東　　海　　大　　學

工業工程與經營資訊學系

碩士論文

以平行計算求解彈性製程之零工式生產排程最佳化

研 究 生：廖康硯

指導教授：黃欽印 教授

陳武林 教授

中 華 民 國 一 〇 二 年 一 月

Job-shop scheduling optimization with routing and process plan flexibility by parallel computing

By
Kang-Yen Liao

Advisor：Prof. Chin-Yin Huang
Prof. Wu-Lin Chen

A Thesis
Submitted to the Institute of Industrial Engineering and Enterprise Information at Tunghai University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in
Industrial Engineering and Enterprise Information

January 2013

Taichung , Taiwan , Republic of China

以平行計算求解彈性製程之零工式生產排程最佳化

學生：廖康硯　　指導教授：黃欽印 教授

陳武林 教授

東海大學工業工程與經營資訊學系

摘　　要

現今對於求解 NP-hard 排程問題的研究已有許多傑出的成果。但是過去較多學者提出啟發式演算法，而透過數理規劃求解排程最佳化的研究較少。因此本研究針對較貼近生產現場情況的彈性製程之零工式生產排程問題，引用 Ozguven 等人提出的數學規劃模型（FJSP-PPFs），進行求解。在規劃 NP-hard 問題時，時常要花費許多計算時間，本研究利用平行計算技術，透過資料分割，將建構的數學模型分割成許多組合，搭配加速求解機制進行平行計算，提升求解效率。

本研究分成三個步驟：第一步驟，建立小規模的數學模型驗證排程規劃結果為最佳解，因為小規模最佳解很容易獲得；接著第二步驟，建立中規模的數學模型驗證規劃結果與 Ozguven 等人提出的研究求解一致；最後第三步驟，建立大規模的數學模型，透過平行計算，分析規劃結果與平行計算的效率。實驗結果顯示，中規模排程問題透過 32 個處理器平行計算搭配加速求解機制，可以減少 90%的規劃時間；大規模排程問題透過 64 個處理器平行計算搭配加速求解機制，可以減少 70%的規劃時間。

關鍵字詞：彈性製程、零工式生產、排程問題、平行計算、加速求解機制

Job-shop scheduling optimization with routing and process plan flexibility by parallel computing

Student：Kang-Yen Liao　　Advisors：Prof. Chin-Yin Huang
Prof. Wu-Lin Chen

Department of Industrial Engineering and Enterprise Information
Tunghai University

ABSTRACT

Today for solving NP-hard scheduling problem, there are many outstanding achievements. In the past, many researchers mainly focused on proposing different heuristic algorithms, but studies of flexible job-shop scheduling based on mathematical planning were rarely addressed. This paper develops mathematical programming model according to Ozguven et al.'s work in 2009 (FJSP-PPFs) for dealing with job-shop scheduling problems that resemble-real-world production problems. Because of consuming much computing time when solving NP-hard problems, this paper solves problems by parallel computing. It divides the mathematical model into a lot of combinations (small problems), and then the divided combinations are assigned to the parallel computers to increase the efficiency of solving the problem.

The study is carried out in three steps. In the first step, the small scale mathematical model is developed to illustrate the accuracy of the schedule planning results as its optimal solution is easily obtained. In the second step, the medium scale mathematical model is proposed to illustrate our schedule planning results are the same as those of FJSP-PPFs developed by Ozguven et al. In the third step, the large-scale mathematical model is developed to analyze the planning results and the efficiency of parallel computing. The experimental results show that it decreases 90% of planning time in the medium scale problem by parallel computing with 32 processors associated with the accelerated solving mechanism. It decreases 70% of planning time in the large scale problem by parallel computing with 64 processors combined with the accelerated solving mechanism.

Keywords：Flexible Job-shop Scheduling, Job-shop Scheduling, Scheduling Problem, Parallel Computing, Accelerated Solving Mechanisms

誌　謝

轉眼間研究所的日子將劃上句點。感謝這段時間，黃欽印老師與陳武林老師的指導，不只於課業、研究上，更多的是學習的態度的身教。我想這些寶貴的學習機會是非常難忘的。這篇論文是結合老師們細心的指導與學長姐們所留下的知識所完成的，在此感謝所有幫助我，使我成長的大家。

感謝黃欽印老師與陳武林老師每週會議指導我們，兩位老師細心的指導我們，讓我在學習上收穫良多。感謝系上多位老師與玠孝學長、詩彥學長、容慈學姐的指導與協助，還有貞翔、芸甄、思翰等研究室夥伴，在我課業和研究遇到困難時，給我勉勵，還要感謝研究室的學弟妹，伯棟、振鈺、乃菱和銘哲的幫忙與支持，感謝你們的付出。最後特別要感謝我的家人與其他好朋友們，在我迷惘的時候，指引我方向；在我與挫折的時候，給予我鼓勵。

謹將這份成果獻給每一位幫助過我的貴人，有你們支持及鼓勵，才能完成此論文。

廖康硯　謹誌於

東海大學工業工程與經營資訊研究所

虛擬企業與資料探勘研究室

中華民國一〇二年一月

目錄

表目錄

圖目錄

第一章 緒論

本章分成 4 小節，依序說明研究背景、研究動機與研究目的，並於第 1.4 節提出本研究之研究方法與研究架構。

1.1 研究背景

現今階段全球化市場的競爭下，企業所需考量的因素越來越多。如何運用有限的資源包括人、機器設備、材料、資金和方法等，提升獲利，維持企業競爭力，是企業所面臨的重要課題。

排程是一可以被廣泛定義為隨時間推移，對資源分配進行規劃的任務。普遍的情況下，生產排程中，待分配、規劃使用的資源為工廠內的生產設備；待規劃的任務為訂單；規劃成果的衡量依據，為訂單完成所需要的時間長短。

拜科技發展所賜，現今工廠內許多生產設備不再僅侷限於特定生產流程，而能提供更多樣化的功能，完成指派之任務。相對的，多樣化的資源可供使用選擇的情況之下，其排程規劃變得更複雜，要能保持高效率的分配使用並不容易。規劃不佳所需負擔的成本（如臨時加班、外包）或風險，也可能因排程規劃複雜化而增加。近年來，儘管有些企業已傾向導入企業資源規劃和建構供應鏈管理來整合企業內資源的使用和供應端、顧客端的資訊，但在面對生產現場，仍缺乏有效控管，不外乎是因為生產現場環境具有許多不確定因素，包含設備故障、緊急插單等。這些因素與複雜的排程規劃緊緊相扣。

企業的訂單允諾時間取決於其生產排程的規劃效率，間接影響企業的接單能力，因此生產排程的規劃效率在企業中是不容忽視的一環。總結以上，企業進行排程規劃時，需要考量以下幾點：

1. 訂單數量
2. 生產順序
3. 資源的使用
4. 面對突發狀況，因應的替代製程方案與替代資源存在與否

最為常見的零工式生產排程問題裡，包含彼此獨立的訂單（Job），每項訂單內包含在有排程順序的一組製程方案（Process plan）。製程方案內的每項作業（Operation）各有對應需求的生產資源（Machine）。

由於完成每組製程方案所需要的時間不同，同一作業也會因為使用的生產資源不同，需求時間有所差異。在不違反事先定義（predefined）的訂單內製程方案之排程順序和生產排程沒有作業重疊（Non-Overlapping）等條件底下，透過排程規劃，適當安排每項訂單，使訂單完成時間最小化，是本研究考量之重點。

1.2 研究動機

對於 NP-hard 生產排程問題，已有許多學者提出不同的求解方法，過去較多學者提出啟發式演算法求解；針對零工式彈性製程之生產排程問題，透過數學規劃藉由平行計算，求得最佳規劃結果的研究較為稀少。

求解排程問題的方法中，較常見的有啟發式演算法、模擬方法和透過數學規劃求解。這三種方法皆有其特色：

1. 啟發式演算法是透過邏輯判斷、經驗基礎應用於求解問題、學習、尋找，藉由多次演算求解的方法。啟發式演算法可證明其求解效率良好且可以找到不錯的解，但不能保證是最佳解。也不能保證每次求解所需的時間長短都相同。常見的啟發式演算法有最基本的試誤法（Try and Error），以及模擬退火法（Simulated Annealing）、螞蟻演算法（Ant Colony Optimization）、禁忌搜尋（Tabu Search）、基因演算法（Genetic Algorithm）等。
2. 模擬方法則是透過參數的設定建立系統模型，模擬現實中的情境進一步求解，模擬的方式所提供的解不一定是全域最佳解。
3. 透過數學規劃求解，常見的方法有整數規劃、混整數規劃、線性規劃。透過針對問題建立數學模型進行求解，優點在於能保證求出最佳解，但是會隨著問題複雜化或是問題規模龐大，需花費龐大的時間求解。

本文針對零工式生產排程問題（Job shop problem），考量訂單特性、排程執行順序、生產機台（資源）的彈性選擇，且為因應突發狀況，建構替代製程方案，建立生產排程之數學模型，透過平行計算與加速求解機制，求解最佳規劃結果。

1.3 研究目的

誠如研究背景與動機所述，對於排程問題，較多學者提出啟發式演算法求解。零工式彈性製程之生產排程問題，透過平行計算，求得最佳規劃結果的研究較為稀少。本研究將透過建立數學模型求解。本研究目的如下：

1. 針對零工式生產排程，考量訂單特性（訂單內所含作業量與執行先後順序）、生產機台（資源）的彈性選擇、替代製程方案等建構生產排程之數學模型。
2. 透過數理規劃軟體與分散式平行計算技術，求解排程規劃結果。
3. 藉由資料分割及加速求解機制，提升規劃效率，降低求解時間。

1.4 研究方法與架構

為達上述目的，本研究根據圖 1.1 研究架構的步驟進行之。

1. 從過去文獻探討各種生產排程的特性，了解在規劃時需考量的限制，運用何種方法求解與其求解的成效。
2. 從文獻了解平行運算的背景與運作的模式。
3. 參考過去文獻，建立針對求解彈性製程之零工式生產排程的數學模型，並說明符號、限制式等，透過數理規劃工具與平行計算技術，將數學模型進行運算。
4. 進行實驗與分析，驗證數學模型的可行性，並比較採用平行計算與為採用平行計算所得到的規劃結果與運算時間。
5. 藉由實驗分析結果，提出結論與建議。

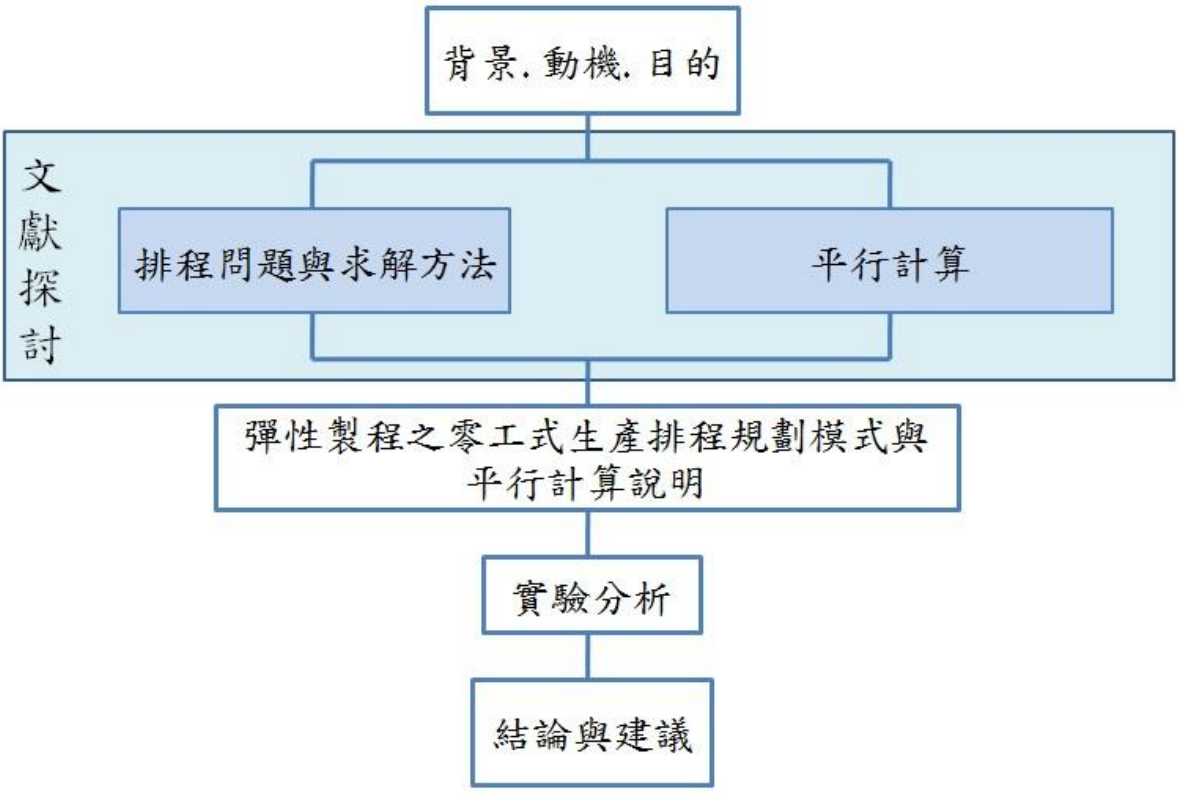
背景.動機.目的
文獻探討
排程問題與求解方法
平行計算
彈性製程之零工式生產排程規劃模式與平行計算說明
實驗分析
結論與建議

圖 1.1 研究架構

第二章 文獻探討

本章探討過去文獻，整理成排程問題與求解方法兩部份分別說明。2.1 節探討過去文獻對於排程問題根據環境、流程特性等所作的分類，並針對本研究所探討的零工式生產於 2.1.1 小節進行說明。求解方法的部分，於 2.2 節，整理過去學者對於排程問題所提出的求解方法；於 2.3 節，說明本研究於求解排程問題使用的平行計算技術。

2.1 排程問題

排程是對於在特定時間完成一組工作，所做的資源配置 Tavakkoli-Moghaddam 與 Daneshmand-Mehr（2005）。排程問題可以被數種方法求解，如分支定界法（Branch and Bound）、切面法（Cutting Plane）、啟發式演算法（Heuristic），與最近幾年較為常見的基因演算法（Genetic Algorithm）、模擬退火法（Simulated Annealing）、機器學習（Machine Learning）等方法。Lee、Lei 與 Pinedo（1997）對於排程問題進行全面性探討。

Graham、Lawler、Lenstra 與 Kan（1979）以及 Pinedo（2012）指出，排程問題可以由 $\alpha/\beta/\gamma$ 三個欄位的符號來表示，α 欄位表示排程的機器環境；β 表示製程的加工特性（Processing characteristics）與限制；γ 表示排程目標。就 α 欄位而言，排程的機器環境有以下幾種：

1. 單一機器（Single Machine）：一整組的任務或工作全指派到一部機器或使用同一個資源。
2. 平行機器（Parallel Machine）：傳統平行機器問題是將 n 件工作規劃安排至 m 部平行機器。常見的平行機器可以分為等效平行機器（Identical parallel machine）與無關聯平行機器（Unrelated parallel machine）兩種。所有的等效平行機器對某一件工作的處理時間皆相同，相反的，無關聯平行機器則是各部機器加工速度皆不同，而且對於同一部機器，不同的工作所需時間也會不相同。
3. 流程式生產（Flow Shop）：流程式生產內的每件工作皆有固定的製造流程，且流程皆相同。

4. 零工式生產（Job Shop）：零工式生產與流程式生產不同，生產過程中，每件工作都有其固定製造流程，但不同的工作之製造流程並不相同。
5. 開放式生產（Open Shop）：在開放式生產裡，每件工作無一定的製造流程。

排程績效有許多衡量指標，陳建良（1995）將績效衡量準則彙整、分類如表 2.1 績效衡量準則，其中表中的衡量指標解釋如下：

1. 總完工時間（Total Completion Time）

工件完成所有加工處理的時間稱為完工時間（Ci）。而所有工件完工時間的總和，則稱為總完工時間，以 Ci 表示。

2. 平均流程時間（Mean Flow Time）

工件從開始時間（Si）到完工時間（Ci），稱為流程時間，以 Fi = Ci － Si 表示。而所有工件流程時間的平均值，則稱為平均流程時間（F）。

3. 平均差異時間（Mean Lateness）

工件在完工時間（Ci）與交期日（di）的差稱為差異時間（Li）。而所有工件差異時間的平均值，則稱為平均差異時間（L）。其中 Li = Ci－di 為表示，若 Li＞0，表示完工時間大於交期日；若 Li＝0，表示完工時間等於交期日；若 Li＜0，表示完工時間小於交期日。

4. 平均早交時間（Mean Earliness）

完工時間提前在交期日前就完成的程度稱為早交時間（Ei），以 Ei = Max（0,di－Ci）表示。而所有工件早交時間的平均值，則稱為平均早交時間（E）。

5. 平均延遲時間（Mean Tardiness）

完工時間超過交期日的程度稱為延遲時間（Ti），以 Ti = Max（0,Ci － di）表示。而所有工件延遲時間的平均值，則稱為平均延遲時間（T）。

表 2.1 績效衡量準則 陳建良（1995）

分類	衡量標準
完工時間相關	平均流程時間(Mean flow time) 最大流程時間(Maximum flow time) 流程時間的變異數(Variance of flow time) 總完工時間(Total completion time) 總加權完工時間(Total weighted completion time) 總時程(Makespan) 生產週期(Cycle time) 單位時間產量(Through out)
交期相關	平均差異時間(Mean lateness) 平均早交時間(Mean earliness) 平均延遲時間(Mean tardiness) 最大延遲時間(Maximum tardiness) 總延遲時間(Total tardiness) 總加權延遲時間(Total weighted tardiness) 延遲工件數(Number of tardy jobs) 延遲工件比率(Proportion of tardy jobs)
機器及暫存區相關	平均機器使用率(Average utilization) 平均準備時間(Average setup time) 平均在製品數量(Average work in process) 平均等候時間(Average wait time) 最大等候時間(Maximum wait time)
物料搬運相關	平均物料搬運系統使用率 物料搬運系統空車時間

2.1.1 零工式生產

在排程問題領域中，零工式生產排程（Job-shop Scheduling Problem）是最難的問題之一。常見的零工式生產排程問題裡，一組工作需在一組機器上被處理，其中每一件工作由一些具有次序的作業所組成。每一項作業需由特定的機器完成，每一台機器在一時間內只能進行一項作業且不能被中斷停止。零工式生產不同於流線式生產（Flow Shop）之處在於每件工作都有其固定的流程，不同的工作，流程也會不同。Pezzella、Morganti 與 Ciaschetti（2008）提出，典型的零工式生產排程問題（JSP）績效衡量的指

標為完成所有工作所需要花費的時間，或稱為完工時間（Makespan）。Brucker（2007）說明零工式生產排程問題，早在 1963 年就已有學者透過分支定界法求解。當時的例子儘管只是十件工作與十台機器，但已經是一 NP-hard 問題。

Hertz、Widmer（1996）與王治平（2003）指出，零工式生產排程（Job shop scheduling）問題有以下幾點特性：

1. 在任何一個時間點，對於同一件工作，只能有一項作業被處理。
2. 作業處理的過程不允許被中斷（preemption），也就是每項作業在開始執行後，直到完成前不能被中斷。
3. 每一部機器在同一時間內，只能處理一項作業。
4. 每件工作的流程是唯一的，沒有分歧與選擇。
5. 允許 re-circulation，即一件工作內的兩項作業可以使用同一部機器。
6. 每項作業所需工時（duration）都是已知的且不變的正整數。
7. 每項作業對於機器的需求量都是已知的且固定不變。
8. 不同工作的作業，彼此之間順序沒有優先的限制。

零工式生產排程是假設每一項作業只能由一部機器完成，每一件工作只包含一種製程方案。Beck 與 Fox（2000）認為貼近實際的生產問題需要有更廣泛的選擇空間，工作有多種製程方案可以選擇執行，作業有多部機器可以選擇使用。針對此問題，Ozguven、Ozbakir 與 Yavuz（2010）對零工式生產排程提出多製程方案與多機器選擇兩個方向的延伸，引用 Manne（1960）的零工式生產排程模型，針對解單一製程之彈性生產排程問題（Flexible job-shop scheduling problems）發展出數學模型（MILP-1）與解彈性製程之零工式生產排程（FJSP with process plan flexibility）的數學模型（MILP-2）。

2.2 排程問題求解方法

如 2.1 節所述，現今有許多求解排程問題的方法，包含分支定界法（Branch and Bound）、切面法（Cutting Plane）、啟發式演算法（Heuristic）、基因演算法（Genetic Algorithm）、模擬退火法（Simulated Annealing）、螞蟻演算法（Ant Colony Optimization）、禁忌搜尋（Tabu Search）、機器學習

（Machine Learning）等。透過不同的方法求解排程問題接有其特色與適合的問題類型。底下對常見的啟發式演算法與過去較少學者提出透過數學規劃求解等兩類方法做介紹。

2.2.1 啟發式演算法

啟發式演算法是透過邏輯判斷、經驗基礎應用於求解問題、學習、尋找，藉由多次演算求解的方法。啟發式演算法可證明其求解效率良好且可以找到不錯的解，但不能保證是最佳解。也不能保證每次求解所需的時間長短都相同。其中模擬退火法、螞蟻演算法、禁忌搜尋、基因演算法皆屬於啟發式演算法。過去不乏許多學者提出啟發式演算法求解排程問題。

DauzerePeres、Paulli（1997）與 Schrich、Armentano 與 Laguna（2004）先後探討透過禁忌搜尋求解排程問題，除此之外，Mazdeh（2009）考量機器設備折舊（Deteroration）與機器設備的作業特性等因子，提出最小化總延誤（Total Tardiness）、最小化機器折舊成本之雙目標數學模型，透過禁忌搜尋，求解平行機器排程問題。Ozguven、Ozbakir 與 Yavuz（2010）針對具有彈性製程的零工式生產排程問題（Flexible Job Shop Problem，FJSP），透過基因演算法求解。Steinhofel、Albrecht 與 Wong（1999）對於零工式生產排程，藉由離散圖形呈現排程問題，並採用兩種以模擬退火法為基礎的啟發式演算法，求解問題之最小總完工時間。Loukil、Teghem 與 Fortemps（2007）則是針對批量生產的彈性零工式生產問題，考量最大完工時間、平均延遲時間、平均完工時間等多目標，發展多目標啟發式演算法。Neto 與 Godinho（2011）透過螞蟻演算法，求解流程式生產的排程問題。

2.2.2 透過數學規劃求解

在最佳解方法中，透過數學規劃求解，包含整數規劃、混整數規劃、動態規劃等方法。透過數學規劃求解所需花費的時間與成本可能隨著問題的複雜度，呈指數成長。但因為此方法求解為最佳解，透過整數規劃求解複雜、龐大的排程問題或面對現實情境所需，更能展現其價值。Ozguven、Ozbakir 與 Yavuz（2010）對於彈性零工式生產排程問題，提出混合整數線性規劃模型（Mixed Integer Linear Programming），透過數理軟體求解，除此之外，更進一步發展出可以處理彈性製程方案（Process plan flexibility）

排程問題的數學模型。在彈性製程方案的零工式生產排程問題（FJSP-PPF）中，每件工作（Job）包含兩種製程（Process plan），在開始處理每件工作前，可以由兩種製程方案之中，挑選一種執行。相當於每件工作包含兩種製程方案，每種製程方案包含具有執行先後次序多項作業（Operation），每項作業有對應處理的機器（Machine），其中，每項作業有兩台機器，擇一進行處理。

除了上述求解方法，Mouret、Grossmann 與 Pestiaux（2011）將時間表示（time representation），如時間點、時間區間、事件點應用在界定工作的處理的優先順序，在此基礎上，透過最大化完全子集（Cliques）與雙完全子集（Bicliques），建構出一避免作業重疊的圖型架構（Non-overlapping graph），並進一步發展求解方法，求解全域最佳解或得到近似最佳解，運用在如石油渠道等單階、多階的批量排程問題。

綜合上述整理成表 2.2 求解方法整理。

表 2.2 求解方法整理

作者	排程環境	求解方法	排程目標
Dauzere Peres et al.（1997）	零工式生產	禁忌搜尋法	最小化最大完工時間
Steinhofel et al.（1998）	零工式生產	模擬退火法	最小化總完工時間
Schrich, CR et al.（2004）	彈性零工式生產	禁忌搜尋法	最小化總延誤
Pezzella et al.（2007）	零工式生產	基因演算法	最小化最大完工時間
(Loukil, Teghem, & Fortemps, 2007)	彈性零工式生產	模擬退火法	考量多目標
Mazdeh（2009）	平行機器	禁忌搜尋法	最小化總延誤
Ozguven et al.（2009）	零工式生產	數學規劃	最小化最大完工時間
Mouret et al.（2010）	批量排程	時間表示法	最小總提早時間
(Tavares Neto & Godinho Filho, 2011)	流程式生產	螞蟻演算法	考量多目標

2.3 平行計算

過去單一處理器所能提供的計算效能，已無法滿足現今處理工程科學上複雜問題的需求。因此叢集電腦的概念因此而發展出來，工程師透過建立具備多處理器的單一主機或是將多部主機以高速網路設備連結成一平行的叢集電腦平台。叢集電腦藉由中介軟體，連結多部電腦，發揮高效能、

高可靠度與簡易操作等三大特性。並透過平行化技術，得到單一處理器無法提供的計算性能。Java RMI 是其中一種提供平行計算的平台，本節底下將對此平行計算平台作介紹。

2.3.1平行計算架構

在早期 Flynn（1972）提出計算機系統結構的分類法，Kai Hwang（1984）也提出平行處理計算機的論述，Bell（1992）亦介紹了多處理器的分類法，使得平行計算和處理開始被廣泛討論。而在平行處理的趨勢發展以及 Message-Passing System 的成功下，MPI（Message Passing Interface）會議 1992 年召開，目的是制定標準的訊息傳輸規格，以提供在叢集電腦上進行分散式平行計算的程式規格。

2.3.2分割與平行化

一般個人叢集電腦系統為分散式記憶體平行電腦，每部電腦有獨立的系統、處理器、記憶體等。對於分散式記憶體平行電腦，選擇區域分割法進行平行計算。透過區域分割法，將循序程式中整體區塊進行分割，在這些分割後的區塊分配到多台電腦進行運算，換句話說，讓多台電腦平均分擔處理原先複雜且龐大的問題，每個處理器獨立並同時運算，減少許多閒置時間。區域分割法運算較為廣泛，不受演算法或數學模型的限制，但會面臨如何將複雜、龐大問題做適當切割的問題。

接續上述，進行切割也是循序程式與平行程式的差異，平行程式需要將問題進行切割，並且透過多次的通訊以及資料交換，Foster（1995）提出一個設計平行系統的方法，稱為有條理的設計（Methodical Design），可分為四個部分，分別為切割（Partitioning）、通訊（Communication）、凝聚（Agglomeration）與對應（Mapping），如葉斯暢（2010）所提出圖 2.1 平行系統設計所示。將較為繁複的問題進行切割後，透過訊息傳遞介面進行資料傳遞的動作，將分割後的問題對應到不同的處理器上。進行平行計算後，將資料回傳給主控端，但此時會有同步化的問題（先被分派的工作晚傳回，晚分派的工作早傳回的狀況），建議可待資料都回傳後再進行資料整合，如此一來可以避免同步化產生的問題。

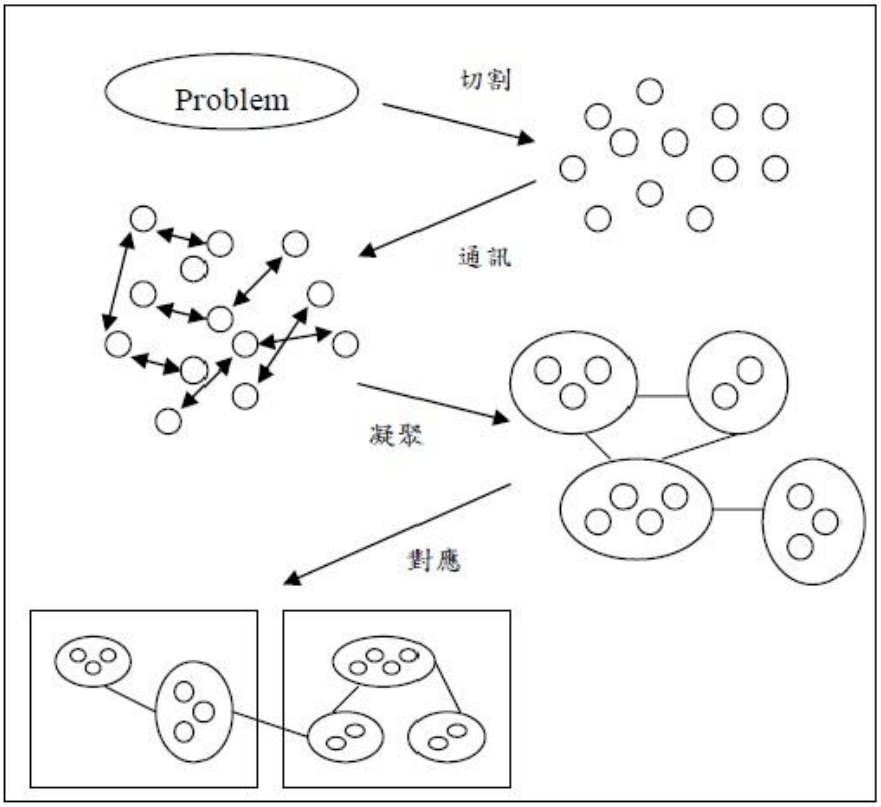

圖 2.1 平行系統設計（葉斯暢，2010）

過去已有研究透過平行計算處理複雜的問題。其中透過 Java RMI 平行計算求解問題的部分，林詩彥（2011）將分散式平行計算的技術，應用於求解供應鏈網絡的規劃，處理多階、多廠區的供應鏈網絡問題。將供應商與製造商家數的排列組合，透過資料分割進行平行計算求解。黃貞翔（2012）則進一步透過平行計算處理有淡季旺季等考量的供應鏈網絡之總體生產規劃問題。

第三章 生產排程規劃模式與平行計算架構說明

本章節分別對於研究所採用的生產排程模式-彈性製程之零工式生產排程與平行計算架構進行說明。首先於 3.1 節說明彈性製程生產排程特性，3.2 節說明本研究的數學模型，包含模型中使用的符號、參數以及目標函數與限制式。3.3 節說明於平行計算技術中，本研究分割資料的方式與說明本研究使用加速求解機制如何應用於本研究的數學規劃。

3.1 彈性製程生產排程

常見的零工式製程排程，由以下主要三個要素組成：

1. 工作（單位：件）
2. 製程（單位：種）
3. 作業（單位：項）
4. 機器（單位：部）

舉例說明，一個由兩件工作、每件工作有兩項作業、共有三部機器可供使用，所形成的生產排程問題，如圖 3.1 工作、作業與機器關係圖所示，彼此關係獨立的兩件工作，各包含一組具有優先順序的作業：工作 1 包含作業 1 與作業 2，先做作業 1，再做作業 2；工作 2 包含作業 2 與作業 3，先執行作業 2，再執行作業 3。不同工作的作業，彼此間關係也是獨立的，而且每項作業缺一不可。這些作業有對應可以使用的機器，在不同的機器上執行，所需要花費的時間也會不同。作業不同，可以使用的機器也不相同，表 3.1 作業與機器使用之對應表為說明作業 1、作業 2、作業 3 可使用的機器編號，作業 1 可以使用機器 1 或機器 2、作業 2 可以使用機器 2 或機器 3、作業 3 可以使用機器 1 或機器 3。

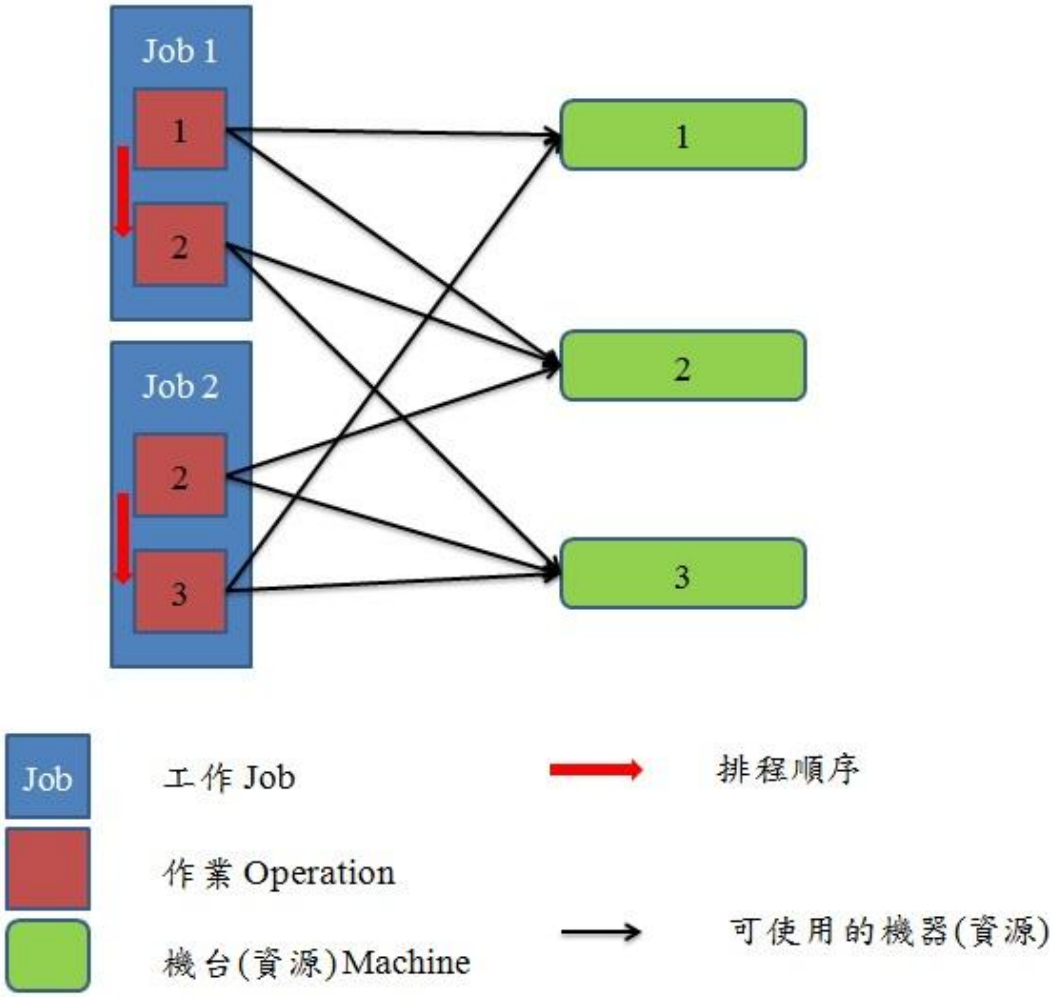

圖 3.1 工作、作業與機器關係圖

表 3.1 作業與機器使用之對應表

Operation	Machine
Operation 1	Machine 1
	Machine 2
Operation 2	Machine 2
	Machine 3
Operation 3	Machine 1
	Machine 3

具有彈性製程的零工式生產排程問題為一種衍生的排程問題。不同於上述的一般零工式生產排程，彈性製程的零工式生產排程內每件工作皆有替代的製程方案（process plan），換句話說，每一件工作，可以由別的排程路徑來完成。接續前例兩件工作、兩項作業、三部機器的排程問題範例，

加入替代製程，成為彈性製程的排程問題如圖 3.2 彈性製程生產排程，工作、作業與機器關係圖所示：

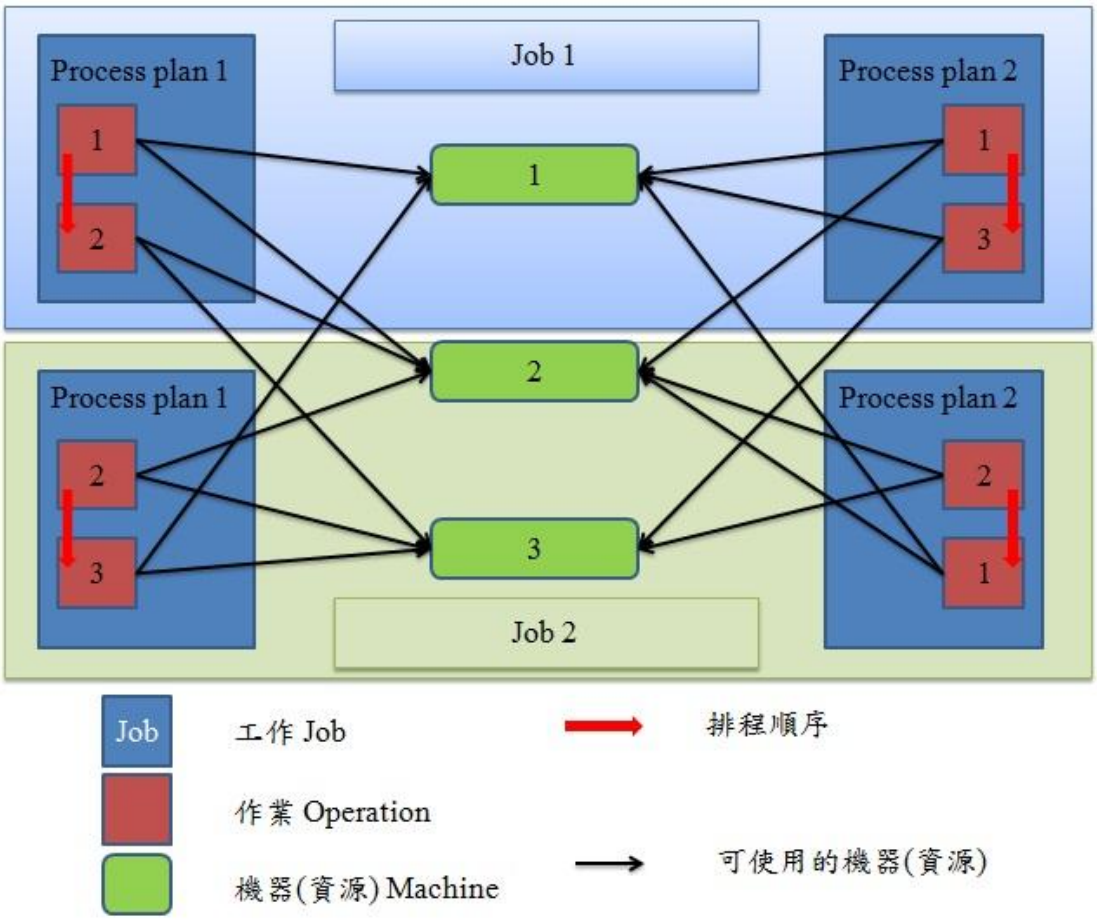

圖 3.2 彈性製程生產排程，工作、作業與機器關係圖

工作（Job 1，Job 2）都有兩種製程方案（Process plan 1， Process plan 2）。工作內，不同的製程方案，其作業、執行順序也不相同。

經過計算後，假如得到的規劃結果為工作 1 交由製程方案 1 執行；工作 2 則是製程方案 2，則可以得到圖 3.3 規劃後之工作、作業與機器關係圖。

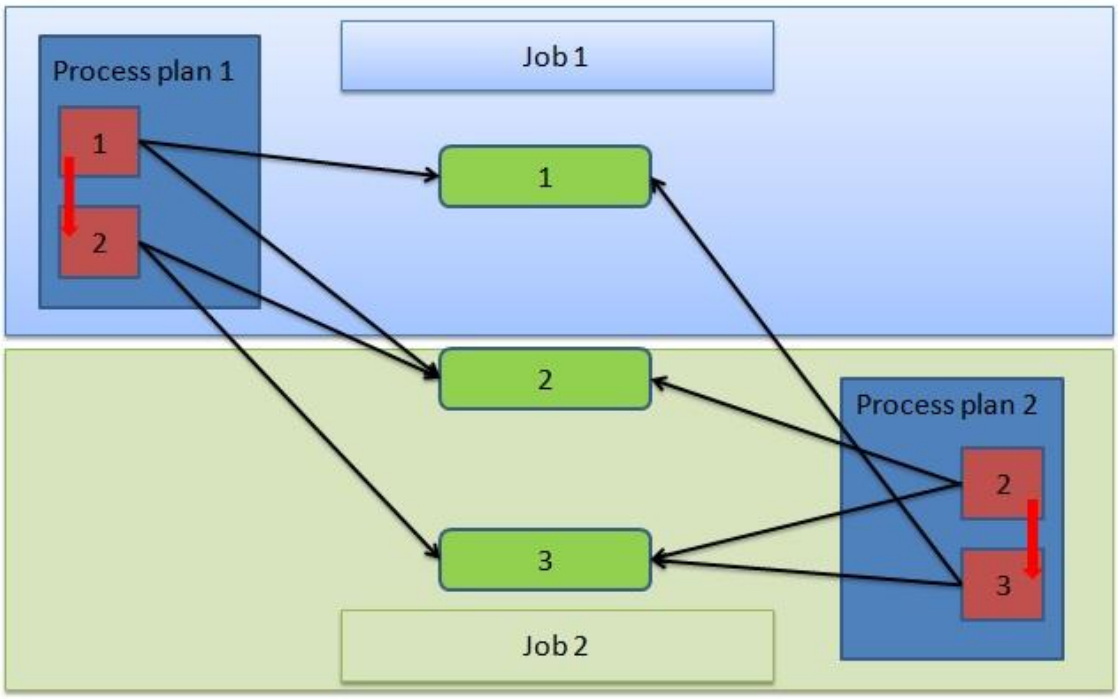

圖 3.3 規劃後之工作、作業與機器關係圖

3.2 生產排程之數學模型

誠如前述，本研究針對彈性製程之生產排程，透過數學規劃，進行求解。底下分別介紹本研究的假設條件與數學模型之符號、目標函數與限制式說明。

3.2.1假設條件

1. 每件工作彼此之間關係獨立，無優先順序。
2. 同一件工作、同一製程方案內的作業，彼此之間有執行先後順序，不可以有跳過或者是違反順序。
3. 同一件工作、同一製程方案內的作業，可以不連續執行，但必須依序執行。舉例說明，機器 1 可以在執行完工作 1 的第 1 項作業後，先執行工作 2 的第 1 項作業，再回到工作 1 接續執行工作 1 的第 2 項作業。
4. 每件工作只能選擇一種製程方案執行，一旦選定開始執行，便不允許更換製程方案。
5. 每件工作皆固定有兩種製程方案。兩種製程方案，規劃選擇時的權重皆相同。

6. 每項作業有指定對應使用的兩部機器可供選擇，兩部機器於規劃選擇時的權重皆相同。
7. 同一時間內不可以有兩項作業在同一機器執行。簡而言之，不允許有作業重疊的情況發生（Non-Overlapping）。
8. 所有排程問題之資訊包含工作數量、作業數量、機器數量、執行時間、每件工作內的製程方案、作業與機器之關係，在排程規劃開始時，皆為已知。
9. 不考慮生產良率或機台之故障率。
10. 不考慮所有工作的完成期限。
11. 規劃期間內，所有工作、作業、機器等，排程資訊不會改變。

3.2.2變數、參數與對應符號說明

本研究所使用的數學模型為引用 Ozguven、Ozbakir 與 Yavuz（2010）所提出的 MILP2 數學模型。本節進行詳細介紹。

下標與符號說明

i 為工作編號 （$i, i' \in J$）

p 為製程方案編號

j 為作業編號 （$j, j' \in O$）

k 為機器編號 （$k \in M$）

J 為工作的集合

M 為機器的集合

O 為作業的集合

$O_{ip_{(i)}}$ 為工作 i 製程方案 $p_{(i)}$ 內有執行先後順序的作業之集合

O_i 為工作 i 內有執行優先順序的作業之集合

M_j　為可以執行作業 j 的所有機器之集合，其中 $M_j \subseteq M$

$M_j \cap M_{j'}$　為可以執行作業 j 與作業 j'的機器之集合

$P_{(i)}$ 為工作 i 的製程方案（ $p_{(i)} \in P_i$ ）

P_i 為工作 i 的製程方案的集合

已知參數

$t_{ip_{(i)}jk}$　為作業 $O_{ip_{(i)}j}$ 由機器 k 執行所需要的時間

L　為一個極大數

決策變數

$Z_{ip_{(i)}}$　為 0 或 1 的整數變數，如果工作 i 的製程方案 $p_{(i)}$ 被選擇，則值為 1；反之為 0

$X_{ip_{(i)}jk}$　為 0 或 1 的整數變數，如果機器 k 被選擇執行作業 $O_{ip_{(i)}j}$，則值為 1；反之為 0

$S_{ip_{(i)}jk}$　為作業 $O_{ip_{(i)}j}$ 於機器 k 執行的開始時間

$C_{ip_{(i)}jk}$　為作業 $O_{ip_{(i)}j}$ 於機器 k 執行的完成時間

$Y_{iji'j'k}$　為 0 或 1 的整數變數，若作業 O_{ij} 在作業 $O_{i'j'}$ 前被執行，則值為 1，反之為 0

C_i　為作業 i 完成時間

$C_{\max}$　為所有工作中，最大完成時間（makespan）

3.2.3數學模型之目標式與限制式說明

本研究數學規劃，目標為最小化最大完成時間，等同於最小化makespan，目標式定義如下：

MIN $C_{\max}$

以下進一步說明限制式：

1. 限制式 （1） 為確保作業$O_{ip_{(i)}j}$只會被指派到一部機器執行。

$$\sum_{k\in M_j} X_{ip_{(i)}jk} = Z_{ip_{(i)}} \quad \forall p_{(i)} \in P_i, \forall j \in O_{ip_{(i)}}, \qquad (1)$$

2. 限制式 （2） 為確保工作 i 只採用一種製程方案。

$$\sum_{p_{(i)}\in P_i} Z_{ip_{(i)}} = 1 \quad \forall i \in J, \qquad (2)$$

3. 假如作業$O_{ip_{(i)}j}$沒有被指派給機器 k，則限制式 （3） 會使得此項作業在機器 k 執行的開始時間與完成時間皆等於零。

$$S_{ip_{(i)}jk} + C_{ip_{(i)}jk} \le (X_{ip_{(i)}jk}) \cdot L \quad \forall i \in j, \forall p_{(i)} \in P_i, \forall j \in O_{ip_{(i)}}, \forall k \in M_j, \qquad (3)$$

4. 相反的，倘若作業$O_{ip_{(i)}j}$被指派到機器 k 執行，則限制式 （4） 能保證其開始時間與完成時間之差，會大於或等於作業$O_{ip_{(i)}j}$在機器 k 上執行的所需時間。

$$C_{ip_{(i)}jk} \ge S_{ip_{(i)}jk} + t_{ip_{(i)}jk} - (1 - X_{ip_{(i)}jk}) \cdot L \quad \forall i \in j, \forall p_{(i)} \in P_i, \forall j \in O_{ip_{(i)}}, \forall k \in M_j, \qquad (4)$$

5. 透過限制式 （5） 與限制式 （6），確保作業O_{ij}、$O_{i'j'}$不會在相同任意時間、任意機器（指可以處理作業O_{ij}、$O_{i'j'}$的機器，$M_j \cap M_{j'}$）被執行。

$$\sum_{p_{(i)}\in P_i} S_{ip_{(i)}jk} \ge \sum_{p'_{(i)}\in P_{i'}} C_{i'p'_{(i)}j'k} - (Y_{iji'j'k}) \cdot L \quad \forall i < i', \forall j \in O_{ip_{(i)}}, \forall j' \in O_{i'p'_{(i')}}, \forall k \in M_j \cap M_{j'}, \qquad (5)$$

$$\sum_{p'_{(i)}\in P_{i'}} S_{i'p'_{(i)}j'k} \ge \sum_{p_{(i)}\in P_i} C_{ip_{(i)}jk} - (1 - Y_{iji'j'k}) \cdot L \quad \forall i < i', \forall j \in O_{ip_{(i)}}, \forall j' \in O_{i'p'_{(i')}}, \forall k \in M_j \cap M_{j'}, \qquad (6)$$

6. 限制式 （7） 為確保不違反每件工作內的作業執行的先後順序。

$$\sum_{k\in M_j} S_{ip_{(i)}jk} \geq \sum_{k\in M_j} C_{ip_{(i)}j-1,k} \quad \forall i < J, \forall p_{(i)} \in P_i, \forall j \in O_{ip_{(i)}} - \{O_{ip_{(i)}f_{(i)}}\}, \qquad (7)$$

7. 並透過限制式 （8） 決定每件作業的完成時間。

$$C_i \geq \sum_{k\in M_j} C_{ip_{(i)},O_{ip(i)l(i)},k} \quad \forall i \in J, \forall p_{(i)} \in P_i, \qquad (8)$$

8. 進一步透過限制式 （9） 決定最大完工時間。

$$C_{\max} \geq C_i \quad \forall i \in J, \qquad (9)$$

9. 以下制定 0 或 1 的整數變數$Z_{ip_{(i)}}$ 、 $X_{ip_{(i)}jk}$ 、$Y_{iji'j'k}$與確保開始時間$S_{ip_{(i)}jk}$與完成時間$C_{ip_{(i)}jk}$為正整數。

$$Z_{ip_{(i)}} \in \{0,1\} \ \forall i \in J, \forall p_{(i)} \in P_i,$$

$$X_{ip_{(i)}jk} \in \{0,1\} \ \forall i \in J, \forall p_{(i)} \in P_i, \forall j \in O_{ip_{(i)}}, \forall k \in M_j,$$

$$S_{ip_{(i)}jk} \geq 0 \ \forall i \in J, \forall p_{(i)} \in P_i, \forall j \in O_{ip_{(i)}}, \forall k \in M_j,$$

$$C_{ip_{(i)}jk} \geq 0 \ \forall i \in J, \forall p_{(i)} \in P_i, \forall j \in O_{ip_{(i)}}, \forall k \in M_j,$$

$$Y_{iji'j'k} \in \{0,1\} \ \forall i \in i', \forall j \in O_i, \forall j' \in O_{i'}, \forall k \in M_j \cap M_{j'},$$

$$C_i \geq 0 \quad \forall i \in J$$

3.3 分散式平行計算架構

如前面章節所說明，透過數學規劃求解所需花費的時間可能隨著問題的複雜度，呈指數成長。因此本研究對於數學模型將進行分散式平行計算求解。

3.3.1循序程式的平行化計算

本節將針對前述之數學模型進行平行化，利用數理規劃軟體 LINGO 10.0 版所提供的動態連結資料庫（Dynamic Link Library, DLL）的功能，搭

配 Java RMI 套件，將數學模型進行問題切割、平行求解以及回饋等動作，求最佳解。平行計算流程如圖 3.4 平行計算流程說明圖：

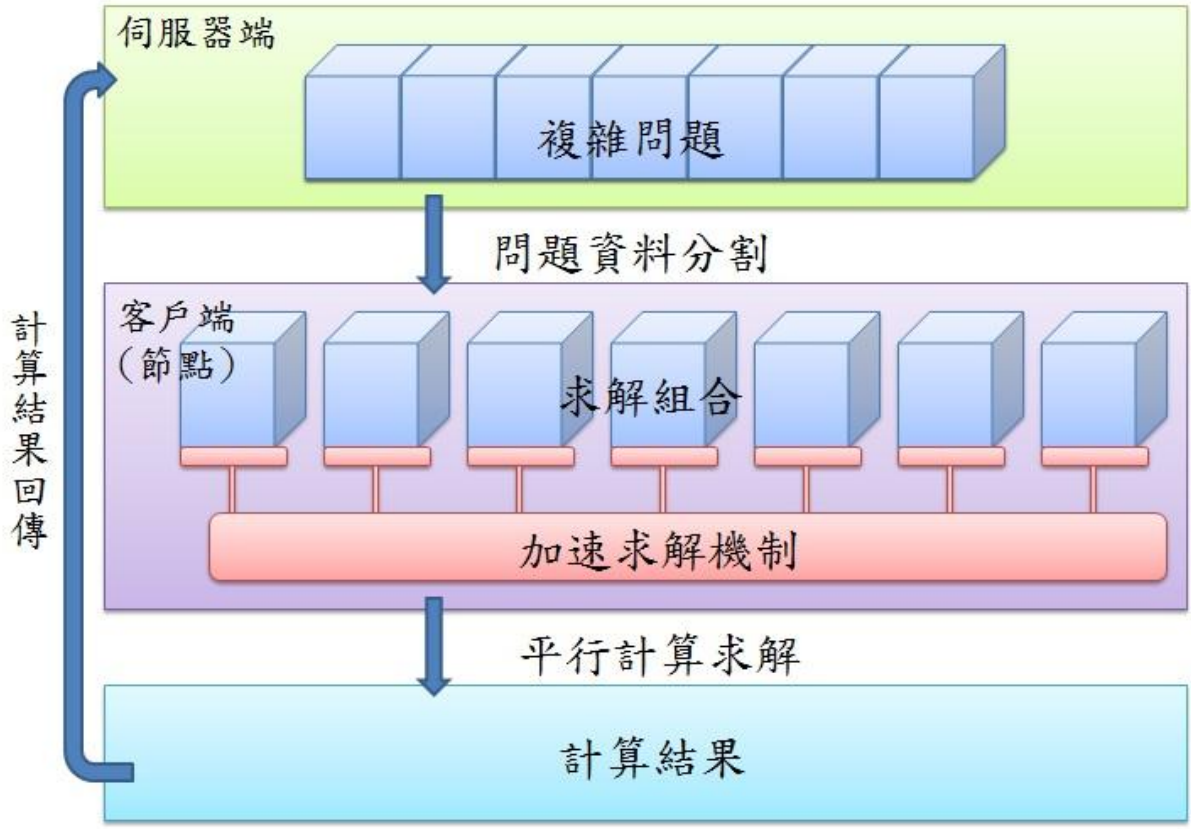

圖 3.4 平行計算流程說明圖

本研究的平行計算採用遠端程序的方法，透過 Java RMI 將分割完的資料，以物件的方式分配給各節點端，各節點端在接收伺服端所傳送的資料後進行計算，並且將計算結果回傳給伺服端，再進行下一組資料的計算，從所有計算結果中找尋最佳的規劃結果。

而在平行計算前必須先作資料切割，使用者於伺服端輸入欲選擇的模型規模，在模型規模必須考量資料分割的組合數，分割後的組合如無法滿足需求，則求解結果會與全域規劃之目標函數值有差異。

表 3.2 小規模資料分割組合表

工作編號 / 製程方案 組合編號 / 編號	Job 1	Job 2	Job 3
1	1	1	1
2	2	1	1
3	1	2	1
4	1	1	2
5	1	2	2
6	2	1	2
7	2	2	1
8	2	2	2

如表 3.2 小規模資料分割組合表所示，本研究針對多件工作與工作內之製程方案進行資料分割。以本研究小規模之生產排程為例，將原本的 3 個工作的彈性製程之排程問題，分割成多個單一製程排程問題。由於每 1 件工作有兩種製程方案可以選擇，3 件工作的排程就存在 8 種製程方案的組合（2^3=8）。透過資料分割，將分割後所產生的 8 種組合資料以物件的方式分派到各個節點端，這些節點端在接收伺服端所傳送的資料後進行運算（每個節點所運算的組合不得重複），並將運算結果回傳到伺服端，在進行下一組資料的運算，從所有運算值找尋最佳的規劃結果。

3.3.2平行計算演算法

在 LINGO DLL 中提供了 LSgetCallbackInfoDoubleLng（ int nLngEnv, int nObject, double dCBInfo[]）的方法，此方法可以提供 LINGO 在計算過程中的上界（Upper Bound）和下界（Lower Bound）資訊，底下說明本研究使用的平行計算演算法：

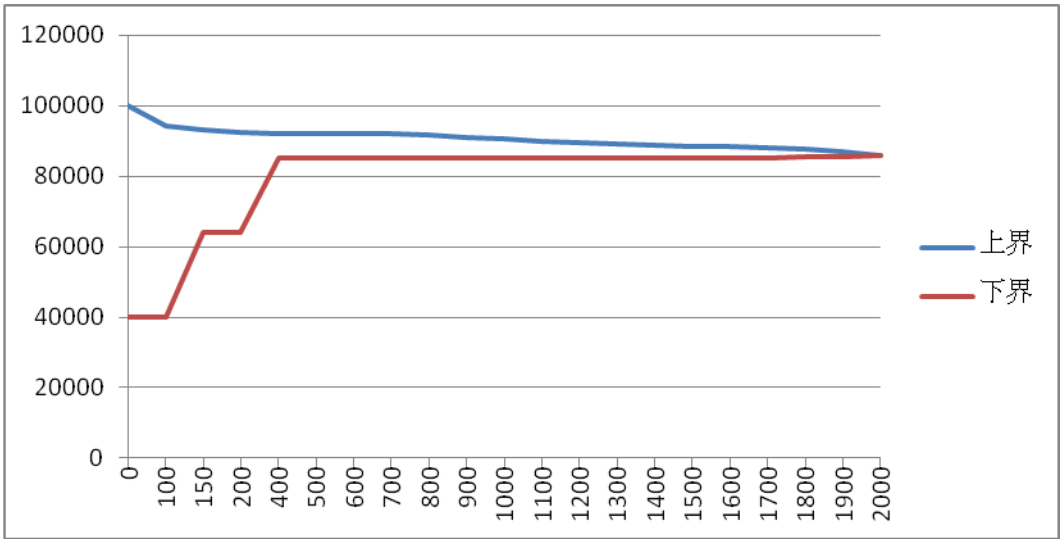

圖 3.5LINGO 計算上界與下界變化

如圖 3.5LINGO 計算上界與下界變化，上界和下界的交會處為所求解問題的最佳解，如果單一從上界或是的收斂程度來停止整體計算的話，那可能會拒絕掉很多有意義的解。因此本研究提出加速求解機制，加速求解機制將每一個求解完成的組合之計算結果先回傳給伺服器，進而持續挑選更佳的計算結果。以求以本研究求解極小化目標函數值為例，如果後續的計算結果中有幾組數學模型計算出來的子集合下界已經大於伺服器端中的現存的最小目標值時，代表這些組合即便運算後，其最佳目標解一定也大於伺服器的現存最小目標值；因此這些組合就沒有求解的意義，節點端將停止對於這些組合的計算，藉此提高平行計算上的效率。

在平行計算的過程中，伺服端會陸續接收來自計算節點所回傳的規劃結果，但是這些結果當中，可能存在比當前所有組合最好的規劃結果更好的目標函數值。以極小化目標函數值為例，從伺服端回傳的規劃結果可能優於當前最小目標函數值（目標函數值更小）。面對此情況，伺服端必須根據不斷判別自計算節點所回傳的結果是否有更優的控制下界值，如果有更佳的控制下界值，則必須立即更新 PREV_Obj 值，將新的控制下界值傳遞給各個計算節點，判別是否應停止當前計算。如果當前運算的下界值，已經高於所收到的新控制下界值，代表該節點運算的結果最終將不會低於當前新控制下界值，並立即停止當下運算，反之則否。

如圖 3.6 加速求解機制說明所示，以目標函數求解最小化 makespan 為例，當前計算組合的下界值高於當前最佳值時，說明此組合如果繼續求解，所得到該組合的最佳解將會大於當前最小值，即當前最佳值優於該組合的最佳解，代表此計算組合已不具有求解意義。因此平行計算程式將中止該組合的計算，以提高求解效率。

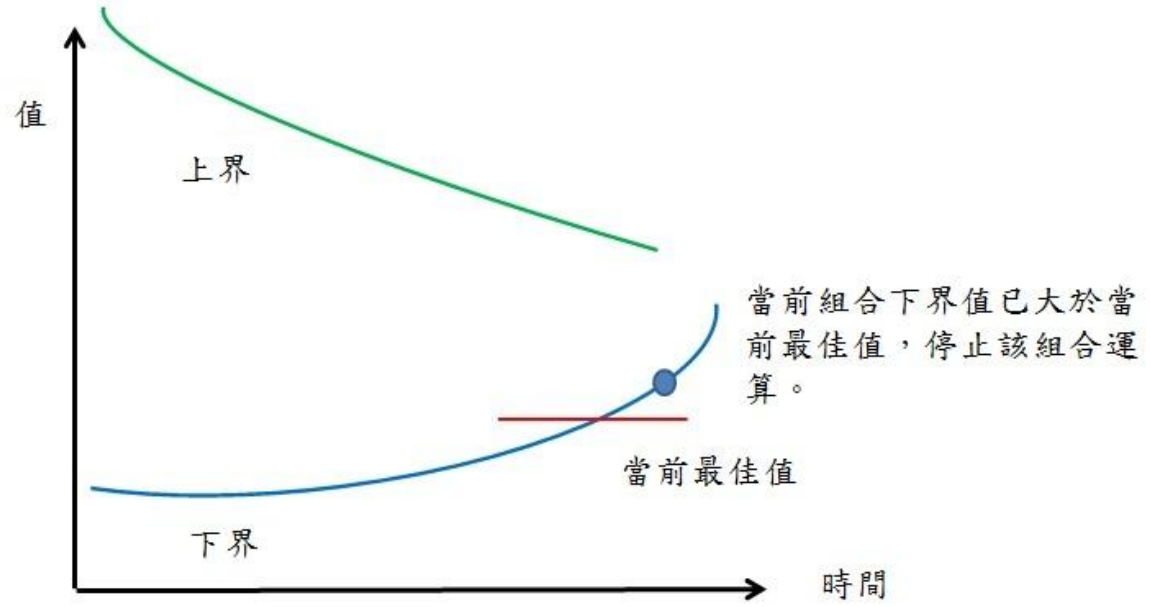

圖 3.6 加速求解機制說明

根據上述的演算法概念，在節點端所執行的界面中，增加一個方法物件負責接收子集合的最佳規劃結果之目標值，並將此種演算法寫入節點端的程式中；其介面中所定義的方法物件如下所示：

1. void parameterTransfer（String fName, byte[] fByte）：此方法物件主要功能為伺服端傳送數學模型的參數給計算節點端，本研究所採用的方式為數學模型的參數寫在文字檔內，以文字檔的方式傳送。參數 fName 為文字檔的檔案名稱，fByte 為文字檔轉換成的位元陣列。
2. void modelConstruct（int nI, int nJ）：此方法物件的主要功能為計算節點端根據伺服端傳送的參數建立數學模型，本研究所採用的方式為利用 Java 建立 LINGO 所需要讀取的 lng 文字檔。
3. void UBLimit（double PREV_Obj）：此方法物件的主要功能為計算節點端根據伺服端傳送的參數來限制上界。參數 PREV_Obj 為子集合的最

佳規劃結果之目標值或者是平行計算中所得到的較佳目標值。

4. void combinationTransfer（String COMBI_IJ）：此方法物件的主要功能為計算節點端接收伺服端所分配的組合，計算節點端將會依照此組合的組合方式進行計算。參數 COMBI_IJ 為依照組合方式所轉換成的字串。

5. byte[] modelSolve（int cCount）：此方法物件的主要功能為伺服端呼叫此物件並且傳送參數給計算節點端後，計算節點端針對所接收到的組合進行數學模型的求解，求解完得到的結果寫入文字檔，轉換成位元陣列並且回傳給伺服端。參數 cCount 為組合的編號，為用來建立文字檔的檔案名稱的參數。

由圖 3.7 平行計算流程架構說明：

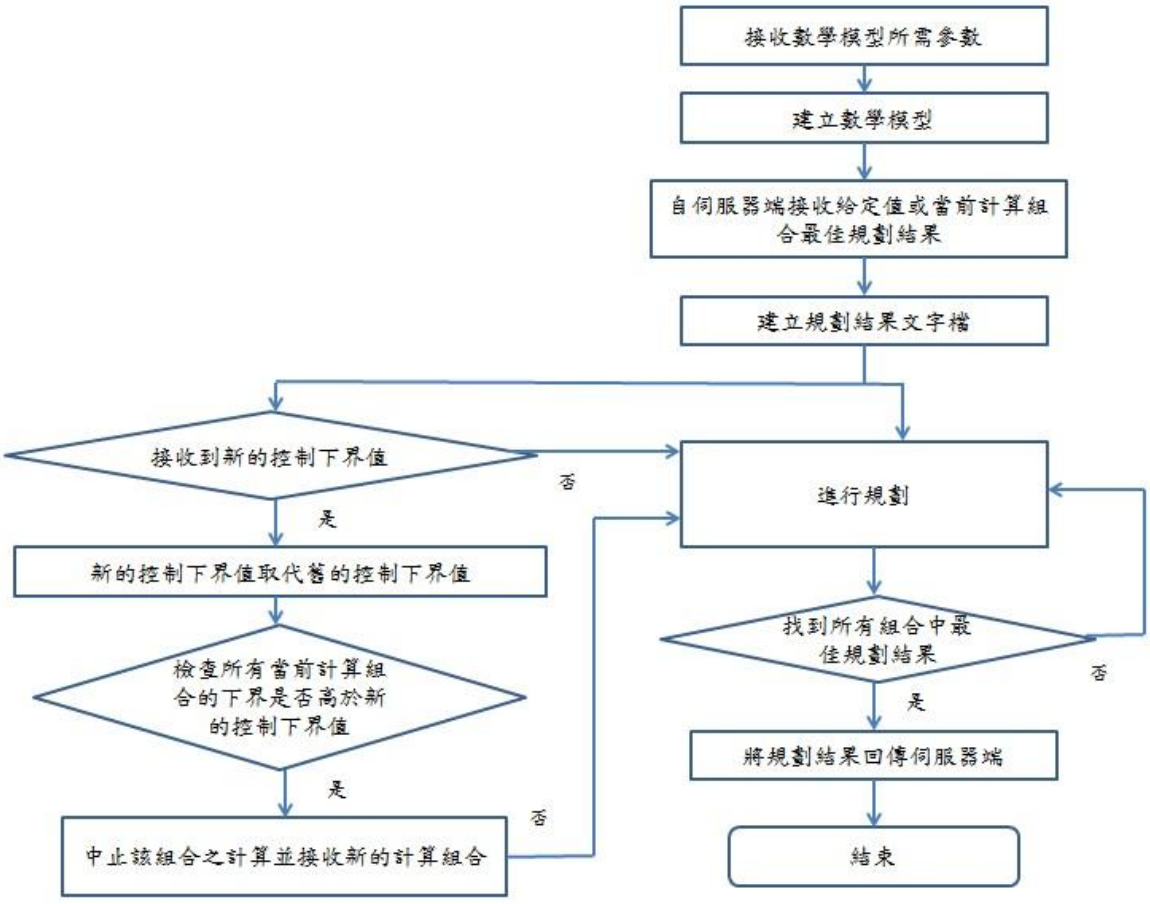

圖 3.7 平行計算流程架構

根據圖所示，在平行計算的過程中，透過加速求解機制檢查更新的控制下界值，如果接收到新的控制下界值，則會檢查當前計算組合之下界值

是否高於新的控制下界值，如果是，代表此項組合之計算為無意義的求解，並中止此組合的計算，接收下一個尚未進行計算求解的組合。

第四章 實驗與分析

本章節將針對本研究之實驗進行說明，透過相同的實驗環境，使用不同的水準因子與參數設定進行實驗與分析。首先針對實驗的環境、參數設定做說明，接著針對本實驗得到的數理規劃軟體與電腦計算之結果進行分析。實驗分析，包含兩個部份：

1. 針對水準因子不同與問題規模不同所得之結果進行分析
2. 針對全域最佳規劃結果與分散式平行計算規劃之結果進行比較。

4.1 實驗說明

4.1.1實驗環境

1. 規劃時限：排程規劃時間以 24 小時為限，超過即停止當前所有計算。
2. 生產排程環境:本研究探討生產排程環境為彈性製程之零工式生產排程。
3. 實驗系統環境：本實驗之作業系統為 WINDOWS 7 企業版，CPU 為 Intel（R）Core（TM）i5-3570CPU@3.40GHz，4.00GB RAM。求解工具為 Java 與 LINGO 10.0 版。

4.1.2實驗因子

實驗因子可分為環境因子與控制因子，本研究之環境因子為待規劃之排程問題規模大小；控制因子為處理器數量。以下說明各因子：

1. 環境因子：排程問題規模分為小規模、中規模與大規模。小規模排程問題包含 2 件工作，每件工作個包含 2 種製程方案，每種製程方案有 1 項作業，共有 2 部機器可供使用。中規模排程問題包含 5 件工作，每件工作個包含 2 種製程方案，每種製程方案至多有 4 項作業，共有 5 部機器可供使用。大規模排程問題則是包含 6 件工作，每件工作個包含 2 種製程方案，每種製程方案至多有 5 項作業，共有 6 部機器可供使用。
2. 控制因子：小規模的控制因子為 1 個處理器數目；中規模的控制因子分為 1 個、2 個、4 個、8 個、16 個和 32 個處理器數目；大規模的控制因子則分為 1 個、2 個、4 個、8 個、16 個、32 個和 64 個處理器數目。

各因子整理如表 4.1 各因子說明：

表 4.1 各因子說明

實驗因子	說明	工作數量	作業數量	機器數量
環境因子: 排程問題規模	小規模	2	1	2
	中規模	5	4	5
	大規模	6	5	6
控制因子: 處理器數量	小規模控制因子：1個處理器			
	中規模控制因子：1、2、4、6、8、16、32個處理器			
	大規模控制因子：1、2、4、6、8、16、32、64個處理器			

4.1.3參數設定

本研究的小規模、中規模與大規模排程問題之參數設定整理成表 4.3 小規模參數設定、表 4.7 中規模參數設定、表 4.10 大規模參數設定，將分別於分析各排程規模排程問題時進行說明。

4.2 實驗分析

本節將分成三個階段進行如表 4.2 實驗階段說明，先於 4.2.1 小節提出小規模排程問題，驗證本研究之數學模型為求解最小化 makespan，接著於 4.2.2 小節建立與 Ozguven、Ozbakir 與 Yavuz（2010）所提出之 P1-11 排程參數與數理模型作為中規模排程問題，驗證本研究求解與其一致。之後於 4.2.3、4.2.4 與 4.2.5 小節，計算大規模之排程問題透過單機計算之結果和比較中規模、大規模的單機計算結果與透過平行化規劃的差異，對規劃結果做分析。

表 4.2 實驗階段說明

實驗階段	建立的模型	模型規模說明	實驗目的
第一階段	小規模 數學模型	由兩件工作組成，每件工作包含兩種製程方案，每種製程方案包含一項作業，共有兩部機器可供使用。	驗證模型為求解最佳解
第二階段	中規模 數學模型	由五件工作組成，每件工作包含兩種製程方案，每種製程方案最多包含四項作業，共有五部機器可供使用。	驗證求解相同排程問題時，規劃結果與引用之文獻一致。
第三階段	大規模 數學模型	由六件工作組成，每件工作包含兩種製程方案，每種製程方案最多包含五項作業，共有六部機器可供使用。	面對規模較大的問題，透過多處理器平行計算分析求解效率的變化。

4.2.1小規模數學模型之全域最佳解

首先建立提供作為驗證之小規模排程問題，進行驗證之排程問題參數設定如表 4.3 小規模參數表，表內說明編號 1 工作與編號 2 工作內不同作業於不同機器上執行所需時間，時間單位為分鐘。驗證之排程問題包含兩件工作，每件工作包含兩種製程方案，每種製程方案內有一項作業，每項作業在不同機器上所需時間如表 4.4 小規模工作內容對應表，表內編號分別為各工作、各製程方案和各作業的編號。

表 4.3 小規模參數表

Operations	Machines	Job1	Job2
1	1	4	6
	2	5	3
2	1	7	4
	2	5	4

表 4.4 小規模工作內容對應表

Job	Process Plan	Operation
1	1	1
	2	2
2	1	1
	2	2

進行驗證的部份，透過列舉方法，如表 4.5 小規模之所有可行組合，得到總計有 16 種可行之規劃組合，圖中 J、P、O、M 分別表示為工作、製程方案、作業和機器的編號，舉例說明，編號組合 1 的規劃結果為工作 1 與工作 2 皆於機器 1 執行，規劃結果得到的最大完成時間為 10 分鐘。從表 4.5、小規模之所有可行組合中，可得知編號 5 與編號 6 之組合為最佳規劃結果，即排程問題經規劃結果後，makespan 為 4 分鐘，編號 5 與編號 6 組合的規劃結果優於其它所有可行之組合。本研究建立數理模型，透過數理軟體 LINGO 計算，所得之結果為編號 5 的組合。驗證本研究之數理模型為求解最小化 makespan。LINGO 規劃結果如圖 4.1 小規模甘特圖，作業 1 於機器 1 執行，作業 2 於機器 2 值型。表中的橫向座標為作業執行時間，單位為分鐘；縱向座標為機器編號。

表 4.5 小規模之所有可行組合

組合編號	機器	執行之作業		makespan	組合編號	機器	執行之作業		makespan
1	M1	J1,P1,O1	J2,P1,O1	10	9	M1	J2,P1,O1		6
	M2					M2	J1,P1,O1		
2	M1	J1,P1,O1	J2,P1,O2	8	10	M1	J2,P1,O1		6
	M2					M2	J1,P1,O2		
3	M1	J1,P1,O2	J2,P1,O1	13	11	M1	J2,P1,O2		5
	M2					M2	J1,P1,O1		
4	M1	J1,P1,O2	J2,P1,O2	11	12	M1	J2,P1,O2		5
	M2					M2	J1,P1,O2		
5	M1	J1,P1,O1		4	13	M1			8
	M2	J2,P1,O1				M2	J1,P1,O1	J2,P1,O1	
6	M1	J1,P1,O1		4	14	M1			9
	M2	J2,P1,O2				M2	J1,P1,O1	J2,P1,O2	
7	M1	J1,P1,O2		7	15	M1			8
	M2	J2,P1,O1				M2	J1,P1,O2	J2,P1,O1	
8	M1	J1,P1,O2		7	16	M1			9
	M2	J2,P1,O2				M2	J1,P1,O2	J2,P1,O2	

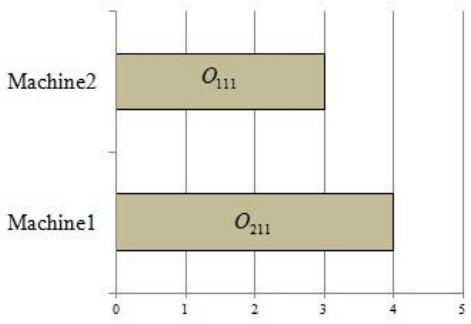

圖 4.1 小規模甘特圖

4.2.2 中規模數學模型之全域最佳解

如第二章所述，本研究之數理模型引用 Ozguven、Ozbakir 與 Yavuz（2010）所提出的 MILP-2 模型。在此，中規模實驗採用 MILP-2 模型與 Ozguven 等人提出的 P1-11 排程問題參數，驗證求解結果與其一致。中規模工作內容如表 4.6 中規模工作內容對應表，舉例說明，工作 1 的製程方案 2，包含執行先後順序為作業 5、作業 4、接著執行作業 2，最後執行作業 3。作業時間如表 4.7 中規模之參數設定。規劃結果如圖 4.2 中規模甘特圖，驗證結果與 Ozguven 等人所提出之結果一致。

表 4.6 中規模工作內容對應表

Job	Process Plan	Operation Sequences
1	1	4→5→2→3
	2	5→4→2→3
2	1	1→2→4
	2	1→4→2
3	1	3→5→1→4
	2	1→5→3→4
4	1	2→3→5→4
	2	3→2→5→4
5	1	4→1→3
	2	4→6

表 4.7 中規模之參數設定

Operations	Machines	Job1	Job2	Job3	Job4	Job5
1	3	-	68	54	-	55
	5	-	84	74	-	85
2	2	40	10	-	7	-
	3	88	15	-	13	-
3	5	5	-	18	40	17
	4	10	-	52	47	42
4	4	71	87	14	11	64
	2	74	88	22	30	65
5	2	18	-	56	66	-
	3	22	-	64	80	-
6	2	-	-	-	-	74
	1	-	-	-	-	88

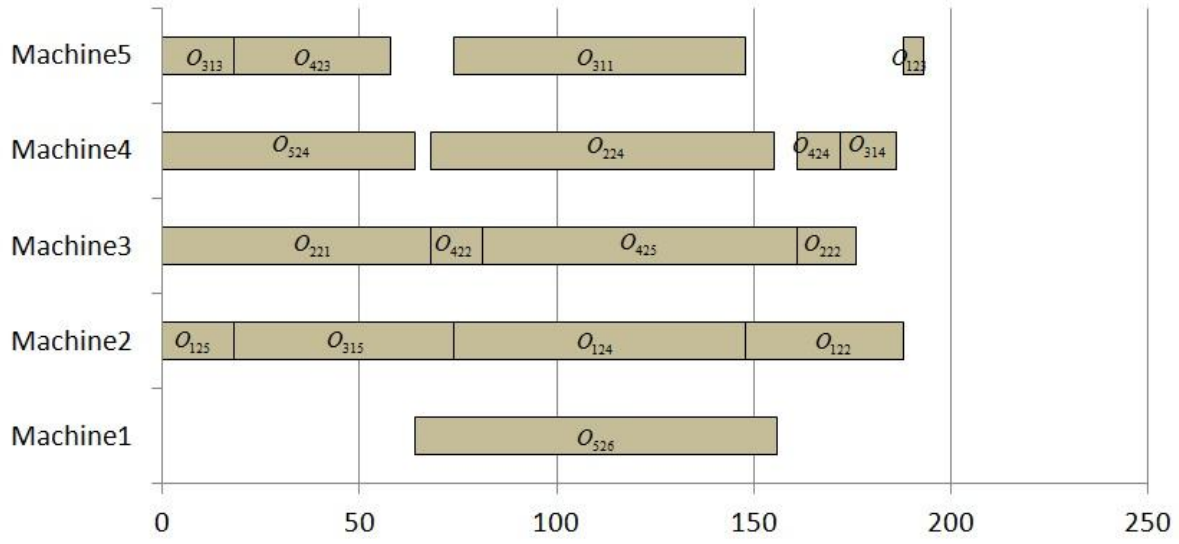

圖 4.2 中規模甘特圖

4.2.3 中規模之平行計算的最佳規劃結果

本節透過 Java RMI 分散式平行計算，同時間多個處理器進行計算，整理所得之規劃結果，進行分析。如表 4.8 中規模多處理器計算結果所示，針對中規模排程問題，進行 2 個、4 個、6 個、8 個、16 個與 32 個處理器的平行計算，得到最佳規劃結果皆為 193 分鐘。計算時間隨平行計算的處理器數量增加而有明顯減少的趨勢，單一處理器求全域最佳解需要花費 31569 秒，透過兩個處理器平行計算，時間可以縮短至 17697 秒，到透過 32 個處理器進行平行計算時，求解時間已經縮短到 2931 秒。如圖 4.3 中規模多處理器計算時間。計算時間單位為秒。

表 4.8 中規模多處理器計算結果

Model size	Job	operation	machine	CPU 數量	Time(s)	Opt. Solution	Best Obj.	Obj. Bound
中	5	4	5	1	31569	193	Global optimal solution found	
中	5	4	5	2	17697	193		
中	5	4	5	4	8551	193		
中	5	4	5	8	4086	193		
中	5	4	5	16	3484	193		
中	5	4	5	32	2931	193		

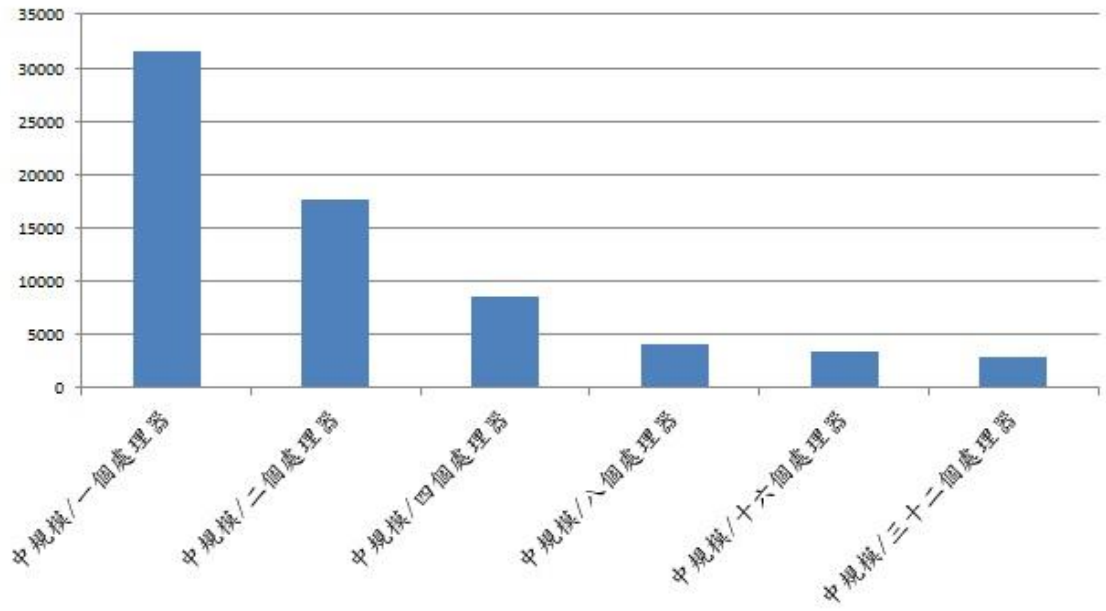

圖 4.3 中規模多處理器計算時間

4.2.4大規模數學模型之全域最佳解

4.2.4 小節與 4.2.5 小節進一步將排程問題將工作數量、作業數量、機器數量等相關排程規模擴大，作為大規模排程問題，進行單處理器與多處理器實驗分析。大規模工作內容如表 4.9 大規模工作內容對應表，作業時間如表 4.10 大規模之參數設定。但大規模排程問題於單處理器計算時，無法在本研究環境設定的 24 小時限內完成最佳解的規劃。

表 4.9 大規模工作內容對應表

Job	Process Plan	Operation Sequences
1	1	4→5→2→3→6
	2	5→4→2→3→6
2	1	1→2→4
	2	1→4→2
3	1	3→5→1→4
	2	1→5→3→4
4	1	2→3→5→4
	2	3→2→5→4
5	1	4→1→3
	2	4→6
6	1	3→2→1→6→5
	2	3→1→2→6→5

表 4.10 大規模之參數設定

Operations	Machines	Job1	Job2	Job3	Job4	Job5	Job6
1	3	-	68	54	-	55	14
	5	-	84	74	-	85	26
2	2	40	10	-	7	-	40
	6	88	15	-	13	-	45
3	5	5	-	18	40	17	83
	4	10	-	52	47	42	40
4	4	71	87	14	11	64	-
	2	74	88	22	30	65	-
5	6	18	-	56	66	-	20
	3	22	-	64	80	-	17
6	2	6	-	-	-	74	13
	1	8	-	-	-	88	11

4.2.5大規模之平行計算的最佳規劃結果

使用 2 個、4 個、8 個、16 個、32 個與 64 處理器進行平行計算，得到的結果整理如表 4.11 大規模多處理器計算結果與圖 4.4 大規模甘特圖。其中，單處理器與 2 個處理器平行計算並沒有於時限 24 小時內完成最佳解的規劃，如圖 4.5、大規模多處理器計算時間的斜線長條圖所示。並列出中止運算時所得到的當前區域最佳解，如表 4.11 大規模多處理器計算結果。

表 4.11 大規模多處理器計算結果

Model size	Job	operation	machine	CPU 數量	Time(s)	Opt. Solution	Best Obj.	Obj. Bound
大	6	5	6	1	>86400	-	195	165
大	6	5	6	2	>86400	-	193	165
大	6	5	6	4	19706	192	Global optimal solution found	
大	6	5	6	8	10633	192		
大	6	5	6	16	8413	192		
大	6	5	6	32	6545	192		
大	6	5	6	64	5997	192		

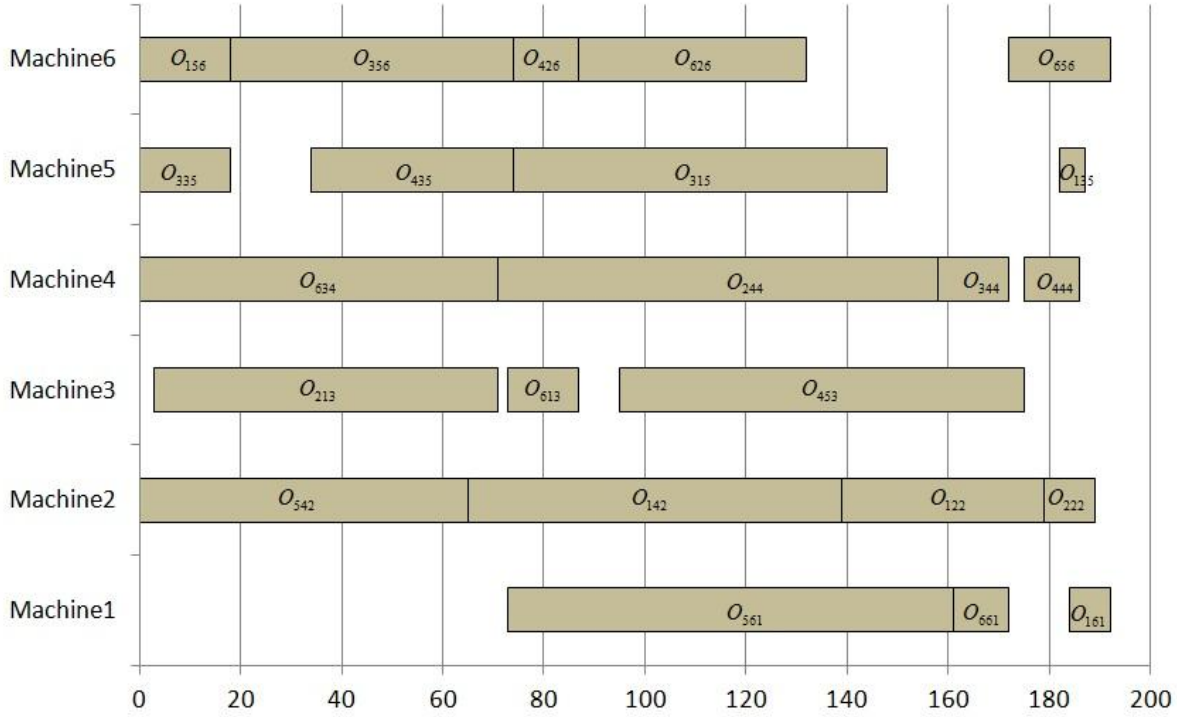

圖 4.4 大規模甘特圖

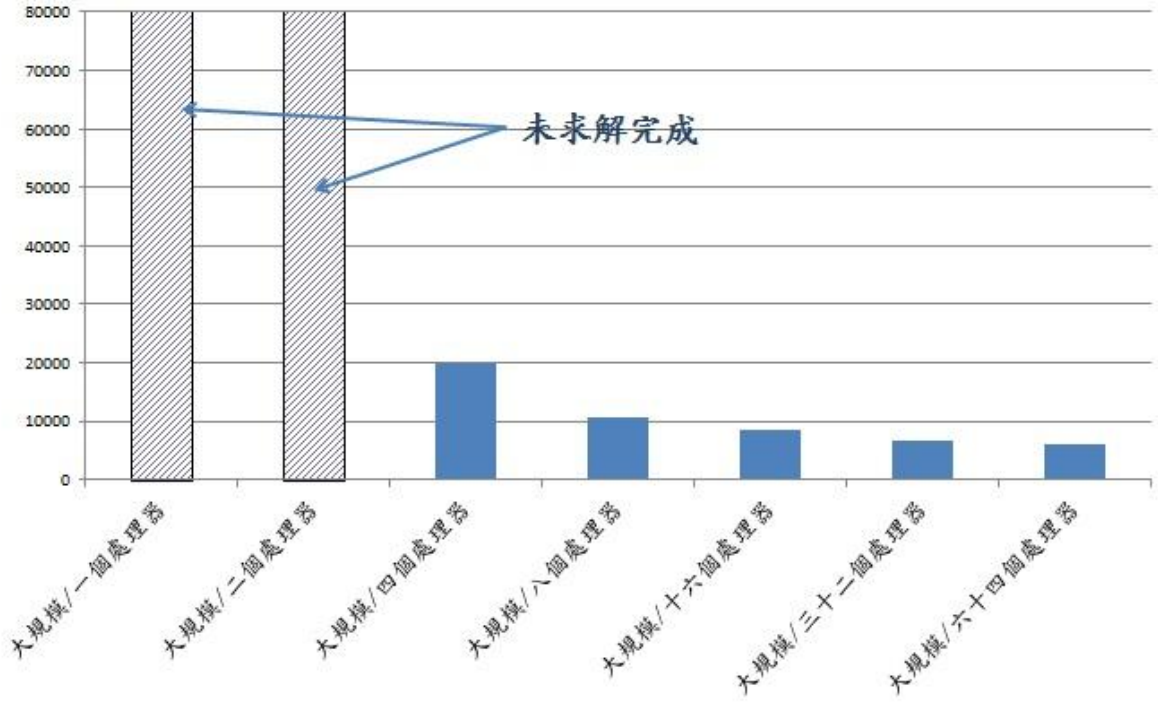

圖 4.5 大規模多處理器計算時間

4.3 結果討論

本研究的實驗分成三個階段進行實驗如表，第一階段建立小規模數學

模型，求解排程問題，並透過手算列舉的方式，整理出所有小規模排程問題可行的規劃組合。透過數學規劃軟體建立小規模數學模型求解問題，所得到的規劃結果與列舉的規劃組合進行比較，驗證本研究所使用的數學規劃軟體為求解最小化 makespan，即所求之解為最佳解。

第二階段建立中規模數學模型，透過數學規劃軟體代入引用文獻的排程問題參數，求最佳規劃結果。將規劃之結果與引用文獻歸化的結果做比較，驗證本研究的規劃結果與引用文獻的規劃結果一致。

第三階段建立大規模數學模型進行實驗，並將中規模與大規模排程問題透過平行計算規劃求解。實驗結果發現，大規模排程問題透過單一處理器計算與兩個處理器的平行計算無法在規劃時限內完成求解，但是在透過四個以上處理器的平行計算能顯示出其求解效率隨著處理器數量的增加有提升的趨勢。

總結本章節所實驗之小規模、中規模和大規模的單處理器與多處理器之運算結果。針對不同排程規模的單處理器計算，可以推導出數學規模逐漸增大時，計算所需時間也明顯增加，包含較大規模無法在規劃時限內求解完成之結果。透過平行計算，將複雜的數學規劃問題，同時由多個處理器計算，有效降低規劃時間。如此說明本研究對於處理 NP-Hard 彈性製程之零工式生產排程的數學模型透過多處理器平行計算求解降低規劃時間，相較於中規模單一處理器的計算，中規模透過 32 個處理器之平行計算可以節省約 90%的規劃時間，如表 4.12 中規模規劃時間比較。

表 4.12 中規模規劃時間比較

中規模 處理器數量	求解時間 (單位:秒)	以單一處理器之計算時間為比較基礎:求解所需時間百分比
1	31569	100%
2	17697	56%
4	8551	27%
8	4086	13%
16	3484	11%
32	2931	9%

由於大規模的排程規劃中，單一處理器與兩個處理器的計算無法在規劃時限內求解完成，因此以大規模四個處理器平行計算所需的規劃時間作為比較基礎。大規模排程的規劃時間，相較於單一處理器的規劃時間，大規模透過 64 個處理器之平行計算，可以節省約 70%的規劃時間，如表 4.13 大規模規劃時間比較，克服了計算時間隨問題規模增大成指數成長的問題，顯示在實務上的可行性與實用性。

表 4.13 大規模規劃時間比較

大規模 處理器數量	求解時間 (單位:秒)	以四個處理器之計算時間為比較基礎:求解所需時間百分比
4	19706	100%
8	10633	54%
16	8413	43%
32	6545	33%
64	5997	30%

第五章 結論與未來發展方向

5.1 結論

實際的現場排程情況相當複雜，本研究亦無法考慮所有細節、限制、情境，深入分析。基於過去研究較多探討啟發式演算法求解排程問題與啟發式演算法無法保證規劃結果為最佳解兩個原因，本研究針對過去研究較少探討的 NP-Hard 彈性製程之零工式生產排程問題，透過數學規劃，計算求解最佳規劃結果。

本研究引用 Ozguven、Ozbakir 與 Yavuz（2010）所提出的解彈性製程之零工式生產排程數學模型，透過數理規劃軟體求解。實驗分成三個階段進行，第一階段，建立 2 件工作，每件工作包含 1 項作業，總共 2 部機器的小規模排程問題的數學模型。透過列舉所有可行組合與數學規劃軟體所計算結果的比較，驗證本研究採用的數學模型為求解最佳解。第二階段，建立 5 件工作，每件工作至多包含 4 項作業，總共 5 部機器的中規模數學模型。驗證本研究數學規劃軟體求得結果驗證與 Ozguven、Ozbakir 與 Yavuz（2010）提出的求解一致。第三階段建立 6 件工作，每件工作至多包含 5 項作業，總共有 6 部機器的大規模數學模型求解。

如第四章的表 4.7、中規模多處理器計算結果與表 4.10、大規模多處理器計算結果所顯示，求解的排程問題之規模從 5 件工作，4 項作業，5 部機器擴大至 6 件工作，5 項作業，6 部機器，單處理器的規劃時間從 31569 秒增加到超過 86400 秒（於規劃時限內仍未求得最佳解），說明複雜的排程問題，求解時間隨問題規模的增大，求解時間程指數成長。因此，本研究進一步透過平行計算技術與數學規劃軟體的加速求解機制，進行多處理器求解。

經由多處理器平行計算與單一處理器進行全域求解的比較，實驗結果顯示，相較於單一處理器計算求解，中規模透過兩個處理器之平行計算，節省近 44%的規劃時間，而透過 32 個處理器之平行計算，可以節省約 90%的規劃時間；大規模的部分，儘管單一處理器進行全域求解與 2 個處理器之平行計算皆無法在規劃時限內求解完成，但 8 個處理器、16 個處理器、

32 個處理器與 64 個處理器之平行算結果，能顯示出經由平行計算所提升的效率。其中大規模的排程規劃，相較於 4 個處理器之平行計算，8 個處理器之平行計算能節省約 46%的規劃時間，32 個處理器之平行計算節省約 57%的規劃時間，而 64 個處理器之平行計算節省近 70%的規劃時間。

經本研究實驗顯示，其所需規劃時間會隨著排程問題規模擴大而呈現指數成長增加的現象，透過分散式平行計算，提升求解複雜問題的效率，降低排程規劃所需時間，使對於複雜排程問題求解最佳化能因為求解時間的下降，更貼近實務上的應用。

5.2 未來發展方向

在本研究所採用的數學模型與本研究之假設前提，為針對特定零工式生產的排程問題，加上實際現場排程特性與許多無法控制的因子，故未來可依照不同的環境條件，修改數學模型，例如將交期、批量生產、良率、設備維護等因子納入考量，提供更多元與彈性的規劃生產排程之數學模型。

對於分散式平行計算，軟體使用的部分，除了 LINGO 軟體外，未來可以考慮採用其它支援 Java API 的數理軟體進行計算；資料分割的部分，則可以採用與本研究不同的分割方式進行計算，分析計算時間的可行性與規劃結果之合理性。

參考文獻

王治平 (民 102)。實際零工式生產排程問題的派工法則。 國立政治大學，台北市。

林詩彥 (民 100)。分散式平行計算應用於以製造廠為中心之供應鏈網絡規劃。工業工程與經營資訊研究所東海大學碩士論文。

陳建良 (民 84)。排程概述。*機械工業雜誌。* 122-137。

黃貞翔 (民 101)。供應網絡之總體生產規劃－以專業電子代工產業為例。工業工程與經營資訊研究所東海大學碩士論文。

葉斯暢 (民 99)。分散式平行系統應用於供應網絡網絡規劃-以記憶體模組產業為例。工業工程與經營資訊研究所東海大學碩士論文。

Beck, J. C., & Fox, M. S. (2000). Constraint-directed techniques for scheduling alternative activities. *Artificial Intelligence, 121*(1–2), 211-250. Bell, G. (1992). A teraflop before its time. Communications of the ACM.

Brucker, P. (2007). The Job-Shop Problem： Old and New Challenges.

DauzerePeres, S., & Paulli, J. (1997). An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research, 70*, 281-306. doi： 10.1023/a：1018930406487

Flynn, M. J. (1972). SOME COMPUTER ORGANIZATIONS AND THEIR EFFECTIVENESS. *IEEE Transactions on Computers, C 21*(9), 948-960.

Foster, I. (1995). *Designing and Building Parallel Programs：Concepts and Tools for Parallel Software Engineering*： Addison-Wesley.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. G. R. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling： a Survey. In E. L. J. a. B. H. K. P.L. Hammer (Ed.), *Annals of Discrete Mathematics* (Vol. Volume 5, pp. 287-326)： Elsevier.

Hertz, A., & Widmer, M. (1996). An improved tabu search approach for solving the job shop scheduling problem with tooling constraints. *Discrete Applied Mathematics, 65*(1–3), 319-345.

Kai Hwang, F. A. B. (1984). *Computer architecture and parallel processing*： McGraw-Hill.

Lee, C. Y., Lei, L., & Pinedo, M. (1997). Current trends in deterministic scheduling. *Annals of Operations Research, 70*, 1-41.

Loukil, T., Teghem, J., & Fortemps, P. (2007). A multi-objective production scheduling case study solved by simulated annealing. *European Journal of Operational Research, 179*(3), 709-722.

Manne, A. S. (1960). On the Job-Shop Scheduling Problem. *Operations Research, 8*(2), 219-223.

Mouret, S., Grossmann, I. E., & Pestiaux, P. (2011). Time representations and mathematical models for process scheduling problems. *Computers & Chemical Engineering, 35*(6),

1038-1063.

Neto, R. F. T., & Godinho, M. (2011). An ant colony optimization approach to a permutational flowshop scheduling problem with outsourcing allowed. *Computers & Operations Research, 38*(9), 1286-1293.

Ozguven, C., Ozbakir, L., & Yavuz, Y. (2010). Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Applied Mathematical Modelling, 34*(6), 1539-1548.

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A genetic algorithm for the Flexible Job-shop Scheduling Problem. *Computers & Operations Research, 35*(10), 3202-3212.

Pinedo, M. (2012). *Scheduling ：theory, algorithms, and systems.*

Schrich, C. R., Armentano, V. A., & Laguna, M. (2004). Tardiness minimization in a flexible job shop： A tabu search approach. *Journal of Intelligent Manufacturing, 15*(1), 103-115.

Steinhofel, K., Albrecht, A., & Wong, C. K. (1999). Two simulated annealing-based heuristics for the job shop scheduling problem. *European Journal of Operational Research, 118*(3), 524-548.

Tavakkoli-Moghaddam, R., & Daneshmand-Mehr, M. (2005). A computer simulation model for job shop scheduling problems minimizing makespan. *Computers & Industrial Engineering, 48*(4), 811-823.

附錄

```
SETS:
job / i1,i2/;
operation / j1/;
job2 /m1,m2/;
operation2 /n1/;
machine / k1,k2/;
process / p1,p2/;
process2 /q1,q2/;
time(job, process,operation,machine ): t, X, S, C, Z;
time2(job2,process2,operation2,machine):S2,C2;
new(job,process,operation,job2,process2,operation2,machine):Y;
ENDSETS

DATA:
t=
 4 5   7 5
 6 3   4 4
;
L=
9999999999;
ENDDATA

MIN = AAA;
AAA= @MAX(time(i,p,j,k):C(i,p,j,k));
@SUM(time(i,p,j,k)|i#eq#1#and#j#eq#1: X(i,p,j,k)-Z(i,p,j,k))=0;
@SUM(time(i,p,j,k)|i#eq#2#and#j#eq#1: X(i,p,j,k)-Z(i,p,j,k))=0;
@FOR(time(i,p,j,k): S(i,p,j,k)+C(i,p,j,k)<=X(i,p,j,k)*L);
@FOR(time(i,p,j,k):C(i,p,j,k)>=S(i,p,j,k)+t(i,p,j,k)-(1-X(i,p,j,k))*L);
@FOR(time(i,p,j,k):@BIN(X));
@FOR(new(i,p,j,m,q,n,k):@BIN(Y));
@FOR(new(i,p,j,m,q,n,k):@BIN(Z));
@FOR(time(i,p,j,k):S(i,p,j,k)>=0);
@FOR(time(i,p,j,k):C(i,p,j,k)>=0);
@FOR(time(i,p,j,k):S2(i,p,j,k)>=0);
@FOR(time(i,p,j,k):C2(i,p,j,k)>=0);
@FOR(time(i,p,j,k):@GIN(S));
@FOR(time(i,p,j,k):@GIN(C));
@FOR(time(i,p,j,k):@GIN(S2));
@FOR(time(i,p,j,k):@GIN(C2));
@FOR(time(i,p,j,k):@GIN(AAA));
@FOR(new(i,p,j,m,q,n,k)|i#eq#1#and#p#eq#1#and#j#eq#1#and#m#eq#2#and#q#eq#1#and#n#eq#1:
@SUM(new(i,p,j,m,q,n,k):S(i,p,j,k))>=@SUM(new(i,p,j,m,q,n,k):C2(m,q,n,k))-Y(i,p,j,m,q,n,k)*L);
@FOR(new(i,p,j,m,q,n,k)|i#eq#1#and#p#eq#1#and#j#eq#1#and#m#eq#2#and#q#eq#2#and#n#eq#1:
@SUM(new(i,p,j,m,q,n,k):S(i,p,j,k))>=@SUM(new(i,p,j,m,q,n,k):C2(m,q,n,k))-Y(i,p,j,m,q,n,k)*L);
@FOR(new(i,p,j,m,q,n,k)|i#eq#1#and#p#eq#2#and#j#eq#1#and#m#eq#2#and#q#eq#1#and#n#eq#1:
@SUM(new(i,p,j,m,q,n,k):S(i,p,j,k))>=@SUM(new(i,p,j,m,q,n,k):C2(m,q,n,k))-Y(i,p,j,m,q,n,k)*L);
@FOR(new(i,p,j,m,q,n,k)|i#eq#1#and#p#eq#2#and#j#eq#1#and#m#eq#2#and#q#eq#2#and#n#eq#1:
@SUM(new(i,p,j,m,q,n,k):S(i,p,j,k))>=@SUM(new(i,p,j,m,q,n,k):C2(m,q,n,k))-Y(i,p,j,m,q,n,k)*L);
@FOR(new(i,p,j,m,q,n,k)|i#eq#m#AND#j#eq#n#AND#p#eq#q:C(i,p,j,k)=C2(m,q,n,k));
@FOR(new(i,p,j,m,q,n,k)|i#eq#m#AND#j#eq#n#AND#p#eq#q:S(i,p,j,k)=S2(m,q,n,k));
@SUM(time(i,p,j,k)|i#eq#1#and#j#eq#1: Z(i,p,j,k))=1;
@SUM(time(i,p,j,k)|i#eq#2#and#j#eq#1: Z(i,p,j,k))=1;
```

附圖 1 小規模排程問題之 Lingo 程式碼