漸漸演化成 1 或 0。那些不會變化的 1 或 0 其實就是問題本身的優先執行順序限制，因此為**硬性限制(hard constraint)**。而那些慢慢演化為 1 或 0，都是經過共識因子不斷演化，產生較好的共識建議，而演化的**柔性限制(soft constraint)**。因此，我們把這兩種都當作限制，不能去違背，之後共識因子產生的方式，在有 1 或 0 的限制下，要先判定沒有 1 或 0 的限制下的合理區塊(feasible region)，在去選取其共識建議。

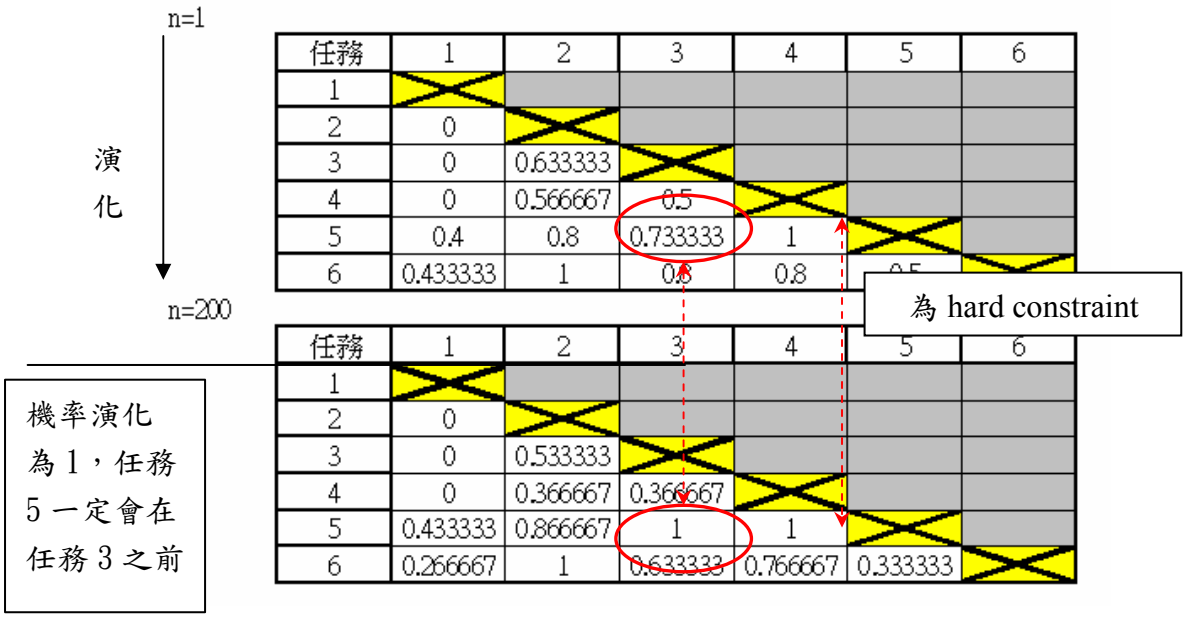


圖 5.6　PWP_matrix 的硬性限制與柔性限制

Step 3: 根據引導機制 PWP_matrix 產生一個新的染色體 Y

首先建立一個空集合 $Q=\phi$，與所有任務的集合 X=$\{t_i;\ 1\le i\le n\}$，Z 為一個很大的數(例如:Z=100)，隨機抽取任務 q_i 從 X 中，因此 X = X-$\{q_i\}$，而根據 PWP_Matrix 去決定 q_i 的優先值 $h(q_i)$ (稱為 PWP 任務優先值)，把已經有 PWP 任務優先值的 q_i 放進 Q 集合中，為 $Q=Q\cup\{qi\}$，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任務 q_i 的 PWP 任務優先值去排序產生，PWP 任務優先值越小任務的順序越前面，如圖 5.7 所示的例子中，排序為 q_2→q_3 →q_1 →q_4。

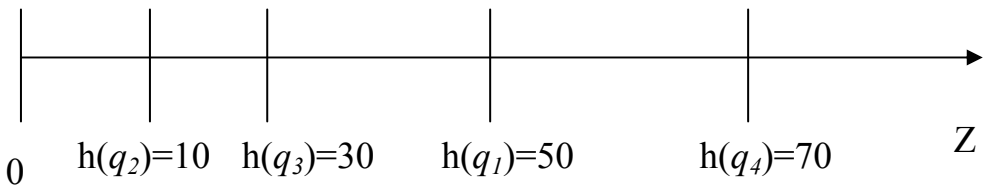


圖 5.7　PWP 任務優先值去排序: $q2$→$q3$ →$q1$ →$q4$

我們已知 PWP_matrix :M 還有在 Q 集合中的所有任務 Q = $\{q_1, \ldots, q_{i-1}\}$和在 Q 中已計算的 PWP 任務優先值 $h(q_k)$，$q_k \in Q$，如何計算任務 q_i 的 PWP 任務優先

值，我們可以分為下列五個步驟:

(1)決定 $h(q_i)$合理區域:

因任務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 $h(q_i)$的合理區域，由之前例子中我們可以在 M 中尋找出強烈限制與柔性限制為 0 or 1，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 q_k，$q_k \in Q$，如果 $m_{ik} = 0$ 和 $h(q_k) > h_s$ 則 $h_s = h(q_k)$，和如果 $m_{ik} = 1$ 和 $h(q_k) < h_e$ 則 $h_e = h(q_k)$，更新後的合理區域，表示新的 q_i 在這區域中都不會有違反優先執行順序的限制。式子如下所示:

For k =1 to i-1, /*for each task in Q*/

If $m_{ik} = 0$ and $h(q_k) > h_s$, then $h_s = h(q_j)$

If $m_{ik} = 1$ and $h(q_k) < h_e$, then $h_e = h(q_j)$

Endfor

(2) 在合理區域中選擇參考任務 q_r:

已知合理區域所有的 Q_h，$Q_h = \{q_k | h_s \le h(q_k) \le h_e, q_k \in Q\}$，為了尋找合理區域中的參考任務(reference task)q_r，我們希望尋找共識比例最大的，因此如下式(1)為 q_i 在 q_k 之後機率最大，和式(2)q_i 為在 q_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任務 q_r，也代表任務 q_i 在任務 q_r 之前的共識比例為 $p = M(q_i, q_r)$

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

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

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

(3) 根據 PWP_matrix，執行伯努力(Bernoulli)試驗:

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

(4) 更新合理區域:

實驗成功: 合理區域更新為 $h_e = h(q_r)$

實驗失敗: 合理區域更新為 $h_s = h(q_r)$

(5) 停止條件:

我們尋找到當合理區域中可插入的任務數 $n(Q_h)$ 少於兩個，便可決定其 $h(q_i)$ 值。因此 $n(Q_h) > 2$ 我們繼續尋找新的參考任務 q_r，回到步驟(2)，直到 $n(Q_h) \leq 2$，則決定 $\boldsymbol{h(q_i) = (h_s + h_e)/2}$。

茲以一釋例說明如何決定 $h(q_i)$，如圖 5.8 所示，在已知 $Q=\{q_1, q_2, q_3, q_4\}$，$Q_h=\{q_k | h_s \leq h(q_k) \leq h_e, q_k \in Q\}$ 下，若要插入任務 q_5，決定 $h(q_5)$。第一步驟:先決定 $h(q_i)$ 合理區域為 $h_s = h(q_1)$，$h_e = h(q_4)$。第二步驟:為了尋找參考任務 q_r，根據(1)(2)(3)式，可以求得 $r=q_2$。第三步驟:根據 q_2 做伯努力(Bernoulli)試驗，如果失敗，$h(q_5) > h(q_2)$。第四步驟:合理區域更新為 $h_s = h(q_r) = h(q_2)$。第五步驟: $n(Q_h) = 2$，停止搜尋，$h(q_5) = (h(q_2) + h(q_4))/2$，$q_5$ 插入在 q_4 和 q_2 之間。

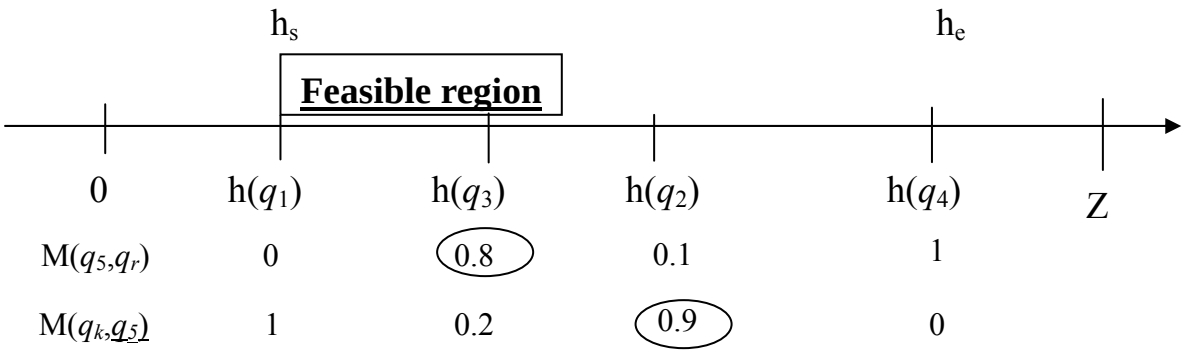


圖 5.8　選擇參考任務 q_r

(B)**調整共識因子影響力**

從實驗發現，由共識因子所產生的染色體，強烈的被 PWP_matrix 所主導，導致所產生的染色體容易陷入局部解當中，為了改進這個缺點，我們希望任務順序可插入的地方可以大幅跳躍，因此我們針對任務 q_i 對於參考任務 q_r 的共識比例 p，乘以一個降級因子 r (degrading factor)，$0 < r < 1$，讓任務 q_i 比較不會照原本抽取的染色體排列順序排列，因此我們在共識因子中設計了兩種方式去產生新的染色體，一種是原本的共識因子 P1=p，產生母體 P(t)* C_1 條染色體;另一種為降級的共識因子 P2=p*r，產生母體 P(t)* C_2 條染色體。

5.5 選擇染色體形成新母體的方法

母體 P(t)之染色體數目 N_p 加上經由共識和降級共識所產生的新染色體數目後，共有 f = P(t) *(1 + C_1 + C_2) 條染色體，此 f 條染色體放在一集合 S_p，選擇策略就是從此集合 S_p 選出 N_p 條的染色體放置於母體 $P(t+1)$中。本文將以 rank-space 演算法。求解，其步驟如下：[21]

步驟一：根據染色體之適應函數值遞減排序，依照排序結果給定一個 Z_i 品質序 (quality-ranking)，表示為 $R_q(Z_i)$。假設排序後結果為 $Z_1, Z_2, \ldots, Z_f$。

步驟二：將品質序第一的染色體從 S_p 移到 $P(t+1)$

$S_p = S_p - \{Z_1\}$；

$P(t+1) \leftarrow \{Z_1\}$；

$Y_1 = Z_1$； /* 將放置在 $P(t+1)$的染色體命名 */

$N = 1$； /*計算放置在 $P(t+1)$的染色體總數 */

步驟三：對 S_p 中的染色體，計算差異指標 $D(Z_i)$ (diversity index)

$$D(Z_i) = \sum_{k=1}^{N} \frac{1}{|Z_i - Y_k|}$$

步驟四：根據 $D(Z_i)$ 對 S_p 中所有剩下的染色體遞增排序，依照排序結果給定 Z_i 一個差異序(diversity-ranking)，表示為 $R_d(Z_i)$。

步驟五：計算 S_p 中所有染色體之品質序和差異序的總合

$T(Z_i) = R_q(Z_i) + R_d(Z_i)$

步驟六：根據 $T(Z_i)$ 對 S_p 中所有染色體遞增排序，排序結果表示為 $R_c(Z_i)$。

步驟七：對 S_p 中所有染色體計算可能存活到 $P(t+1)$的機率

$r = R_c(Z_i)$；

$Prob(Z_i) = p \cdot (1-p)^{r-1}$； /* p 為事先給定的機率，一般訂為 0.667 */

步驟八：產生一隨機值，根據上述的機率分佈從 S_p 中隨機挑選一條染色體至下一世代母體 $P(t+1)$，假設 Z_m 為被挑選之染色體，則移動 Z_m 至下一世代。

$S_p = S_p - \{Z_m\}$；

$P(t+1) \leftarrow \{Z_m\}$；

$Y_m = Z_m$； /* $P(t+1)$的染色體重新命名 */

$N = N+1$； /* 更新 $P(t+1)$的染色體數量 */

步驟九：確認是否選滿 Np 個染色體

if $N < N_p$，則回步驟三

else 停止

5.6 終止條件

基因演算法終止條件有兩種，即當近似最佳解持續維持達 N_c 個世代；或者演化世代 t 達到 N_g 時，則停止演化，並且認定最終近似解為最佳解。因為實際問題最佳解無法求證得知，我們只採用演化世代 t 達到 N_g 時，則停止演化，N_g=100。

5.7 基因演算法流程

如圖 5.9 所示，演算法可以被分為兩個階段。第一階段為一個初始化的階段。對於 DHC 問題輸入 DAG、期望執行時間矩陣 E(machine list)。還有輸入使用者基因演算法參數: N_i 表示用 PSGA 方式產生的染色體個數，N_P 表示在母體 $P(t)$的染色體總數，t 表示所經過之世代，N_g 為演化的世代個數，C_1 為共識因子產生比例，C_2 為降級共識因子產生比例，r 為降級因子。首先利用 PSGA 方式去計算在 DAG 中每個任務的優先值，根據每個任務的優先值去轉換成任務執行順序的染色體，產生個數為 N_i 個，抽取前 N_P 個，作為初始母體 $P(0)$。

第二階段為一個搜尋階段，母體 $P(t)$的染色體經過一些基因運算子的處理，並根據適應值 (fitness value)決定其存活率，再被挑選到下一世代母體 $P(t+1)$。基因運算子主要有三種，複製 (reproduction)、共識(consensus)、降級共識(degrading consensus)，而當符合終止條件 (termination conditions)時，則不再進行世代的演化，取得求解答案。

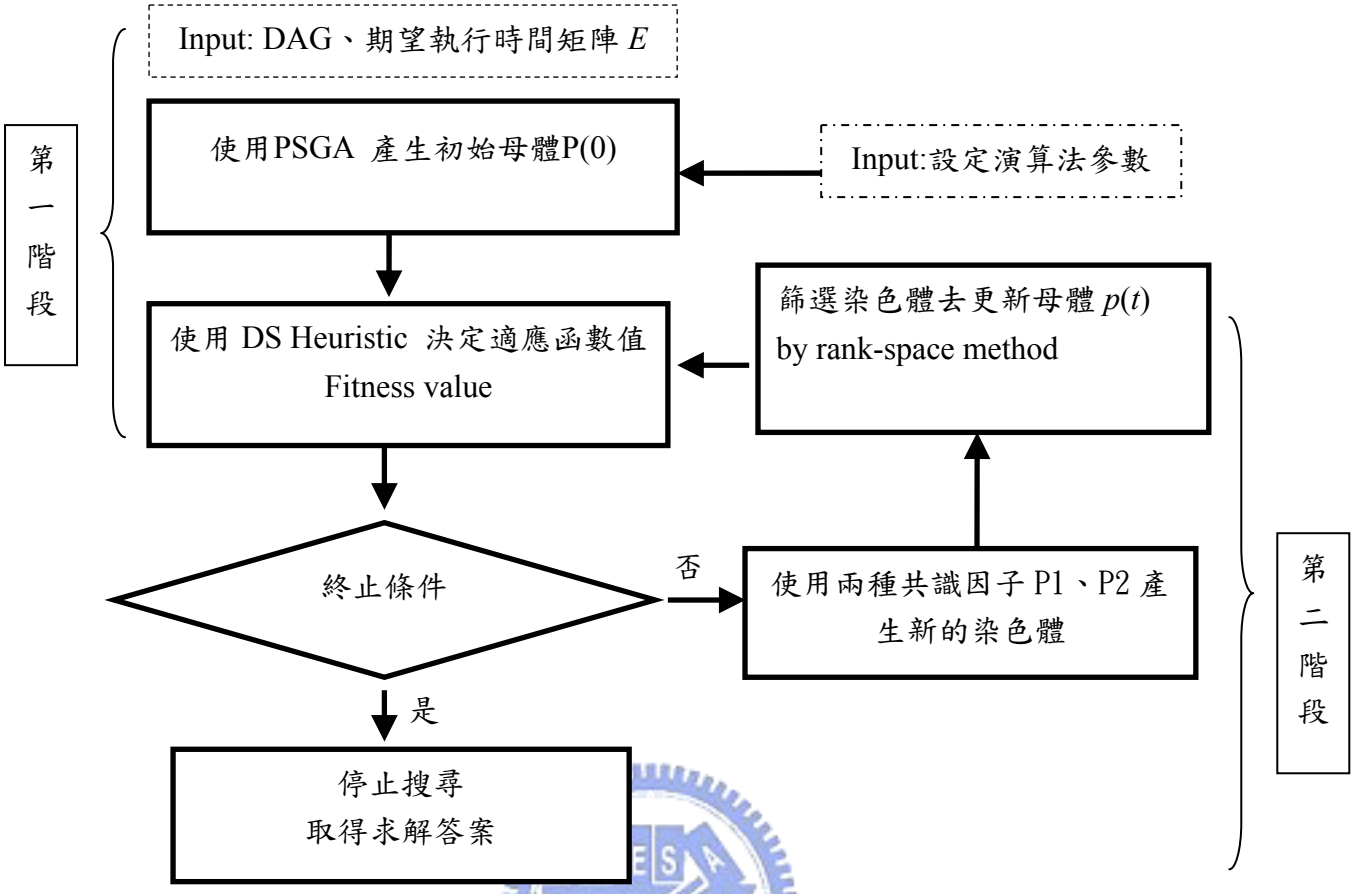


圖 5.9　基因演算法架構圖

第六章　實做與驗證

6.1 模擬情境

在 DHC 問題環境中由於沒有標準的測試案例，我們採用 Boyer[15]的方式，產生 12 種模擬情境根據 100 任務 20 個機器，每個模擬情境根據 DAG 與 machine list 組成。DAG 分為不同相依關係程度；任務的異質度程度、機器的異質度程度和機器是否有一致性(consistent)特性所產生的期望執行時間矩陣 E(machine list)，最後介紹參數設定和模擬平台，描述如下。

(A) DAG:

模擬情境 DAG 的相依關係程度，採用由 Wang[14]的方式，針對每個模擬情境產生連結狀況，T 為應用程式分解任務的總個數，G 為有相依關係，且需要傳輸資料量，設定**高度(High)相依關係**為$(2/3)T < G < T$，**中度(High)相依關係**為$(1/3)T < G < (2/3)T$，**低度(High)相依關係**為 $0 < G < (1/3)T$，與**無**傳輸時間影響，但具有**高度相依關係的狀況**等四種相依程度。傳輸資料量設定為 0 到 1000 之間的隨機變數，且網路之間可同時傳輸。

(B) machine list:

期望執行時間矩陣 E 由隨機數值產生器 (random number generator)產生，矩陣 E 的每個元素(E_{ij})代表任務 i 在機器 j 的期望執行時間，而任務的異質度程度、機器的異質度程度控制著隨機數值產生器產生數值大小。因此，在初始行向量中 B(size T，任務 T 個)用均一分配 (uniform distribution)1~Γ_B (Γ_B:任務的異質度程度)，產生 T 個隨機數值。然而列向量 R(Size M，機器 M 個)用均一分配 1~Γ_R (Γ_R:機器的異質度程度)，產生 M 個隨機數值，E_{ij} 由行向量 i 與列向量 j 乘積產生。本模擬情境設定高度任務異質度為 3000，低度任務異質度為 100，而高度機器異質度為 1000，低度機器異質度為 10。

期望執行時間矩陣可以有一致性 (consistent)的特性，如果設定為有一致性，則運算強的機器，對於任何任務都比運算弱的機器快，因此如果一致的話，列向量需要先排序過。機器 1 對於所有任務都比機器 M 執行速度快。反之，無一致性，運算快的機器對於某任務不一定比運算慢的機器快。

(c)參數設定和模擬平台：

這模擬平台的規格與作業系統如下：

- Microsoft Windows XP
- Pentium 4 3.0 GHz
- 512 MB RAM
- 80 GB HD

本研究把我們提出的演算法和最近幾年提出的演算法 RS、PSGA 做實驗比較，還有改良 RS 方法以單點方式搜尋，採用基因演算法，以多點方式去搜尋，為 RSGA。各實驗方法主要使用 C++語言撰寫，利用 Visual C++ 6.0 編譯完成。我們所提出的演算法的參數如下：

- Consensus rate: C_1=0.3
- degrading Consensus rate: C_2 = 0.3
- degrading factor: r = 0.8
- Initial Set: N_i =1000
- Population: N_p =60
- Generation: N_f =100

PSGA 採用的參數為交配率: 0.6，突變率: 0.05，母體:60，演算世代到 100 世代停止，RSGA 採用跟 PSGA 一樣的參數，RS 為相同的解次數為 15000 次就停止。我們採用的模擬情境如下表 6.1:

表 6.1 測試情境:100 任務 20 個機器 DHC 系統

情境	連結狀況 (dependency)	任務異質度	機器異質度	matrix 一致性
1	高	高	高	否
2	高	高	低	否
3	高	高	高	是
4	高	高	低	是
5	中	高	高	否
6	中	高	低	否
7	中	高	高	是
8	中	高	低	是
9	無	高	高	否
10	無	高	低	否
11	無	高	高	是
12	無	高	低	是

6.2 實驗比較 (RS、RSGA、PSGA、proposed GA)

本研究把我們提出的演算法和最近幾年提出的演算法 RS、RSGA 和 PSGA 做實驗比較，在小 case 的題目中，幾乎所有方法都可以求解到最佳解，因此我們把焦點放在大 case 的題目中，在十二個模擬情境，根據 20 個機器組成的 DHC 和 100 個任務的 DAG，詳細的資訊放在附錄中，其結果見圖 6.1 表、圖 6.1 和圖 6.2 的實驗結果。

由實驗結果圖 6.1 所示，本研究所提出的演算法明顯優於其他演算法。表 6.2 為根據 10 seed，在 95%信心水準與 PSGA 和 RS，有無顯著差異，從結果中，共識

因子為基礎的演算法比其他兩者演算法有顯著差異。

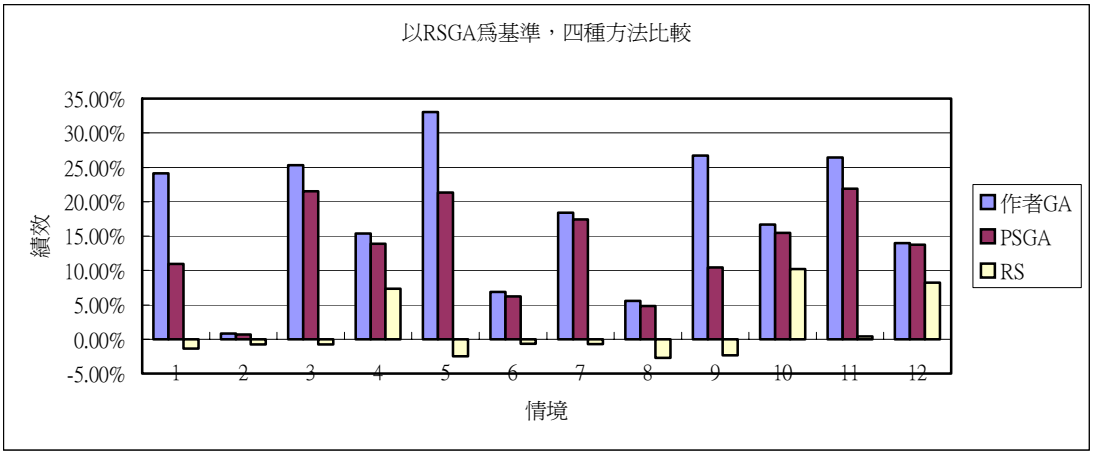


圖 6.1　實驗結果

表 6.2　95%信心水準 proposed GA 平均績效勝過比較方法

情境	*Proposed GA VS. PSGA*	*Proposed GA VS. RS*
1	有顯著	有顯著
2	有顯著	有顯著
3	有顯著	有顯著
4	有顯著	有顯著
5	有顯著	有顯著
6	無	有顯著
7	有顯著	有顯著
8	無	有顯著
9	有顯著	有顯著
10	有顯著	有顯著
11	有顯著	有顯著
12	無	有顯著

圖 6.2 所示，為以共識引子為基礎的演算法比較與 PSGA 的績效改善程度。績效改善率為(makespan of PSGA –makespan of proposed GA)/ makespan of PSGA，由結果得知在各情境當中改善率都大於 0，且在 1、5、9 情境改善率都大於 15%以上。而在長時間演化比較，以 300 世代比較兩方法，而實驗結果也顯現所提出的共識因子演算法比 PSGA 結果較好，詳細結果放在附錄中。

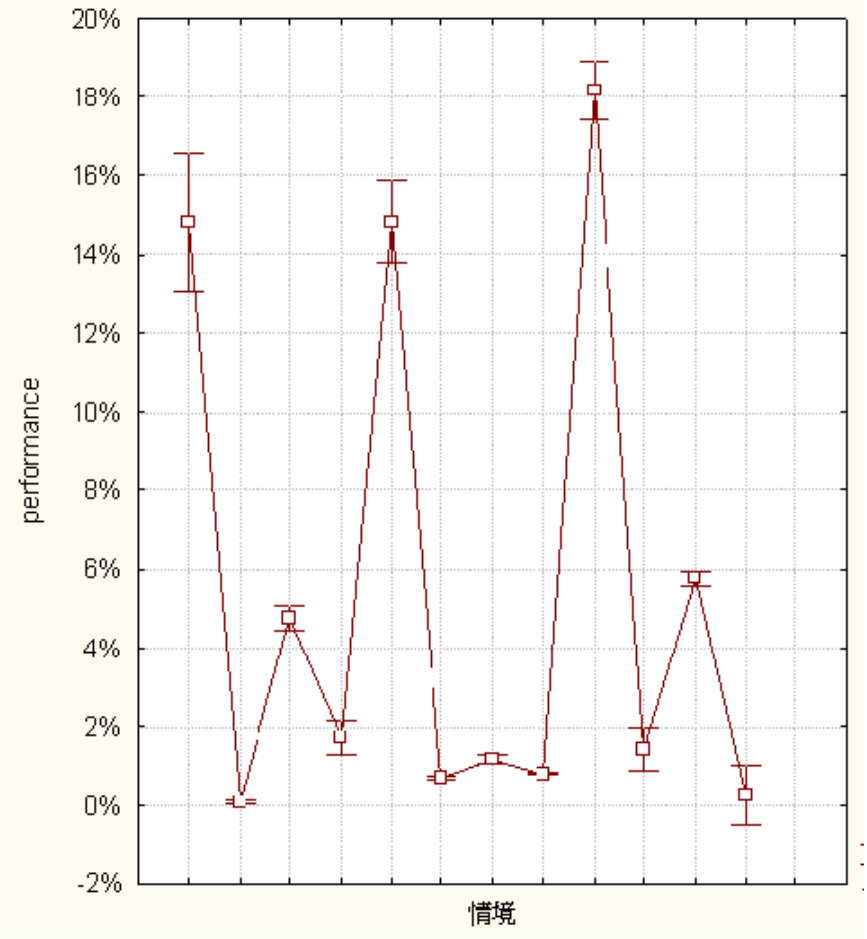


圖 6.2　proposed GA 比 PSGA 績效改善率

很明顯的在方法上，非演算式(隨機產生的方式)還是遠輸於有演化式的方式，因此 RS 和 RSGA 的績效遠遠低於 proposed GA 和 PSGA，且 RSGA 以多點的搜尋方式還是比單點搜尋方式的 RS 有較好的績效，然而時間卻是拉長了許多。而在運算時間上，如表 6.3 所示，基本上 RS 都是最快速的，而 RSGA 為基因搜尋，時間大幅上升，最長的是 PSGA，然而我們提出的方法解比 PSGA 優秀，時間稍微快一點。

表 6.3　各方法在各情境的平均時間

情境	Proposed GA	PSGA	RS	RSGA
1	59.6	63.3	25.1	54
2	59.5	63.3	25.8	53.7
3	60.1	63.1	24.8	53.8
4	60	63.8	26.2	54
5	59.4	63.2	25.3	53.1
6	59.1	63.4	25.2	53.1
7	59.2	63.1	25.4	53.4
8	59	63.2	25.2	53.1
9	59	62.3	24.2	53.2
10	60.1	62.3	24.2	53.9
11	59.4	63.2	24.3	52.9
12	60.7	61.5	24.2	53.7

6.3 交配因子與共識因子比較:

傳統的基因運算子:交配因子與共識因子最主要的差異為共識因子為一種有導引式(guided)的創造，而交配因子則為非導引式(unguided)的創造。交配因子的優點可能會避免陷入局部解，但缺點是可能會創造出許多不好的解，而本研究所提出的共識因子除了根據導引機制創造出較好的解，也為了避免陷入局部解當中，提出降級共識因子(degrading factor)的概念，來達到可以求得全域最佳解(global solution)的目的。

第七章 結論與未來研究

本研究所提出的演算法主要是針對有相依關係的排程問題，改變以往基因演算法，基因運算子在不斷搜尋中，有時是盲目搜尋(blind search)，我們提出一個全新概念的基因運算子:共識因子(consensus operators)根據上一代好的染色體，建立一個導引(guided)機制，把好的基因順序保留，壞的基因順序去除，產生好的染色體，不斷的演化，更新導引機制，加速我們求解品質和收斂的速度。而這提出的演算法在取得近似的最佳解也是非常的有效跟穩健的。

未來工作可以把類似有優先關係任務的指派排程問題，像 APS 先進生產排程，利用此演算法來應用。還有如何加強共識因子的效能，也是另一個研究主題。

參考文獻

[1] T. Kis, “Project scheduling: a review of recent books,” *Operations Research Letters*, vol. 33, pp. 105-110, 2005.

[2] H. Khamooshi, “Dynamic priority–dynamic programming scheduling method $(DP)^2SM$: a dynamic approach to resource constraint project scheduling,” *International Journal of Project Management*, vol. 17, pp. 383-391, 1999.

[3] H. Khamooshi, “$(DP)^2SM$: A dynamic approach to resource constraint project scheduling,” *Computers and Industrial Engineering*, vol. 35, pp. 507-510, 1998.

[4] K. Kim, Y. Yun, J. Yoon, M. Gen, and G. Yamazaki, “Hybrid genetic algorithm with adaptive abilities for resource-constrained multiple project scheduling,” *Computers in Industry*, vol. 56, pp. 143-160, 2005.

[5] C. H. Chen, S. F. Ling and W. Chen, “Project scheduling for collaborative product development using DSM,” *International Journal of Project Management*, vol. 21, pp. 291-299, 2003.

[6] M. Gen, K. W. Kim and G. Yamazaki, “Project scheduling using hybrid genetic algorithm with fuzzy logic controller in SCM environment,” *Journal of Tsinghua Science and Technology*, Vol .8, pp 19-29, 2003.

[7] Y. S. Yun and M. Gen ,”Advanced scheduling problem using constraint programming techniques in SCM environment,” *Computers and Industrial Engineering*, vol. 43, pp. 213-229, 2002.

[8] M. P. Fanti, B. Maione, D. Naso and B. Turchiano, “Genetic multi-criteria approach to flexible line scheduling,” *International Journal of Approximate Reasoning*, vol. 19, pp. 5-21, 1998.

[9] C. Moon and Y. Seo, “Evolutionary algorithm for advanced process planning and scheduling in a multi-plant,” *Computers & Industrial Engineering*, vol. 48, pp. 311-325, 2005.

[10] Y.-K. Kwok and I. Ahmad, “Efficient scheduling of arbitrary task graphic to miltprocessors using a parallel genetic algorithm,” *Journal Parallel Distributed Computing*, pp. 58-77, 1997.

[11] G. C. Sih and E. A. Lee, “A compile-time scheduling heuristic forinterconnection-constrained heterogeneous processor architectures,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, pp.175-186, 1993.

[12] R. Nossal, “An evolutionary approach to multiprocessor scheduling of dependent tasks,” *Future Generation Computer Systems*, vol. 14, pp. 383-392, 1998.

[13] R. Kolisch, “Serial and parallel resource-constrained project scheduling methods revisited: theory and computation,” *European Journal of Operation Research*, vol. 90, pp.320-333, 1996.

[14] L. Wang, H. J. Siegel, V. P. Roychowdhury, A. A. Maciejewski, “Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach,” *Journal Parallel Distributed Computing*, vol 47, pp. 8–22, 1997.

[15] W. F. Boyera and G. S. Hurab, “Non-evolutionary algorithm for scheduling dependent tasks in distributed heterogeneous computing environments,” *Journal Parallel Distributed Computing*, vol. 65, pp.1035 – 1046, 2005.

[16] T. D. Braun, J. H. Siegel, N. Beck, L. Ladislau and M. Maheswaran, “A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems,” *Journal of Parallel and Distributed Computing*, vol. 61, pp. 810-837, 2001.

[17] O. Sinnen and L. Sousa, “List scheduling: extension for contention awareness and evaluation of node priorities for heterogeneous cluster architectures,” Parallel Computing, vol. 30, pp. 81-101, 2004.

[18] G. Grassard and P. Bratley, Fundamentals of algorithms, Prentice-Hall, Englewood Cliffs, NJ, 1996.

[19] M. Gen and R. Cheng, Genetic algorithm and engineering optimization, John Wily and Sons, New York , 2000.

[20]G. Syswerds, “Scheduling optimization using genetic algorithms in Davis, L. Ed.: handbook of genetic algorithms,” Van Nostrand Reinhold, New York, pp.332-349, 1991.

[21] P. H. Winston, Artificial intelligence. Addison-Wesley, USA, ch. 25, pp. 520-527, 1992.

[22] P. Shroff, D. W. Watson, N. S. Flann and R. F. Freund, “Genetic simulated annealing for scheduling data-dependent tasks in heterogeneous environments,” *Proceedings of the Heterogeneous Computing Workshop*, pp.98-104, 1996.

[23] M. Dhodhi, I. Ahmad, A. Yatama, I. Ahmad, “An integrated technique for task matching and scheduling onto distributed heterogeneous computing systems”, *Journal Parallel Distributed Computing*, pp. 1338-1361, 2002.

[24] Y. K. Kwok, I. Ahmad, “Static scheduling algorithms for allocating directed task graphs to multiprocessors,” *ACM Computing Surveys*, pp 406–471, 1999.

[25] H. Topcuoglu, S. Hariri, and M. Y. Wu, ”Performance-effective and low-complexity task scheduling for heterogeneous computing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, pp.260-274, 2002.

[26] Y. Chung and S. Ranka, “Applications and performance analysis of a compile-time optimization approach for list scheduling algorithms on distributed memory multiprocessors”, *Proceedings of the Super-Computing*, pp. 512–521, 1992.

附錄

情境	Proposed GA	PSGA	RS	RSGA
1	**4737759**	**5560996**	**6329567**	**6244343**
2	**456894**	**457380**	**464103**	**460607**
3	**840327**	**882378**	**1133121**	**1124678**
4	**554878**	**564582**	**607429**	**655602**
5	**3117720**	**3660934**	**4770459**	**4655358**
6	**246932**	**248670**	**266922**	**265174**
7	**813192**	**822801**	**1003535**	**996462**
8	**298927**	**301342**	**325276**	**316620**
9	**4329840**	**5291226**	**6045605**	**5908109**
10	**547191**	**555100**	**589892**	**656811**
11	**819795**	**870046**	**1109142**	**1113863**
12	**554069**	**555545**	**590834**	**643994**

各方法在各情境的平均 makespan

情境一	EGA	PSGA	RS	RSGA
1	4.82E+06	5481280	6407310	6304720
2	4.64E+06	5530380	6357430	6202840
3	4.94E+06	5910760	6267400	6339040
4	5.00E+06	5531750	6282190	6336210
5	4.64E+06	5540910	6448320	5799000
6	4.64E+06	5540910	6366550	6368190
7	4.64E+06	5481280	6426680	6407310
8	4.64E+06	5536390	5977860	6150300
9	4.64E+06	5481280	6458880	6392620
10	4.76E+06	5575020	6303050	6143200
平均	**4737759**	**5560996**	**6329567**	**6244343**
平均秒數	59.6	63.3	25.1	54

各方法在情境 1 的實驗結果

情境二	EGA	PSGA	RS	RSGA
1	456457	456967	462852	462587
2	456672	457581	465629	461127
3	457366	457349	464779	461193
4	456672	457455	466666	458954
5	457071	457746	458864	458321
6	457071	457318	463572	460219
7	456457	457286	462474	461378
8	457131	457366	463625	461520
9	457366	457366	464043	462012
10	456672	457362	468527	458759
平均	**456894**	**457380**	**464103**	**460607**
平均秒數	59.5	63.3	25.8	53.7

各方法在情境 2 的實驗結果

情境三	EGA	PSGA	RS	RSGA
1	842218	883335	1.13E+06	1.12E+06
2	841123	882548	1.14E+06	1.12E+06
3	841123	876491	1.14E+06	1.12E+06
4	843721	883397	1.12E+06	1.14E+06
5	831234	887609	1.11E+06	1.13E+06
6	844431	883923	1.13E+06	1.14E+06
7	844622	876654	1.15E+06	1.14E+06
8	837081	885329	1.14E+06	1.12E+06
9	839071	881135	1.12E+06	1.10E+06
10	838647	883362	1.15E+06	1.13E+06
平均	**840327**	**882378**	**1133121**	**1124678**
平均秒數	60.1	63.1	24.8	53.8

各方法在情境 3 的實驗結果

情境四	EGA	PSGA	RS	RSGA
1	552781	572481	604765	664271
2	552107	565633	605062	653085
3	563232	563027	607989	662430
4	554655	565402	613417	674998
5	550894	560114	610557	632065
6	553651	559789	592275	660387
7	554655	565522	600043	643088
8	556120	564440	616208	648119
9	555411	565183	612055	664425
10	555276	564229	611917	653155
平均	**554878**	**564582**	**607429**	**655602**
平均秒數	60	63.8	26.2	54

各方法在情境 4 的實驗結果

情境五	EGA	PSGA	RS	RSGA
1	3092580	3679350	4729980	4586320
2	3170300	3661970	4747020	4589960
3	3163300	3666670	4744850	4788520
4	3111640	3669520	4842600	4746730
5	3169910	3633030	4711470	4556000
6	3056200	3679640	4822370	4629740
7	3195310	3674600	4823180	4681090
8	3052360	3633030	4695870	4648630
9	3072500	3684300	4782050	4586320
10	3093100	3627230	4805200	4740270
平均	**3117720**	**3660934**	**4770459**	**4655358**
平均秒數	59.4	63.2	25.3	53.1

各方法在情境 5 的實驗結果

情境六	EGA	PSGA	RS	RSGA
1	246769	261484	269326	259875
2	246835	247283	273525	263668
3	247009	247244	273240	263543
4	247457	246835	267858	262374
5	246835	247299	256047	274014
6	246835	247009	275984	267687
7	246835	246835	258954	263730
8	246835	247637	261666	264123
9	247070	247299	268479	259730
10	246835	247774	264144	272991
平均	**246932**	**248670**	**266922**	**265174**
平均秒數	59.1	63.4	25.2	53.1

各方法在情境 6 的實驗結果

情境七	EGA	PSGA	RS	RSGA
1	811495	823437	989861	998506
2	812062	823285	1.02E+06	988703
3	814683	821980	1.00E+06	999929
4	814046	822332	1.00E+06	999832
5	813941	822907	1.01E+06	1.00E+06
6	812062	821822	1.00E+06	993697
7	812646	821620	1.00E+06	990389
8	812701	823956	1.00E+06	994753
9	814341	822352	998710	1.01E+06
10	813941	824321	1.01E+06	991258
平均	**813192**	**822801**	**1003535**	**996462**
平均秒數	59.2	63.1	25.4	53.4

各方法在情境 7 的實驗結果

情境八	EGA	PSGA	RS	RSGA
1	298897	298955	317659	314230
2	298897	298897	327574	323982
3	298831	298831	322334	320223
4	298881	300300	328777	314861
5	298897	298955	328792	313629
6	298820	320191	319741	310990
7	299132	299592	326383	316885
8	298728	298786	326661	322178
9	299132	300012	334147	320989
10	299050	298897	320691	308228
平均	**298927**	**301342**	**325276**	**316620**
平均秒數	59	63.2	25.2	53.1

各方法在情境 8 的實驗結果

情境九	EGA	PSGA	RS	RSGA
1	4.30E+06	5.09E+06	6.06E+06	5.81E+06
2	4.41E+06	5.58E+06	6.09E+06	5.63E+06
3	4.30E+06	5.47E+06	6.15E+06	6.01E+06
4	4.30E+06	5.58E+06	6.08E+06	6.09E+06
5	4.30E+06	5.18E+06	6.07E+06	5.92E+06
6	4.30E+06	5.54E+06	5.83E+06	5.99E+06
7	4.46E+06	4.93E+06	5.93E+06	5.98E+06
8	4.30E+06	5.17E+06	6.04E+06	5.84E+06
9	4.30E+06	4.93E+06	6.06E+06	6.07E+06
10	4.30E+06	5.45E+06	6.15E+06	5.76E+06
平均	**4329840**	**5291226**	**6045605**	**5908109**
平均秒數	59	62.3	24.2	53.2

各方法在情境 9 的實驗結果

情境十	EGA	PSGA	RS	RSGA
1	544000	552738	578520	661434
2	554328	560778	586060	652027
3	546805	551969	585606	658795
4	551616	556208	600297	664793
5	539668	553775	592298	653217
6	546812	565351	569960	650193
7	544695	556121	598934	664542
8	546430	550763	602018	642910
9	550408	557753	590675	660690
10	547152	545540	594551	659507
平均	**547191**	**555100**	**589892**	**656811**
平均秒數	60.1	62.3	24.2	53.9

各方法在情境 10 的實驗結果

情境十一	EGA	PSGA	RS	RSGA
1	819728	870058	1.12E+06	1.12E+06
2	820566	869746	1.12E+06	1.10E+06
3	821413	869618	1.11E+06	1.11E+06
4	821912	868048	1.08E+06	1.12E+06
5	820955	872018	1.12E+06	1.10E+06
6	821413	868362	1.08E+06	1.10E+06
7	815962	871506	1.12E+06	1.11E+06
8	815941	869458	1.12E+06	1.12E+06
9	818650	870884	1.10E+06	1.12E+06
10	821413	870758	1.11E+06	1.12E+06
平均	**819795**	**870046**	**1109142**	**1113863**
平均秒數	59.4	63.2	24.3	52.9

各方法在情境 11 的實驗結果

情境十二	EGA	PSGA	RS	RSGA
1	541907	555149	598050	672524
2	553850	556115	607808	603283
3	554003	543108	584426	643687
4	564533	553207	575080	649317
5	554319	558987	575953	623352
6	554543	563771	588983	623352
7	551139	552138	594977	661590
8	553633	553119	599309	659094
9	554015	561917	592625	656709
10	558745	557937	591133	647035
平均	**554069**	**555545**	**590834**	**643994**
平均秒數	60.7	61.5	24.2	53.7

各方法在情境 12 的實驗結果

情境	作者 GA	PSGA
1	**4737759**	**5560996**
2	**456642**	**457380**
3	**837671**	**882378**
4	**553506**	**564582**
5	**3153484**	**3660934**
6	**246906**	**248670**
7	**813684**	**822801**
8	**298834**	**301492**
9	**4352176**	**5013588**
10	**544316**	**555100**
11	**818720**	**870046**
12	**544773**	**555545**

演化至 300 世代，所提出 GA 平均 makespan 低於 PSGA