東　　海　　大　　學

工業工程與經營資訊研究所

碩士論文

分散式平行系統應用於供應鏈網絡規劃
— 以記憶體模組產業為例

研 究 生：葉斯暢

指導教授：黃欽印 博士

陳武林 博士

中 華 民 國 九 十 九 年 六 月

Distributed parallel system applied to supply chain network planning - A Case of Memory Module Industry

By

Sz-Chang Yeh

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

June 2010

Taichung , Taiwan , Republic of China

分散式平行系統應用於供應鏈網絡規畫-以記憶體模組產業為例

研究生：葉斯暢　　指導教授： 黃欽印 博士
陳武林 博士

東海大學工業工程與經營資訊研究所

摘　要

隨著全球化市場的競爭壓力下，許多企業於各地區設有配銷中心及製造廠，傳統中企業僅追求本身的淨利極大化的目標已嫌不足，已經轉變為整體供應鏈的總淨利最大化，所面臨的生產活動，亦從過去的單階單廠區生產演變成為供應鏈中多階多廠區生產模式。因此，在目前的許多探討議題中，有關供應鏈已聚焦在整體實務面的操作，藉由供應鏈全面性的考量以創造整體價值最大化來取代傳統的目標。

過去的文獻中，經常提及排程規劃為 *NP-Complete* 的問題，因此，本研究提出符合記憶體模組產業的供應網絡生產規劃模式，考慮物料供給限制、原物料採購價格、各廠區產能限制、生產及運輸前置時間等特性，於多階多廠供應網絡環境中，建構出一個非線性整數規劃 (Integer Non-Linear Programming, INLP)的數學模型，並配合 LINGO 10.0 extended 求解多限制式，多目標的排程問題。

針對數學模型的 *NP-Complete* 問題，本研究建構出分散式平行系統進行數值運算，而在分散式平行計算的技術方面，採用 Java 遠程方法調用（Java Remote Method Invocation, JavaRMI）和推演出來的演算邏輯來實現高速平行計算，達到計算時間上的改善。經由實驗結果顯示，本研究方法相較於數學模型求解的作法，可在短時間內達到最佳的規劃結果，所求得的結果可以讓規劃人員在短時間內做出決策。

關鍵字詞：多廠區生產規劃、記憶體模組、非線性整數規劃、平行計算、Java 遠程方法調用

Distributed parallel systems applied to supply chain network planning - A Case of Memory Module Industry

Student: Sz-Chang Yeh Advisor: Prof. Chin-Yin Huang
Prof. Wu-Lin Chen

Department of Industrial Engineering and Enterprise Information
Tunghai University

ABSTRACT

With competive pressure of globalization, many companies have distribution centers and manufacturing plants at various locations. Tranditional enterprises only persue their goal of profit maximization had been not enough. They have been transformed into the whole supply chain to maximize profit. The production activities has also altered from the previous single-stage and single-plant production into a multi-stage supply chain and multi-plant production. Therefore, many of the current issues in the supply chain have been focused on the whole practical side. Through a comprehensive consideration of the supply chain to maximize the overall value to replace the traditional goal.

The past literature often referred to schedule planning is a NP-Complete problem. Therefore, this study found the memory module industry supply network planning model of production, considering material supply constraints, the purchase price of raw materials, the plant capacity constraints, transportation lead time, the multi-stage, multi-factory supply network environment and other features. This paper has to propose a non-linear integer programming mathematical model and uses LINGO 10.0 extended to solve multi-constraint and multi-objective scheduling problem.

For NP-Complete problem of Mathematical model, this study constructs a distributed parallel system for numerical computing. In distributed parallel computing technologies, using Java Remote Method Invocation and the deduced algorithm to achieve high-speed parallel computing and improve computing time. The experimental results show that our method compared with solving mathematical model can achieve the good planning results in a short time. The results obtained allow planners to make decisions in a short time.

Keywords: multi-plant production planning、memory module industry、non-linear integer programming、parallel computing、Java Remote Method Invocation

誌　謝

研究所兩年的時間一眨眼就畫上了句點，在這兩年的期間，不論是有關與課內或是課外的學習，都讓我成長了許多，在此我要先感謝黃欽印老師，老師對於我的教導不是僅限於課內的學習，甚至教導我許多的人生態度，以及在我求學過程中心情最低落的時候，老師對我的關懷及鼓勵是讓我重新站起來的動力；在課內的學習部分，老師不斷的要求我學習各種技能，如此才能讓我今天能應用各種不同的技術，完成如此艱辛的研究。如今，我即將邁向人生的另一個旅程，相信在未的日子裡，我會以東海工業工程這個大家庭為榮。

此篇論文的完成，主要要感謝黃欽印老師以及陳武林老師，在每週的報告時，總是用心的指導我們，引導我們學習到許多研究的方法與思考的方向，使我能逐步的完成這份論文。而在研究學習的過程中，讓我不僅僅學到了解決問題的方法也慢慢訓練自己邏輯思考及口頭報告的能力，不論是課業或做人處事態度上，師長們總是不厭其煩的叮嚀與囑咐，讓我能夠持續地進步往前。

兩年的研究所生活中，承蒙多位系上老師以及振辰、宏霖、弼仁、靜怡等學長姐的帶領及指教，還有靖雅、文冠、怡芳和世倫等研究室夥伴們的互相討論及勉勵，玠孝、容慈及詩彥等學弟妹們給予我的幫助與支持，使我在研究的過程中能夠事半功倍。

最後特別要感謝我最愛的家人，謝謝你們在我廢寢忘食的時候，從沒忘記替我加油打氣，還有最親愛的朋友們，總是陪我一起開心一起難過，在我最灰心失落的時候給予我最大的力量，如果沒有你們的關心與陪伴，讓我能一心一意的專注於學業上，就無法成就今天的我，謝謝你們。

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。

葉斯暢　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

中華民國九十九年六月

目　錄

摘　要 ..... **I**

ABSTRACT ..... **II**

目　錄 ..... **IV**

圖目錄 ..... **VI**

表目錄 ..... **VIII**

第一章 緒論 **1**

1.1 研究背景 1

1.2 研究動機 4

1.3 研究目的 6

第二章 文獻探討 **7**

2.1 記憶體模組產業介紹 7

2.1.1 記憶體模組產業現況 7

2.1.2 產業鏈關係 8

2.1.3 記憶體模組製程 9

2.1.4 記憶體模組產業特性 10

2.2 供應鏈網絡生產規劃的定義與特性 10

2.3 供應網絡生產規劃相關研究 14

2.4 平行計算 19

2.4.1 平行計算系統之位址空間架構 19

2.4.2 平行計算系統之控制機制 22

2.4.3 平行計算系統之訊息傳遞模式 23

2.5 分散式系統 24

2.5.1 JavaRMI 25

2.5.2 分散式物件系統之工業標準(CORBA) 29

第三章 供應網絡生產規劃模式 **31**

3.1 記憶體模組產業供應網絡生產規劃現況方法 32

3.2 供應網絡之數學模型 35

3.2.1 以視覺化圖形描述供應網絡 36

3.3 分散式平行系統之架構 46

3.3.1 循序程式的平行化 47

3.3.2 平行計算之演算法 54

第四章 實驗設計 **72**

4.1 實驗方式與環境建構.....72
4.1.1 實驗環境.....72
4.1.2 實驗因子及數學模型的參數設定.....73
4.2 實驗設計及分析.....77
4.2.1 小規模數學模型的全域最佳解.....77
4.2.2 小規模之平行計算的最佳規劃結果.....80
4.2.3 小規模之平行計算與全域最佳解比較.....83
4.2.4 大規模之平行計算的最佳規劃結果.....87
第五章 結論與未來發展方向.....**95**
5.1 結論.....95
5.2 未來發展方向.....96
參考文獻.....**97**

圖目錄

圖 1.1、2006 年記憶體模組銷售金額分區統計(以各公司所在區域分類).....1
圖 1.2、記憶體模組產業供應網絡架構圖 3
圖 2.1、記憶體模組之產業鏈 9
圖 2.2、記憶體模組製程簡介 9
圖 2.3、供應鏈架構之示意圖 10
圖 2.4、一般性協同及多廠區協同示意圖 13
圖 2.5、多階多廠示意圖 13
圖 2.6、有向圖形描述供應網絡 16
圖 2.7、多種品項、多廠區及多個時期之供應網絡圖 17
圖 2.8、共享式記憶體多處理器系統架構圖示意圖 20
圖 2.9、分散式記憶體多處理器系統架構示意圖 21
圖 2.10、MPMD架構圖.....23
圖 2.11、SPMD架構圖 23
圖 2.12、採用JAVARMI建構的分散式物件系統 26
圖 2.13、JAVARMI的運作架構 27
圖 2.14、基於JAVARMI分散式物件系統的架構 28
圖 2.15、利用SOCKET傳送實數 28
圖 2.16、利用JAVARMI傳送實數(呼叫方法).....29
圖 3.1、記憶體模組廠供應網絡架構 31
圖 3.2、記憶體模組廠之現行供應網絡生產規劃方法 33
圖 3.3、視覺化圖形定義說明 37
圖 3.4、案例：供應網絡生產規劃模式示意圖-以視覺化圖形描述 38
圖 3.5、資料的平行處理 47
圖 3.6、分散式平行系統之架構示意圖 48
圖 3.7、伺服端的計算流程 50
圖 3.8、計算節點端的計算流程 52
圖 3.9、LINGO計算過程中的上界和下界 54
圖 3.10、數學模型之集合的概念 55
圖 3.11、子集合的最佳規劃結果之目標值來控制上界.....56
圖 3.12、伺服端的計算流程(子集合之最佳規劃結果控制上界).....58
圖 3.13、計算節點端的計算流程(子集合之最佳規劃結果控制上界).....60
圖 3.14、平行計算中得到的更佳目標值來控制上界 62
圖 3.15、伺服端的計算流程(以更佳的目標值控制上界).....64
圖 3.16、計算節點端的計算流程(以更佳的目標值控制上界).....67
圖 3.17、演算法之決策流程圖 69

圖 4.1、多階多廠生產環境 72
圖 4.2、小規模 30%需求水準的全域最佳規劃結果 77
圖 4.3、小規模 60%需求水準的全域最佳規劃結果 78
圖 4.4、小規模 80%需求水準的全域最佳規劃結果 79
圖 4.5、小規模平行計算之處理器數目和計算時間的比較 82
圖 4.6、小規模 30%需求水準之處理器數目和計算時間的比較 83
圖 4.7、小規模 60%需求水準之處理器數目和計算時間的比較 84
圖 4.8、小規模 80%需求水準之處理器數目和計算時間的比較 85
圖 4.9、小規模之處理器數目和計算時間的比較 86
圖 4.10、大規模 30%需求水準的處理器數目和計算時間的比較 88
圖 4.11、大規模 60%需求水準的處理器數目和計算時間的比較.....89
圖 4.12、大規模 80%需求水準的處理器數目和計算時間的比較 90
圖 4.13、大規模平行計算之處理器數目和計算時間的比較 91
圖 4.14、大規模 30%需求水準之全域規劃結果 91
圖 4.15、大規模 60%需求水準之全域規劃結果 92
圖 4.16、大規模 80%需求水準之全域規劃結果 92

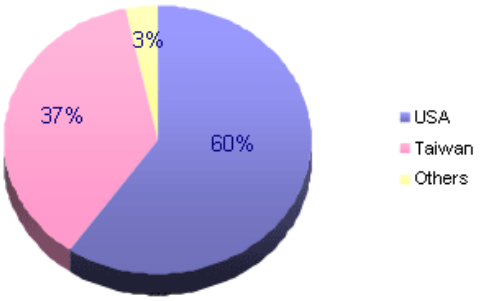
表目錄

表 1.1、2006 年自有品牌記憶體模組銷售排行 2
表 2.1、供應鏈體系層級圖 12
表 2.2、多廠區生產規畫相關研究 (本研究整理) 18
表 2.3、共享式記憶體平行電腦 v.s分散式記憶體平行電腦 21
表 2.4、CORBA可支援的程式設計語言 30
表 3.1、現況方法與本研究模式之供應網絡比較表 35
表 3.2、平行計算的結果 53
表 4.1、控制因子 73
表 4.2、數學模型之參數設定(大規模) 74
表 4.3、數學模型之參數設定(小規模) 75
表 4.4、演算法之決策流程的參數設定 76
表 4.5、小規模之環境因子組合表 80
表 4.6、環境組合A到C之規劃結果 80
表 4.7、環境組合D到F之規劃結果 81
表 4.8、環境組合G到I之規劃結果 81
表 4.9、小規模 30%需求水準之總淨利和計算時間的比較 83
表 4.10、小規模 60%需求水準之總淨利和計算時間的比較 84
表 4.11、小規模 80%需求水準之總淨利和計算時間的比較 85
表 4.12、大規模之環境因子組合表 87
表 4.13、環境組合J到L之規劃結果 88
表 4.14、環境組合M到O之規劃結果 89
表 4.15、環境組合P到R之規劃結果 90
表 4.16、大規模 30%需求水準之總淨利和計算時間的比較 93
表 4.17、大規模 60%需求水準之總淨利和計算時間的比較 93
表 4.18、大規模 80%需求水準之總淨利和計算時間的比較 93

第一章 緒論

1.1 研究背景

近幾年來，記憶體模組(Memory Module)產業是現今電子產業中面臨景氣波動衝擊影響甚鉅的產業之一，緣自各種消費性電子產品的引用，記憶體產品的應用近年來快速走向多元化，結合消費性電子產品的產業特性，產品生的命週期趨向更為短促。供給與需求面的失衡經常引起市場價格的劇烈變動，企業庫存價值損失的風險相對提升。記憶體模組產業在台灣扮演著重要的角色，根據集邦(DRAMeXchange)研究資料顯示，2006 年自有品牌記憶體模組全球獲利達 87 億 6030 萬美元，其中，全球排名前 20 名有 9 間為台灣的企業，其市佔率總和約為 27%，如表 1.1 所示；除此之外，以自有品牌記憶體模組銷售金額分區統計(以各公司所在區域分類)而言，台灣則佔全球總銷售金額將近 40%，顯然地台灣已成為全球記憶體模組的生產重鎮，如圖 1.1 所示。



Source: DRAMeXchange

圖 1.1、2006 年記憶體模組銷售金額分區統計(以各公司所在區域分類)

表 1.1、2006 年自有品牌記憶體模組銷售排行

公司名	地區	2006 排行	2006 市占率(%)	2006 自有品牌模組營收(百萬美元)	2005 排行
Kingston Technology 金士頓	美國	1	20.55	$1800.0	1
A-Data Technology 威剛	台灣	2	7.21	$631.4	2
Smart Modular Technologies 美商世邁	美國	3	5.50	$482.0	3
Trancend Information 創建	台灣	4	4.02	$352.0	9
MA Labs	美國	5	3.82	$335.0	6
Apacer Technology 宇瞻	台灣	6	3.46	$303.0	14
Crucial Technology	美國	7	3.33	$292.0	5
Corsair Memory 海盜船	美國	8	3.25	$285.0	8
TwinMOS Technologies 勤茂	台灣	9	3.22	$282.0	12
Geil 友懋	台灣	10	3.10	$272.0	7
PNY Technologies 美商必恩威	美國	11	2.17	$190.0	4
Patriot Memory 美商博帝	美國	12	1.93	$169.0	19
PQI 勁永	台灣	13	1.86	$163.0	17
Wintec Industries 資料國際	美國	14	1.77	$155.0	13
Veritech 橋集	台灣	15	1.74	$152.0	23
Melco 巴比祿	日本	16	1.63	$143.0	11
Viking Interworks 維京	美國	17	1.56	$137.0	16
Kreton 拜通	台灣	18	1.49	$130.3	22
Kingmax Semiconductor 勝創	台灣	19	0.92	$81.0	10
Golden Mars 晶芯	香港	20	0.86	$75.0	na
Others 其他			20.60	$2330.6	

資料來源：DRAMeXchange

台灣大部分記憶體模組業者幾乎沒有一家從事記憶體晶片(DRAM/Flash)的生產，單純的由記憶體生產廠商取得記憶體零件，自行或外包(Outsoucing)加工組裝，然後銷售到通路商和系統組裝廠。在面對如此大量的全球需求，大部分記憶體模組廠會在各地設置進行加工的製造廠及配銷中心，以增加產能而滿足需求。然而，就目前記憶體模組產業而言，如何在規格標準化的產品來自少數幾家供應商以及廠區的增加，和記憶體晶片價格劇烈波動以及記憶體模組最終成品沒有明顯差異下，如何整合上下游多廠區以進行生產規劃及追求營運成長，以有效協調各廠區的供給及面對全球的需求，進而達到局部競爭優勢是共通的問題。

就記憶體模組產業而言，其供應鏈結構是由供應商、製造廠及配銷中心所組成，是謂“多階層”的環境，且每個階層都有超過一個以上的廠區，

形成“多廠區”的環境，而“多階層”及“多廠區”的結合，便形成“供應網絡”的生產環境，如圖 1.2 所示。

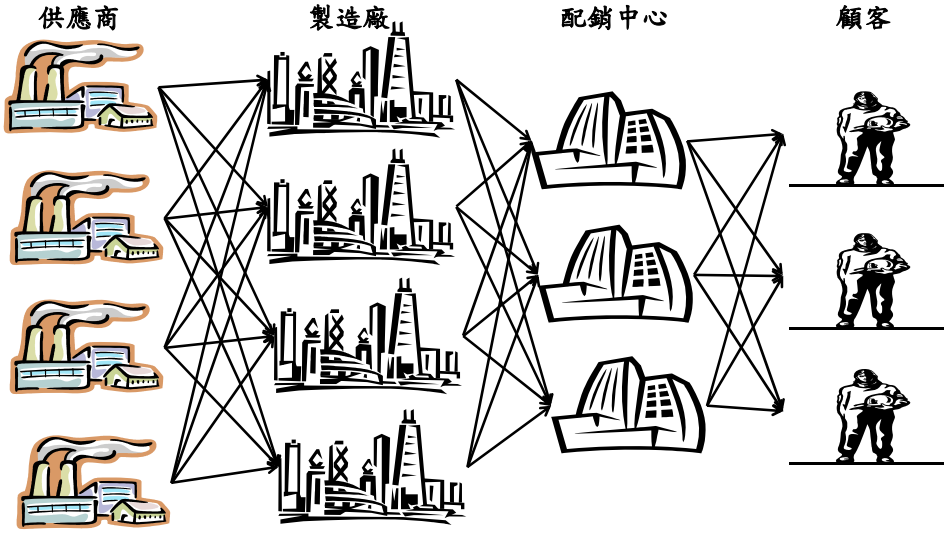


圖 1.2、記憶體模組產業供應網絡架構圖

除了供應網絡的生產環境，記憶體模組產業有其生產規劃上的特性。舉例而言，關鍵原料動態隨機存取記憶體(Dynamic Random Access Memory, DRAM)顆粒價格波動劇烈，營運波動幅度大－動態隨機存取記憶體顆粒的採購成本佔總成本的比例非常高，其他為專用集成電路芯片(ASIC IC)與印刷電路板(PCB)與加工費用。由於關鍵原料動態隨機存取記憶體顆粒價格波動劇烈，因此有效掌握動態隨機存取記憶體顆粒供應來源及控制庫存水準為立足記憶體模組產業的關鍵要素。

若記憶體模組廠商與記憶體供應商之間的合作不順利，可能在供不應求、價格飛漲時，無法取得貨源；或在供過於求、價格急跌時，卻有相當的庫存無法消化。這都會造成相當大的損失。另外，記憶體模組本身是屬於高單價，買賣金額龐大，現金周轉需求高，但功能性差異不大，若企業無法因應這些外在環境的話，是無法在數家廠商中脫穎而出。因為，下列三項的金額往往超過百萬美金：

1. 已下訂單給供應商，但尚未交貨的在途零件
2. 尚未完成組裝的半成品及零件
3. 已接客戶訂單但尚未交貨給客戶

因此在進行生產規劃時，必須要考量運輸及生產的前置時間及其可能

造成的存貨成本之影響。此外，由於規劃時間內各廠區的各項成本不盡相同，當配銷中心有需求存在時，如何在有限的規劃時間、供給數量及產能下，分配訂單至各個製造商以達到最佳規劃結果，亦是記憶體模組產業需考量的重點。

總體而言，記憶體模組產業的供應網絡生產規劃必須考量下列各點：

1. 多階多廠供應網絡的生產規劃環境
2. 各廠區的各項成本
3. 考量原物料供給數量的限制
4. 各廠區的產能限制
5. 運輸及生產前置時間

1.2 研究動機

在觀察業界現行供應網絡生產規劃方法得知，其以啟發式演算法考量運輸前置時間及各品項之數量，逐筆訂單進行規劃，決定各訂單由哪些製造廠及哪些品項生產而成，此方法雖然容易，卻存許多問題，如下所示：

1. 在現行規劃模式下，規劃人員根據優先順序逐筆訂單進行規劃，無法達到整體的最佳規劃結果。
2. 就廠區優先順序而言，僅依據運輸前置時間決定各廠區進行生產之優先順序，並未考量產能限制，因此，可能因產能不足而造成訂單延遲或缺貨的情況。
3. 在現行規劃模式下，並未考量各項成本所造成的影響，因此規劃的結果並非最佳規劃結果。
4. 由現行的啟發式演算法規劃的結果，並無考慮廠區之間運輸及生產前置時間所造成的庫存影響，因此規劃結果往往會造成各廠區有大量的庫存數量。

針對供應網絡規劃的問題，過去文獻亦有大量的研究，其相關主題所用的建模方法大致上可分數學模式、模擬方式及啟發式演算法。然而這些方法通常亦存在著以下的問題：

1. 單純採用數學模式建模之研究而言，變數之數量會隨著規模大小(品項種類、廠區數量及需求數量等)成指數增加，當規模較大時，其規劃求解時間過長，在實務上應用不便。

2. 以模擬方式及啟發式演算法研究而言，往往為部份最佳化，而非整體的最佳化，尤其在記憶體模組產業，此種規劃方法無法取得整體最佳之分配。
3. 過去研究大多未針對記憶體模組產業之供應網絡生產規劃進行探討，且無同時考量記憶體模組產業之主要生產特性，如下所示：
   (1) 多階多廠供應網絡的生產規劃環境
   (2) 原物料供給數量的限制

Wu (2004)提出以有向圖形建立供應網絡環境，將不同的廠區、不同的品項及不同的時期都區分為不同的節點，再進行生產規劃以滿足各時期的需求節點，表達多種品項、多廠區及多個時期之供應網絡生產環境。然而，作者證明在此模式下，即使為單一品項、多廠區、多時期、無限產能且無運輸成本的生產規劃問題下，將會是個 *NP-Complete* 的問題。

過去的研究針對供應鏈網絡的 *NP-Complete* 的問題提出了許多的求解方法，但對於利用平行計算以達到最佳規劃結果的研究卻非常的稀少。因此，在如此之供應網絡環境下，如何考量記憶體模組產業的生產特性，建構一套適用之供應網絡的平行計算系統，能在合理規劃時間內，利用有限之供應鏈資源限制，求得最佳規劃結果以供規劃人員參考，是值得研究之議題。

1.3 研究目的

基於上述背景與動機，本研究將針對記憶體模組產業進行分析，提出符合記憶體模組產業生產規劃特性之供應網絡生產規劃模式。具體而言，本研究主要目的如下：

1. 歸納記憶體模組廠多階多廠供應網絡生產規劃模式的特性，提出符合記憶體模組廠供應網絡的數學模型。
2. 針對記憶體模組廠供應網絡之數學模型，利用分散式物件的技術和數理規劃工具建構出分散式平行系統進行數值計算。
3. 經由實驗設計，比較本模式的最佳規劃結果與小規模的全域最佳解之優劣，再運用本模式來針對大規模求解，在計算的過程中也比較處理器數目的多寡所造成計算時間上的差異。

第二章 文獻探討

本節分別針對記憶體模組產業、供應網絡生產規劃的定義與特性、供應網絡生產規劃相關研究、平行計算及分散式系統等五方面進行探討。

2.1 記憶體模組產業介紹

以下將針對記憶體模組產業進行介紹，分別就產業現況、產業鏈關係、記憶體模組製程及產業特性等部分說明。

2.1.1 記憶體模組產業現況

記憶體模組主要功能在個人電腦(PC)的處理器(CPU)處理訊息時，作為短暫性快速儲存讀取之記憶體，以提升電腦處理速度及記憶容量，是電腦不可或缺之重要零組件之一。一般而言，標準型記憶體模組主要應用於桌上型電腦及筆記型電腦，而隨資訊化程度提高，電腦處理速度加快，所需搭配的儲存容量亦與日俱增，且因自行組裝容易，故汰換率較高，全球市場規模大。目前國外記憶體模組大廠以KINGSTON（金士頓）全球市佔率約二成居冠，至於國內大廠有威剛（國內最大、全球第4大）、創見、勁永、勤茂、勝創等等。

記憶體模組廠係從事於記憶體模組及快閃記憶體(Flash Memory)應用產品之製造及買賣，其產品的主要關鍵零組件係為動態隨機存取記憶體及快閃記憶體等兩項記憶體元件。動態隨機存取記憶體是記憶體集成電路(IC)的一種，主要是用於電子系統產品之資料儲存或程式記憶，動態隨機存取記憶體為一個電晶體與一個電容器所構成之記憶體元件，DRAM Cell 將資料寫入後，儲存在電容器中，由於電容之電荷會隨時間增加而減少，因而有遺失資料之情形，故需要在一定時間內，對寫入的資料作更新的動作，同時切掉電源後資料會消失。快閃記憶體是非揮發性記憶體中最重要的技術，主要是用於電子系統產品之資料儲存或程式記憶，即使在無外部電源供電時，也能夠保存資訊內容，這使得裝置本身不需要浪費電力在資料的記憶上。目前的趨勢是朝加大容量、單一工作電壓與低電壓的方向發展，由於其具有體積小、容量高及便於攜帶的特性，而快閃記憶體與動態隨機存取記憶體兩者的最大差異處，就在於快閃記憶體具有非揮發的特性。

標準型動態隨機存取記憶體模組製程單純、流通性佳且不需投入龐大

之資本支出，因而吸引許多相關業者紛紛設立。然而，由於動態隨機存取記憶體模組價格暴漲暴跌，導致模組廠業績波動也頗大。除了產品售價隨動態隨機存取記憶體顆粒行情而波動外，一旦動態隨機存取記憶體模組行情重挫，模組廠必須承擔原先高進價的原料產生過高的採購成本及庫存積壓成本；相反的，當動態隨機存取記憶體模組行情高漲時，則可享受龐大的低價庫存利益。因此模組廠商唯有保持最佳的營運效率、以最低的成本、最快的生產效率，搭配良好的行銷通路才能在激烈的競爭環境中存活。

2.1.2 產業鏈關係

記憶體模組產業位於產業鏈的中游，如圖 2.1 所示。

1. 上游：記憶體模組廠之上游供應商包含動態隨機存取記憶體廠商、靜態隨機存取記憶體(SRAM)廠商、快閃記憶體廠商、印刷電路板廠商、晶片組(CHEPSET)廠商、連接器(CONNECTER)廠商及電子配件廠商等。
2. 中游：記憶體模組產業的中游為記憶體模組之製造及買賣廠商，主要是將記憶體、印刷電路板、晶片組、連接器及電子配件等原料加工製造成記憶體模組之電子元件成品，銷售予下游應用廠商、通路廠商及最終顧客。其中，原物料記憶體晶片佔記憶體模組產品成本比重較高，且採購價格會隨著時間而有所差異，直接影響到記憶體模組廠商之獲利狀況，此一市場情勢下，記憶體模組廠商面對及時性價格變動狀況，故在策略上以現貨價格為參考依據，規避價格波動風險。
3. 下游：記憶體模組產業下游為電子產品應用廠商，包括主機板廠商、繪圖卡廠商、筆記型電腦廠商、PC 組裝廠、經銷商、代理商、家電廠商、數位產品廠商、通訊產品廠商及消費性電子產品廠商等。

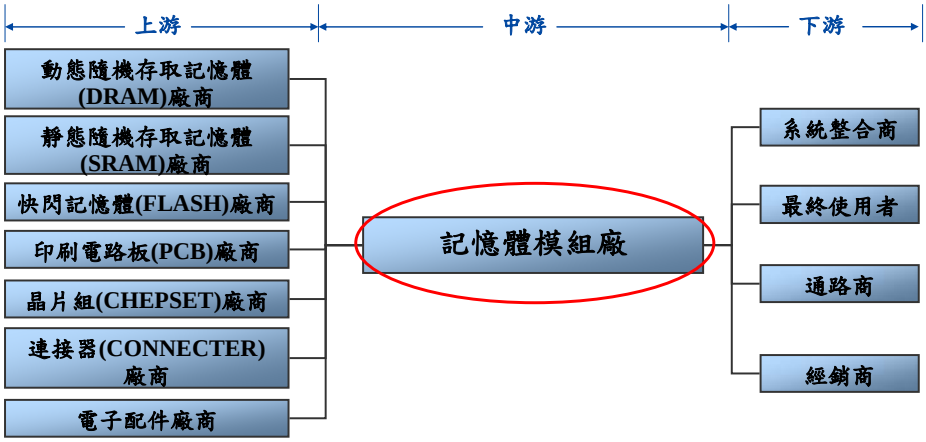


圖 2.1、記憶體模組之產業鏈

2.1.3 記憶體模組製程

就製程而言，記憶體模組是將可儲存並處理大量資料的記憶體顆粒，以表面黏著(Surface-mount technology, SMT)生產製程將記憶體顆粒鑲嵌於印刷電路板上，其製程如圖 2.2 所示。記憶體模組產業製程單純，依產品規格購入所需顆粒，加上所需的印刷電路版、晶片組、相關電子配件等，以表面黏著生產技術將各元件鑲嵌於印刷電路版上，再進行切割動作，即為我們在市面上所見的以條為單位的記憶體模組，經過打印以及檢驗後即可出貨。

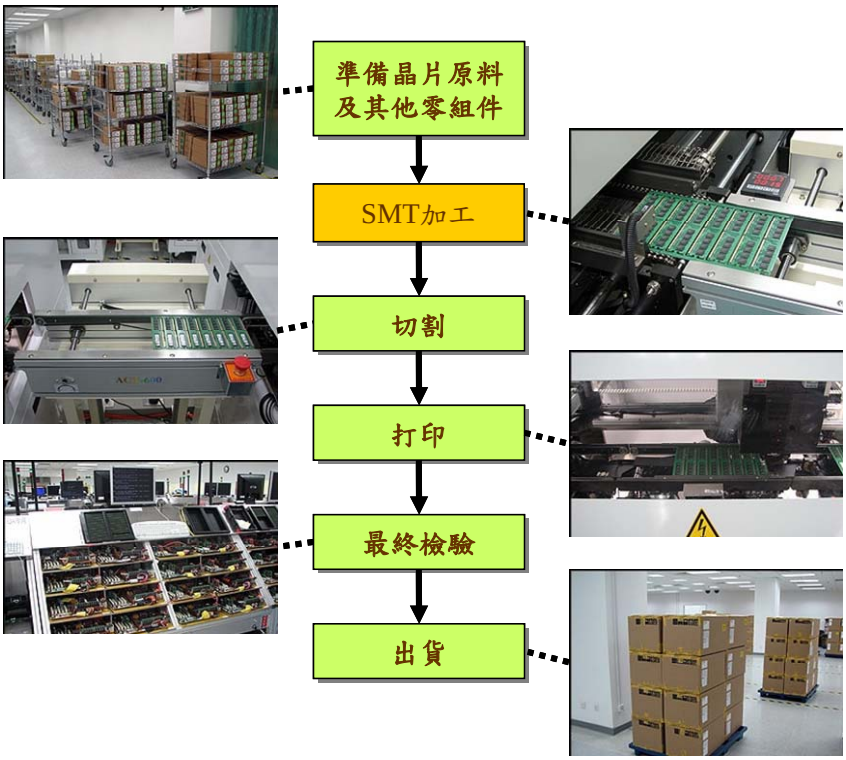


圖 2.2、記憶體模組製程簡介

2.1.4 記憶體模組產業特性

1. 多階多廠生產環境：記憶體模組在製程上並不複雜，然而，其供應鏈皆由多個製造廠與多個配銷中心所組成，因此規劃人員需要同時考慮到多階多廠區生產、產能限制、各項成本以及生產與運輸前置時間等，決定訂單的分配。
2. 關鍵原物料的採購：關鍵原物料記憶體晶片隨著不同的時間點而各家供應商有不同的採購價格，且全球市場中記憶體的供貨來源有限，不一定每次都能提供足夠數量，或是確切滿足中游記憶體模組廠所需型別之晶片顆粒，造成關鍵原料供給的不穩定性(包含價格及數量)。

因此，如何在規劃週期內，考量各個供應商原物料的採購數量及有效地分配訂單給各個製造廠，是記憶體模組產業供應網絡生產規劃的重點考量。

2.2 供應鏈網絡生產規劃的定義與特性

Santoso et al., (2005)說明了供應鏈即是一個由供應商、製造工廠、倉庫、配銷通路所構成的網路，以取得原物料、轉換成成品，並配銷到顧客手上。Stock and Lambert (2001)指出所謂的供應鏈管理，乃將最前端之消費者到最後端之供應商的各個關鍵商業流程整合在一起，提供產品、服務、資訊，為顧客和利害關係人增加附加價值，圖 2.3 為供應鏈架構之示意圖：

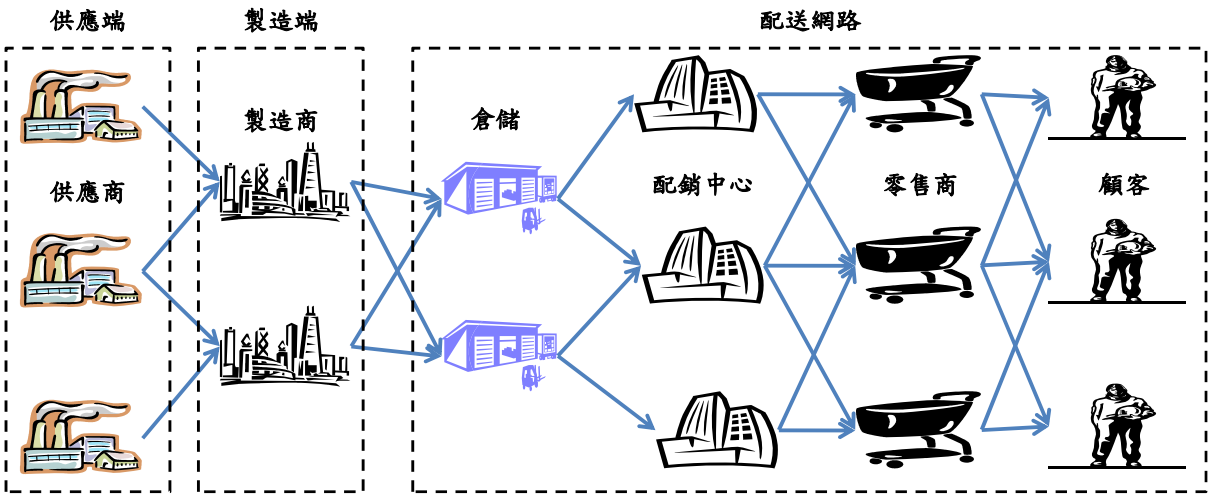


圖 2.3、供應鏈架構之示意圖

Shapiro (2001)指出供應鏈的架構可分為三個主要階層，包括：供應商(suppliers)、製造商或工廠(manufacturers/plants)、以及配送網路(distribution network)，以下將針對各項作簡單說明：

1. 供應商：供應商提供各項產品、原物料或服務給負責生產或組裝的廠商。此處的供應商通常是指直接供應商(immediate suppliers)，即直接將其原物料提供給廠商者。
2. 製造商或工廠：製造商或工廠主要是透過特定的生產程序，將上述供應商所提供的各項元素轉換成最終的產品或是服務，可包含製造廠商(fabrication plants)、次組裝廠(subassembly plants)以及最終組裝廠(final assembly plants)等等。
3. 配送網路：配送網路包含了配送中心(distribution centers)、零售商(retailers)以及終端顧客(end customers)等等，其主要工作是將已完成的產品或服務轉移到終端顧客的手中，以滿足其需求。

至於供應鏈架構中各階層之間的關係，則必須考慮各個企業所處的競爭位置為何。假使有某個企業之力量大到足以掌控整個供應鏈，則供應鏈中其他個體只能扮演跟隨者的角色，彼此間的關係也相對單純，這種供應鏈體系可視為集中式的供應鏈(centralized supply chain)。然而，由於現今的生產環境大多是由甲地供應原料、乙地生產及丙地組裝，甚至最後再交由另一個地方負責產品配銷，因此供應鏈中每個個體其實是以一種相依或互補的關係共存著，彼此間透過合作或聯盟，進行包括物流、金流、資訊流等生產運籌之整合，此種供應鏈體系可視為分散式的供應鏈(decentralized supply chain)。

Harland (1997)提出供應鏈體系概念，將其分為四個層級，分別為內部鏈(internal chain)、二重關係(dyadic relationship)、外部鏈(external chain)及網絡(network)，如表 2.1 所示：

表 2.1、供應鏈體系層級圖

層級	說明
內部鏈(internal chain)	此階層體系只擁有一個製造廠，屬於單廠的生產環境
二重關係(dyadic relationship)	此階層體系表示同一功能擁有多個製造廠，各個廠彼此間有支援及替代的作用
外部鏈(external chain)	此階層體系擁有多個不同功能的製造廠，是屬於多廠的製造環境，但每個廠間依據製造程序，有順序性的關係。(相同功能只有一個廠)
網絡(network)	此階層體系是由二重關係(dyadic relationship)及外部鏈(external chain)結合而組成，因此，其具有順序性及支援替代的特性。

因此，網絡(network)包含了二重關係(dyadic relationship)及外部鏈(external chain)的概念，在生產規劃勢必較其他層級要來的複雜。

關於供應鏈規劃問題的分類最早由 Bhatnagar (1993)等人提出來，他將協同規劃問題分為一般性協同(general coordination)與多廠區協同(multi-plant coordination)，如圖 2.4。一般性協同為協同整合供應鏈上下游不同功能間的決策，如採購、製造、配送等功能，包含決定設施位置、存貨規畫、生產計畫、配送行為和行銷策略等；而另一種多廠區協同則為組織內不同生產鏈階層間之同一功能的協同策略規劃之問題，同一功能是指均為製造生產功能，而多“廠”指的是多階層的“製造設施”，此類問題主要是探討垂直整合公司內許多製造工廠的單廠生產計畫之連結而成為一整合性生產計畫。

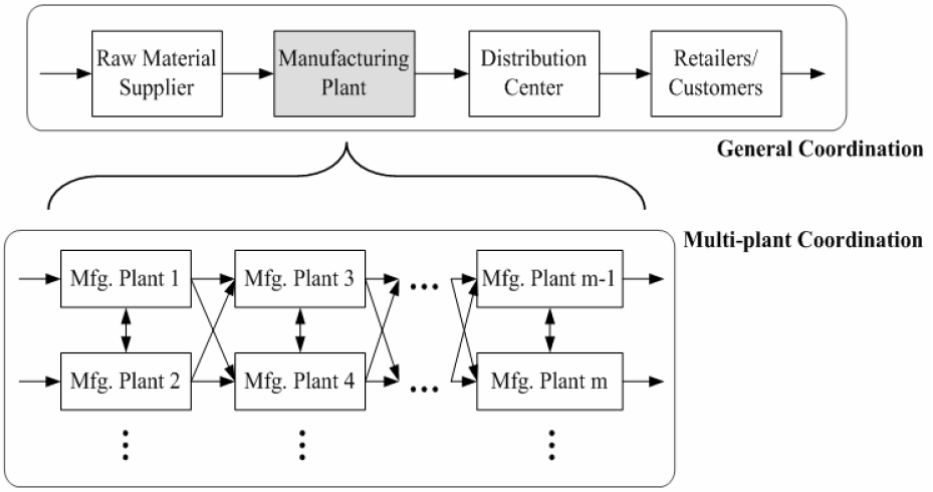


圖 2.4、一般性協同及多廠區協同示意圖

此外，Lin and Chen (2007)提出，由於顧客的需求日益遽增，企業必須透過擴廠或外包以增加產能來滿足需求，因此，製造環境已由傳統的單廠環境演化至現今的多廠製造，許多企業將面臨生產規劃上的困難。就目前製造程序而言，往往是由許多個步驟所組成，作者定義此狀況為“多階”，除此之外，在每一個階層，亦具有多個製造廠，作者定義每階層內多個製造廠為“多廠”，而“多階”及“多廠”的生產環境即建立為“供應網絡”生產環境。如圖 2.5 所示：

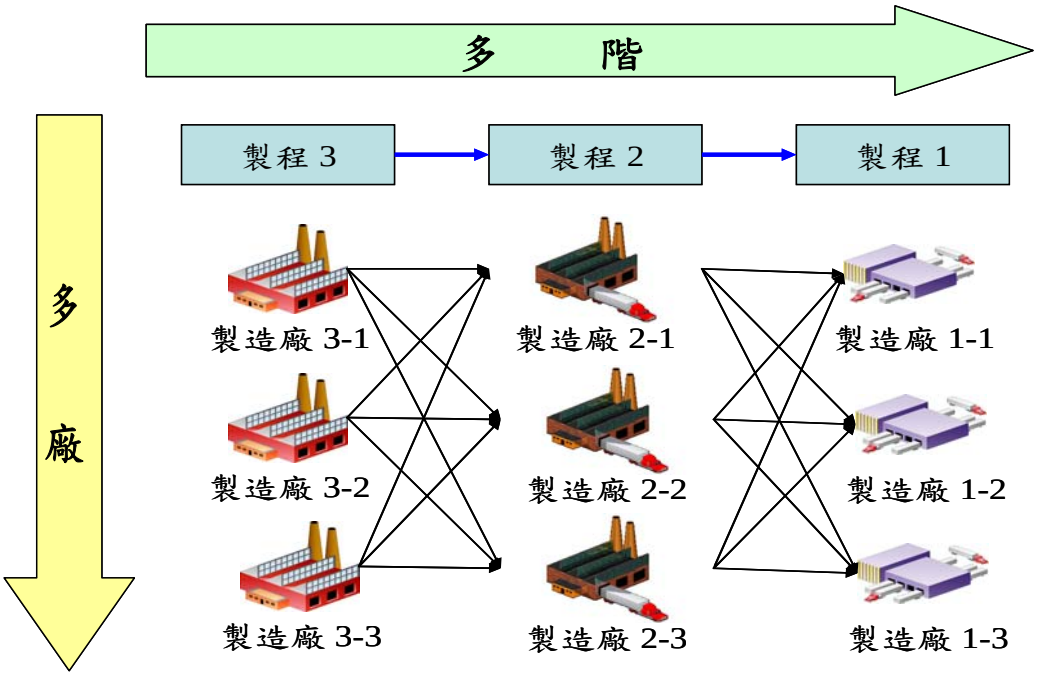


圖 2.5、多階多廠示意圖

此外，Samxsivan (2002)亦提出，多廠區規劃問題所規劃的主要內容與產出為(1)每個廠區所要生產的產品種類與數量，(2)每個廠區在各個規劃期末所需持有的存貨數量，(3)廠區與廠區之間的轉換數量。

在了解供應鏈網絡生產規劃的定義與特性後，接下來部分將進一步探討過去文獻對此領域之研究。

2.3 供應網絡生產規劃相關研究

近年來，有許多的研究利用各種技術(包含線性規劃、模擬、代理人或啟發式演算法等)解決供應網絡生產規劃之訂單分配問題。

Arntzen et al., (1995)提出一全球供應鏈規劃模式，利用混合整數線性規劃(Mixed-Integer Linear Programming, MILP)，考量多品項及其物料清單(Bill of Material, BOM)，整合製造廠及配銷中心間之規劃決策。Timpe and Kallrath (2000)以一化學工廠為研究對象，利用混合整數線性規劃建立模式，以訂單數量、工廠產能、物料流量、存貨及運送為其限制條件，在公司獲利最大目標下，探討供應鏈體系之生產規劃模式，求得工廠最適當的生產數量，主要著重在分配及協調各工廠間之生產數量。Guinet (2001)提出一啟發式演算法，考慮多品項於多廠區環境，並根據每種品項的物料清單決定訂單分配計畫。Moon et al., (2002)利用基因演算法(Genetic Algorithm)解決多廠區生產規劃問題，考量產能限制及運輸前置時間。Giglio et al., (2003)利用混整數規劃方法，考量產能及運輸限制、運輸承載上限及供應商供給限制，提出一多廠區物料需求規劃模式(MRP multi-site model)。Shen et al. (2003)利用網際網路為基礎建設的網路合作代理人系統(Internet-enabled collaborative agent systems)，運用於多廠區環境，解決生產規劃問題。Chan et al., (2005)採用 AHP 為準則權重和基因演算法算法來決定工作分配。Lanshun et al., (2008)利用基因演算法及拉格蘭日鬆弛法(Lagrangean Relaxation method)解決多廠區生產規劃問題。總體而言，上述研究皆僅考慮單階多廠的生產環境，並未考量供應網絡多階層之議題。

另外，尚有其他研究考量多階多廠環境之供應網絡生產規劃問題。Lendermann et al., (2001)提出以模擬的方式，建立多階多廠的供應網絡架構，考量多種品項、物料替代關係及製造廠間之物料調撥，然而，此研究並未考量各製造廠之產能限制。Watson and Polito (2003)提出以 TOC 為基礎

之啟發式演算法模式，解決多品項於多階多廠生產環境之訂單分配問題。Gen and Syarif (2005)利用使用生成樹為基礎的混合基因演算法算法和模糊邏輯控制器用來自動調整基因演算法的參數。Yeh (2005)提出了混合整數線性模型，並且結合貪婪策略(Greedy method)和區域搜尋法(Local Search method)求解，此研究考量多階多廠區生產環境訂單分配問題，但並未考量作業的前置時間。Yeh (2006)再次提出以結合基因演算法、貪婪啟發式和區域搜尋法的瀰集演算法(Memetic algorithm, MA)，解決多階多廠生產環境訂單分配問題，但作業的前置時間依然沒有考量到。Lim et al., (2006)提出全面性考量的供應網絡數學模型，並以模擬的方式解決多階多廠區訂單分配的問題。Aliev et al., (2007)利用模糊規劃(Fuzzy Programming)來建立數學模型，並且結合基因演算法來尋求最佳解。Lin and Chen (2007)利用混整數規劃方法，提出一多階多廠訂單分配模式，但此研究並未考量原物料供給限制。Kanyalkar and Adil (2010)提出一線性規劃模式，考量多種品項於多階多廠生產環境，解決訂單分配問題。除此之外，在多階多廠之供應網絡環境下，有部分學者提出以有向圖形(Directed Graph)描述此種生產環境，將供應網絡生產規劃問題轉變為有向圖形上，更方便於決定供應網絡生產規劃之路徑選擇及數量指派。Altiparmak et al., (2009)提出穩態基因演算法(Steady-state Genetic Algorithm)，並且跟線性規劃、啟發式拉格蘭日(Lagrangian heuristic)、混合式基因演算法(Hybrid Genetic Algorithm)和模擬退火法(Simulated Annealing)的結果做比較。Altiparmak et al. (2006)以有向圖形描述供應網絡環境，如圖 2.6 所示，提出以基因演算法為基礎之多目標供應網絡規劃模式，利用基因演算法決定訂單滿足優先順序，解決多階多廠訂單分配問題，然而，此研究是以逐筆訂單進行滿足之方式，並沒有考慮訂單交期及運輸前置時間。

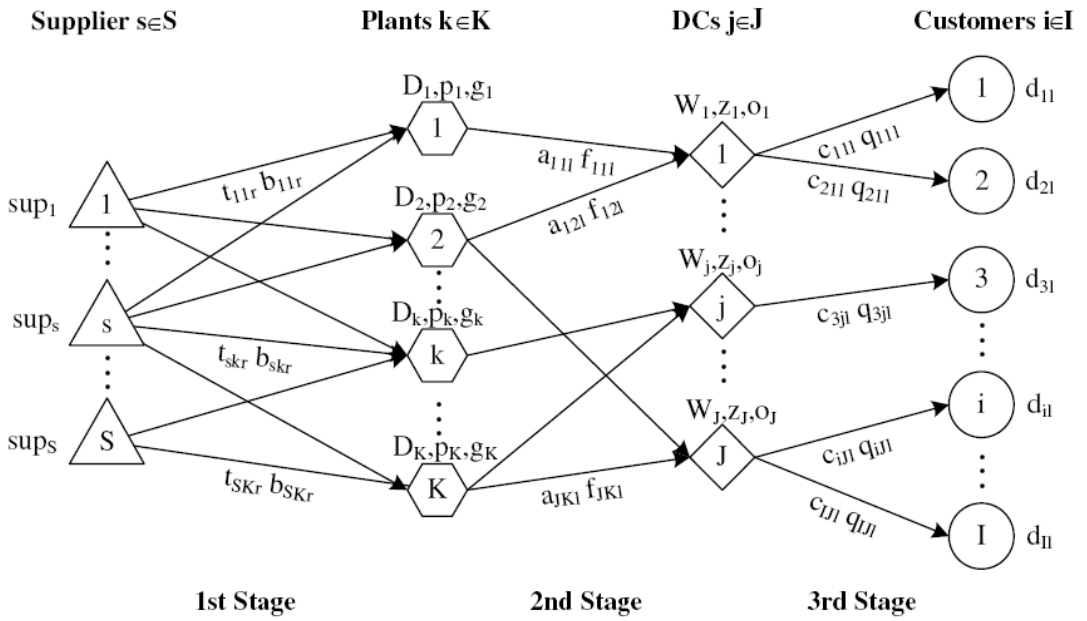


圖 2.6、有向圖形描述供應網絡

Chern et al., (2007)利用有向圖形描繪供應網絡環境，提出一多目標主排程演算法(Multi-objective Master Planning Algorithm, MOMPA)，在多種品項之基礎下，解決供應網絡主排程問題，然而，此研究以逐筆訂單滿足的方式，進行訂單分配，無法達到整體最佳解。另一方面，由於記憶體模組產業其關鍵原物料記憶體採購價格會隨著時間點由有所差異，因此，進行規劃時必須要考量時間點的因素。Wu (2004)提出以有向圖形建立供應網絡環境，表達多種品項、多廠區及多個時期等因素，如圖 2.7 所示，將不同的廠區、不同的品項及不同的時期都區分為不同的節點，考量多品項於多階多廠環境，利用拉格蘭茲鬆弛法進行生產規劃，以滿足各時期的需求節點(D)，然而，此研究並未考量運輸前置時間及原物料供給限制。

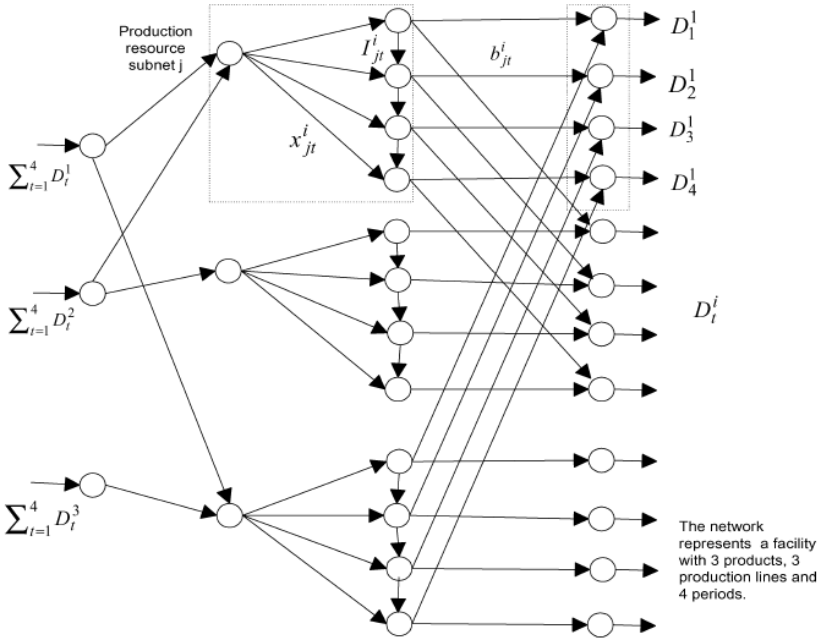


圖 2.7、多種品項、多廠區及多個時期之供應網絡圖

總體而言，上述多階多廠相關研究都各自提出了求解的方法，然而對於以數學模型為基礎的平行計算之研究卻非常稀少，然而在其它非供應鏈網絡的問題之研究中，應用平行計算對於求解問題的效率都有顯著的影響，因此應用平行計算於供應鏈網絡是一個值得探討的議題。

此外，Wu (2004)進一步證明，即使為單一品項、多廠區、多時期、無限產能且無運輸成本的供應網絡生產規劃問題下，將會是個 *NP-Complete* 的問題，因此，作者進而提出可利用最短路徑法 (Shortest Path Algorithm)，搜尋成本最低的路線，便能有效解決此模式的問題。Chen and Chern (1999) 即利用最短路徑法，在已知的物料清單及供應網絡運輸路線下，解決供應網絡生產規劃問題，然而，最短路徑法必須在特定的限制下才能實行，如下列所述：

1. 必須要有起始節點及終止節點
2. 一次只能規劃一筆需求，無法同時規劃所有需求
3. 無法同時由兩個供給節點滿足一個需求節點

在進行訂單分配時，若逐筆訂單進行規劃，其規劃結果可能因滿足的優先順序而有所差異，無法達到整體之最佳解，以記憶體模組產業而言，若以逐筆訂單規劃的方式，可能優先規劃的訂單會使用到之後訂單必須使用的原物料或半成品，而造成優先順序較低的訂單因缺料而無法滿足。另

一方面，在供應網絡環境中，當一筆需求只能由一個供給途程進行滿足，無法同時由多個廠區進行滿足，便失去供應網絡多廠區間彼此支援的特性，亦可能使訂單因單廠產能限制而無法滿足。目前多廠區相關文獻其考量特性及求解技術整理如表 2.2。

表 2.2、多廠區生產規畫相關研究 (本研究整理)

文獻	求解工具	規劃範圍	規劃特性			
			物料供給限制	訂單交期	產能限制	運輸前置時間
Moon et al. (2002)	基因演算法	單階多廠		✓	✓	✓
Giglio et al. (2003)	混整數線性規劃	單階多廠	✓	✓		✓
Shen et al. (2003)	代理人	單階多廠		✓		✓
Watson & Polito (2003)	啟發式演算法	多階多廠		✓	✓	✓
Wu (2004)	混整數線性規劃	單階多廠		✓	✓	
Chan et al. (2005)	基因演算法	單階多廠			✓	✓
Gen & Syarif (2005)	基因演算法	多階多廠		✓		
Yeh (2005)	啟發式演算法	多階多廠	✓		✓	
Altiparmak et al. (2006)	基因演算法	多階多廠	✓	✓	✓	
Lim et al. (2006)	模擬方法	多階多廠	✓	✓	✓	
Yeh (2006)	啟發式演算法	多階多廠	✓		✓	
Aliev et al. (2007)	基因演算法	多階多廠		✓	✓	
Chern et al. (2007)	啟發式演算法	單階多廠		✓	✓	✓
Lin & Chen (2007)	混整數線性規劃	多階多廠		✓	✓	✓
Lanshun et al. (2008)	基因演算法	單階多廠		✓	✓	
Altiparmak et al. (2009)	基因演算法	多階多廠	✓	✓	✓	
Kanyalkar & Adil (2010)	線性規劃	多階多廠		✓	✓	✓
本研究	INLP 的平行計算	多階多廠	✓	✓	✓	✓

就記憶體模組產業之供應網絡生產規劃而言，屬於多階多廠、多品項及多時期之環境，並且需考量產能及生產/運輸前置時間，相較於 Wu (2004) 學者所考量之供應網絡環境要複雜許多，故亦屬於 *NP-Complete* 的問題，為了能有效解決此模式問題，需要一演算法在適當時間內，求得近似最佳解。此外，由於記憶體模組產業關鍵原物料採購價格及供給數量隨時間而有所差異等產業特性，不適合以逐筆訂單的規劃方式進行規劃。因此，本研究將有效地描繪記憶體模組廠多階多廠的供應網絡及建立一個符合記憶體模組產業的數學模型，利用分散式平行系統在適當時間內，以平行計算的方式求得最佳規劃結果，解決記憶體模組產業供應網絡生產規劃之議題。

2.4 平行計算

為了追求解決工程及科學計算上的需求，人類不斷的製作更高速的處理器(CPU)與計算主機。由於高速計算的需求並不因為半導體技術的的進步而減少，且單一處理器的性能並不足以滿足計算的需求，故工程師開始建構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叢集電腦(cluster)平台，利用程式平行化的技術，以獲取單一處理器所無法提供的計算性能。

平行計算被提出來討論的原因乃是由於計算量及計算複雜度的增加，Flynn (1972)在早期的文獻提出計算機系統結構的分類法，Hwang and Briggs (1984)也提出早期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。

平行處理有兩種應用層面：(1)多個處理器去分別執行不同的工作。對一般的個人電腦來說，同時要上網、聽音樂、燒錄光碟、影像處理、多媒體檔等，這些是非常消耗處理器資源的，因為在一個時間點上處理器只能執行一個程序，因此上述一連串的工作，就會讓所有工作效率被拖慢，因為處理器會每一種程序都花一點時間在上面。因此對於上述的工作若是有多個處理器存在時，不同的工作可以丟給不同的處理器去執行，這樣會讓多功執行效率非常高，這樣的電腦又可稱為 OpenMosix。(2)多個處理器去執行一個工作。這部分就是所謂的平行計算，因為原本一個處理器要解問題所花的時間太長，因此透過多個處理器來分擔工作，減少解決問題所花費的時間。不過也因為這個狀況是將一個程序讓多個處理器去執行，因此對於將程式平行化所需要的程式改寫無法避免，以下將詳細的說明平行計算系統的位址空間架構、硬體架構及訊息傳遞模式。

2.4.1 平行計算系統之位址空間架構

Wilkinson and Allen (1999)將目前的平行電腦依記憶體的配置分為兩大類：

1. 共享式記憶體平行電腦(Shared memory multi-process system)

2. 分散式記憶體平行電腦(Distributed memory multi-process system)

共享式記憶體就是一台電腦中有非常多個處理器，而他們都共用同一個記憶體，所有的處理器都可以存取記憶體的任何位置。像是 SGI Origin 2000 或是 Sun HPC servers 都是共享式記憶體架構的電腦，這又稱為對稱式處理器(Symmetric Multiprocessor, SMP)系統。其作業系統的核心會動態分配工作至各處理器，進而平衡各處理器之間的負載，無論是循序程式或是平行程式都可以同時執行。只要實際的記憶體足夠，所有執行中的程序都可以要求它們所需要的記憶體。所以在處理器數量非常多的時候，對於記憶體的存取會成為共享記憶體架構之電腦的瓶頸，圖 2.8 說明了共享式記憶體多處理器系統的架構。

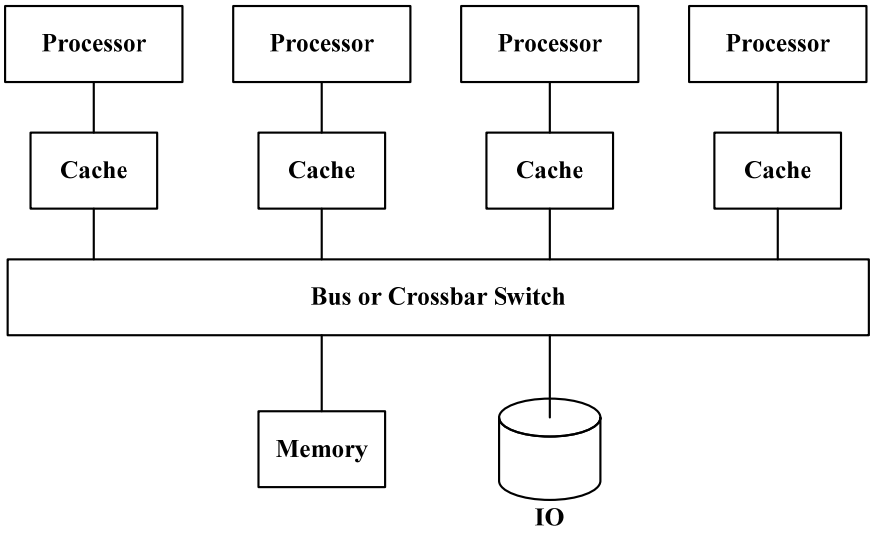


圖 2.8、共享式記憶體多處理器系統架構圖示意圖

分散式記憶體平行電腦系統中每一個處理器都有各自的記憶體和操作系統，也就是由多個獨立電腦組成的集合體，當你想要存取其他處理器的計算資料時，就必須透過訊息傳遞介面網路(Message Passing Interconnection Network)，此介面網路並非一定是實體之線路，另外它也無法自動完成處理器之間的負載平衡，通常訊息傳遞的機制是利用中介軟體來達成，如常見的 PVM(Parallel Virtual Machine)或 MPI，由於此類電腦利用介面網路作為傳輸介面，所以不會因為硬體限制其平行處理器的數目，所以此類電腦又稱為大量平行系統(Massive Parallel Processors, MPP)，圖 2.9 說明了分散式記憶體多處理器系統的架構。而分散式記憶體架構的電腦又可以分成兩

類：

1. 商用分散記憶體平行電腦：像是 IBM SP2、Cray T3E。
2. 將個人電腦或工作站藉由網路串連而成的叢集電腦系統：例如以 Linux 作業系統搭配個人電腦的 Beowulf cluster。

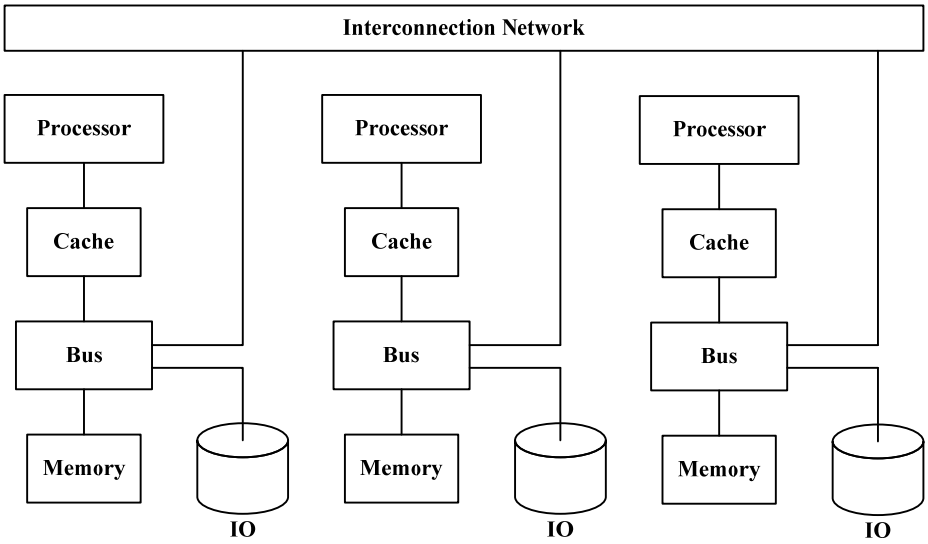


圖 2.9、分散式記憶體多處理器系統架構示意圖

對於相同處理器數目的共享式記憶體以及分散式記憶體架構的電腦，前者通常比後者要昂貴，後者的花費往往跟處理器的數目成線性關係，表 2.3 比較了兩種模式的優缺點。

表 2.3、共享式記憶體平行電腦 v.s 分散式記憶體平行電腦

	共享式記憶體平行電腦	分散式記憶體平行電腦
優點	1. 系統會把工作交給負載較輕的處理器去執行，可以做到處理器之間的負載均衡分布。 2. 沒有處理器之間的資料傳輸問題。 3. 循序程式是很容易移植至平行電腦上。	1. 存取局部記憶體的延遲時間短。 2. 硬體成本較低廉，擴充性較佳。
缺點	當處理器增加時必須要花費大量的成本，如此一來擴充性就會打折扣。	須將原本的循序程式做平行處理。

在平行電腦中，處理器之間的通訊是非常重要的一環，而共享式記憶體

中的匯流排(Bus)即相當於分散式記憶體電腦中的網路。

2.4.2 平行計算系統之控制機制

Flynn (1996)依電腦的提取指令和資料之運行機制的不同進行分類，在單處理器的電腦中，由程式生成的指令，依序對數據項進行運算，這種程式的電腦稱為單指令單數據式電腦(Single Instruction Single Data, SISD)。在多處理器的電腦中，每一顆處理器擁有一個獨立的程式，每一個程式為該處理器產生指令，每條指令對不同的數據進行運算，這種形式的電腦稱為多指令多數據式電腦(Multiple Instruction Multiple Data, MIMD)。如果將電腦設計成由單一程式生成單一指令集，卻允許有多個數據流存在時，當程式的指令被傳到各個處理器，每一個處理器實質上是一個沒有控制器的算術運算處理器，而由一個獨立的控制器負責從記憶體中取出指令，並將這些指令傳送到各個處理器中。每個處理器同步執行相同的指令，但使用不同的數據資料。因不同數據群組形成的處理器陣列，可在一個指令週期內對整個陣列進行運算，Flynn 將這稱為單指令多數據式電腦(Single Instruction Multiple Data, SIMD)。

Wilkinson and Allen (1999)說明了 MIMD 電腦中，每一個處理器擁有獨立的執行程式，其下又可分為多程式多數據(Multiple Program Multiple Data, MPMD)結構，以及單程式多數據(Single Program Multiple Data, SPMD)結構。多程式多數據結構，主要是不同程式在不同的處理器上執行，通常是一種主從架構，也就是說主要的處理器(Master)將會負責分配程式到每一個副處理器(Slave)上去執行，換句話說，所有的副程式和計算資料都是由主程式來啟動和分配出去的，圖 2.10 為 MPMD 的架構圖。而單程式多數據則僅需編寫一個主程式，此程式可以負責和控制所有的工作，也就是說一個程式可以在多個處理器上同時執行。對於主副結構來說，該程式的一部分是由主處理器執行，而另一部分則由其他的副處理器執行，圖 2.11 為 SPMD 的架構圖。

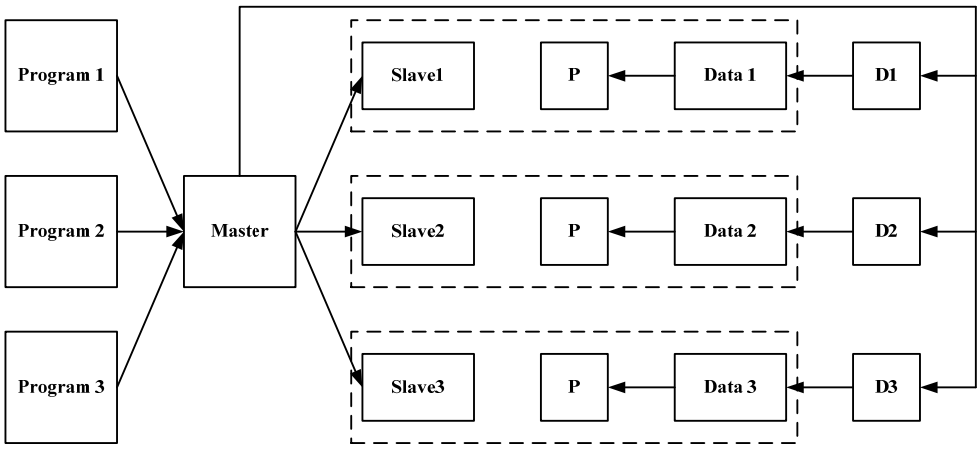


圖 2.10、MPMD 架構圖

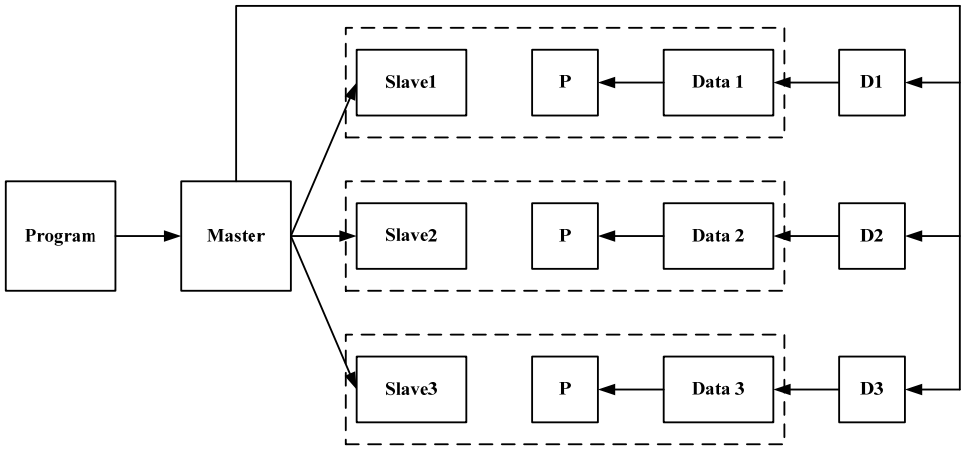


圖 2.11、SPMD 架構圖

2.4.3 平行計算系統之訊息傳遞模式

個人叢集電腦系統一般都屬於分散式記憶體平行電腦系統，每一個獨立的節點有它自己的處理器、記憶體與操作系統。此分散式記憶體平行電腦中各計算單元是用特製的高速網路連結在一起，訊息的傳遞和資料的交換全靠此網路完成。針對各項不同硬體架構應選擇適合的平行編譯機制，才能將平行化之後的額外負擔(Penalty)降到最低，提升整體效能。

在訊息傳遞模式中，一個平行之程序是由多個平行之任務所完成。每個處理單元除了進行本身的資料計算外，也須透過資料的交換完成工作。各平行任務間數據的交換是透過呼叫訊息交換的語法來完成。一般常見的訊息傳遞模式有兩個，即 PVM 和 MPI。

PVM 是由美國 Tennessee 大學、Oak Ridge 國家實驗室等研製的平行程式開發環境。它可以把多個異質環境下的電腦組織起來成為一個易於管理、可擴展及易編譯使用的平行計算資源。它的計算結點可以是 SMP 或

MPP 類型的平行電腦、向量超級電腦、專用圖形及標量的工作站等。這些結點可以透過多種網路互聯，成為一個網路虛擬電腦。處理問題時，計算資料被分配到各個計算節點，由多個節點同時計算進行平行處理。

因為 PVM 這類的 Message-Passing System 的成功導致 MPI 的出現。PVM 的成功讓研究平行計算的一些學者認為有必要凝聚共識達成標準，經由討論使 MPI 實踐的標準不會因為各家方法的不同而產生交流上的困難，讓應用程式能更有效能和移植性，同時也可以加速平行計算領域的研究和應用，使其更加的成熟。

PVM 與 MPI 所提供的功能大致相同，只是兩者重點方向略有不同，幾乎所有平行計算系統都支援 PVM 和 MPI。PVM 特色在異質環境下的可攜性和互操作性，程式間可以互相通信，且支援動態的資源管理和容錯；MPI 更強調性能，不同的 MPI 間缺乏互操作性，本身也不支持容錯(可透過專門的容錯軟體來支援)。就一般而言，PVM 比較適合異質性之叢集系統而 MPI 因有較高的通訊效能，比較適合於開發在同質性之平行程式。

陳權 (2009)說明相對於高速計算平台各式硬體架構之蓬勃發展，平行化之系統軟體相對落後；此軟體系統發展較為緩慢主要有下列幾項因素：

1. 平行化之程式較傳統序列程式更為複雜且具更高技巧與挑戰性。
2. 序列程式要轉換成平行程式會因平行編譯系統不同而有不同的模型。
3. 平行化之除錯程式(Debuger)較少，且除錯的環境難以模擬。
4. 各計算節點的計算量不均或 I/O 與網路寬頻造成系統更複雜。

本節所介紹的平行計算與客戶端/伺服端(Client/Server)導向模式不同，平行化之系統軟體同等對待各個 PE(Processor Element)並以平行處理的方式進行工作處理，廣泛應用於需平行處理的大規模科學計算中，下節將介紹分散式物件系統，說明如何應用多個伺服器進行分散式數值運算，實現高速平行處理。

2.5 分散式系統

在程式語言的快速發展之下，大部分的程式語言都可以寫出一個透過網際網路連線的系統，而此系統通常都採用 Client/Server 的架構。伺服端負責向跟它提出請求的客戶端提供服務。然而，在目前的網際網路的大量使用下，伺服端和客戶端的互動變得更複雜而且龐大，因此需要存在著一

種機制來建構、維護和擴充原來的 Client/Server 系統。

在目前來講，為了解決上述的問題，「分散式物件」技術變得越來越重要，此方式是處理 Client/Server 兩端物件之間的關係，將單一的 Client/Server 系統切割成許多單一且易於管理的的物件，讓分散環境中不同地方的物件能像在同一台電腦中相互通訊，如此可以建立更有彈性的 Client/Server 系統。由於物件是在網路上分散執行的，因此，這樣的系統稱為分散式系統。

Orfali et al., (1996)提出分散式物件所帶來的好處主要有以下幾點：

1. Plug and Play：藉由定義完整的介面，即使在複雜和龐大的系統也可以是由許多物件組裝而成，而系統內的物件可以輕易的增加和移除。
2. Interoperability：經過介面定義的物件可以藉由 ORB(Object Request Broker)互相傳遞訊息，且傳遞的訊息不只包含資料還包含資料的處理方法，當一個物件存在於分散式系統中時，它就可以提供服務給其它物件，並且可以向其它物件請求服務。
3. Portability：在分散式系統中所有的物件，都必須透過介面(Interface)來定義其所提供的服務，而不需要考慮物件的位置或者是物件以何種方式實現(Implement)，因此系統中所有的物件可以在不同的網路平台和作業平台上作業，具有良好的可攜性。
4. Coexistence：傳統非物件導向的程式可以經過包裝而成為新的物件，舊有的程式可以再被使用。

分散式物件技術為目前主從架構發展的趨勢，有許多分散式物件標準為大家所採用的，例如 Java 中的 RMI 機制、由 OMG(Object Management Group)所制定的 CORBA 規格和微軟所制定的 DCOM 架構與網路服務(Web Service)等，以下將針對本研究所採用的 JavaRMI 做詳細的介紹，及目前最受業界支持的 CORBA 作簡單的介紹。

2.5.1 JavaRMI

許良政(2008)說明了 JavaRMI 是 Java 語言中的遠端程序呼叫(Remote Procedure Call, RPC)機制，Client/Server 可以分散於不同電腦，各物件之間經由網路來進行資料交換的一種資料結構。JavaRMI 可使在不同機器上執行的各 Java 物間之間能像在同一台機器上執行一樣進行通訊。再採用 JavaRMI 的分散式物件系統中，執行於不同的 Java 虛擬機器(Java VM)上的

Java 物件之間可以自由的進行資料交換，如圖 2.12 所示。

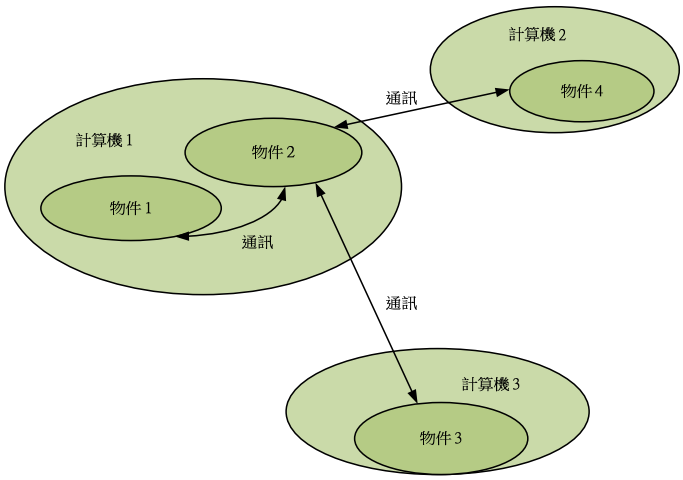


圖 2.12、採用 JavaRMI 建構的分散式物件系統

顏春煌 (2008)指出遠端程序呼叫式開發網際網路應用成是常用的方法，主要是因為使用 socket 介面撰寫網路應用程是時常需考慮各種通訊的細節，例如資料行是與結構的轉換、連線的管理等。假如我們能將這些細節簡化，則程式設計者可以把時間轉移到應用系統的需求上，這才是使用者最關心的。遠端程序呼叫的基本原理，是把網路的通訊看成是程式中的程序(procedure)，當程式需要遠端程序提供某種服務或接受訊息時，直接呼叫一個程序，而這個程序有相對應的遠端程序，可以在呼叫後於遠端的電腦上被執行。

郭尚君 (2008)指出相較於以往呼叫遠端程序的遠端程序呼叫技術，與 JavaRMI 基本觀念相同，但 JavaRMI 能夠將執行於遠端電腦的程式視為遠端物件(Remote object)，因此，更能符合物件導向觀念。應用上，可以將需要大量運算的敘述封裝為遠端物件的方法，然後置於運作能力較佳的伺服器上，透過 JavaRMI 呼叫，並取得計算結果。JavaRMI 的運作架構如圖 2.13 所示。

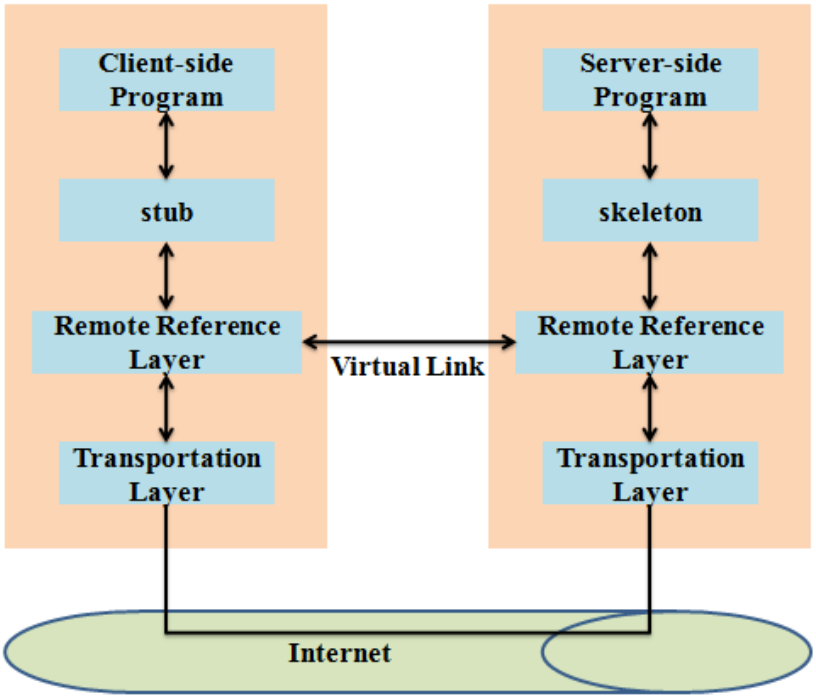


圖 2.13、JavaRMI 的運作架構

JavaRMI 的主要運作核心為 stub 和 skeleton 程式，當客戶端程式呼叫遠端物件的方法時，stub 將封裝傳入方法的參數，然後透過網路傳輸給伺服端電腦，然後由 skeleton 程式拆解參數，呼叫物件的方法。完成執行後，將在封裝回傳值，並送回客戶端電腦，由 stub 程式拆解回傳值。因此，程式設計師並不需要處理連接伺服端電腦、傳遞資料與處理型別轉換...等瑣碎工作，只需要專注於程式邏輯的處理。

而在客戶端程式內，將以介面參考引用遠端物件，這個參考將連接至本機的(客戶端)的 stub 程式，由該程式處理網路通訊的相關問題。因此，客戶端必須取得遠端物件所時做的界面。

圖 2.14 顯示以 JavaRMI 為基礎的分散式物件系統之架構，在應用 JavaRMI 建構系統時，程式設計師要是先對客戶端和伺服端進行分工，接著在 JavaRMI 中以下面的三種功能支援分散式物件程式設計：

1. 遠端物件的配置：應用 JavaRMI 進行分散式物件設計時，物件時計分散在網路上甚麼位置是由 RMI 系來管理的，因此，應用程式需要向 RMI 登陸物件所存放的位置。在 RMI 中，經由 RMIRegistry(RMI 命名註冊器)結構，為程式的執行提供必要且與物件配置相關的訊息。
2. 遠端物件之間的通訊：應用 RMI 時，不必刻意區分使用應用程式內部的物件和網路上其他不同計算機的物件。因此，RMI 要代理應用程式

進行詳細的網路處理工作。

3. 物件自身的互動：在 JavaRMI 中，不僅是對資料，同時也是針對物件的位元組碼提供了在分散式系統內部的互動方式。

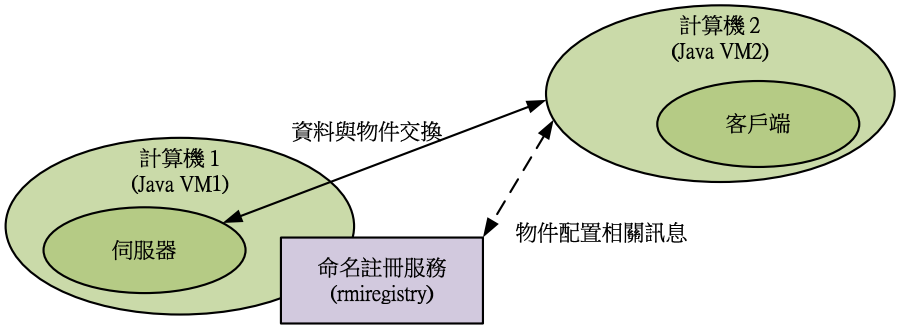


圖 2.14、基於 JavaRMI 分散式物件系統的架構

使用 JavaRMI 於物件導向的分散式系統中時，網路通訊的細節會被隱藏。換句話說，與使用 socket 進行通訊的情況相較，使用 JavaRMI 不必詳細描述網路細節，故可以進行較高層次的抽象程式設計。當然在分散式物件系統中，仍可以採用 socket 進行通訊。但是採用 socket 進行通訊時需要準備許多與網路使用相關的工作，而且 socket 通訊的基本原則是每次只能發送一個字元。因此，當需要傳送整數、實數或要傳送具有一定結構的資料時，須要將這些資料調整轉換成以字元為單位的形式，如圖 2.15 所示。

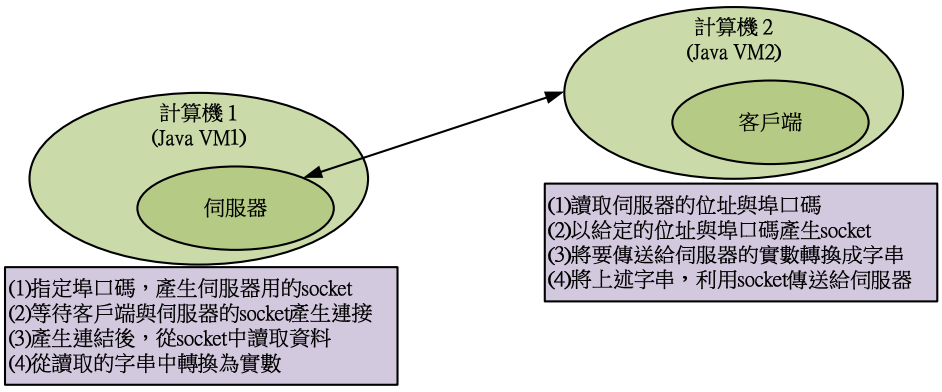


圖 2.15、利用 socket 傳送實數

在上述的過程中不僅要處理網路位址及埠口碼等網路的詳細訊息，還必須在 Java 程式終將使用的資料型態由實數轉換為字串，同時還包含字串的反轉。如果在客戶端與伺服器間利用 JavaRMI，則傳送實數時的處理如下。首先，在伺服器端除了要啟動伺服器類別外，只要先執行 RMIRegistry，

就可以提供伺服器的功能。客戶端的處理程序也很簡單，如果指定它使用RMIRegistry，只要以呼叫伺服器端的方法就可以利用伺服器所提供的服務，如圖 2.16 所示。

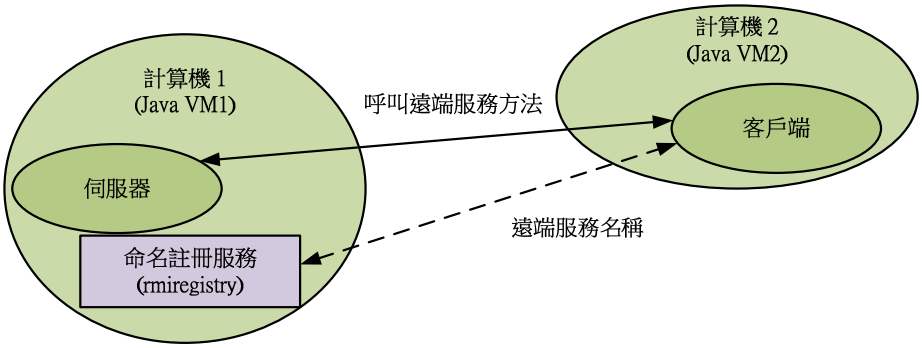


圖 2.16、利用 JavaRMI 傳送實數(呼叫方法)

2.5.2 分散式物件系統之工業標準(CORBA)

隨著分散式系統及物件導向技術的發展，以物件為基礎的分散式架構也受到高度重視。因此物件管理組織(Object Management Group, OMG)在 1990 年制定了一個分散式物件溝通的標準 CORBA(Common Object Request Broker Architecture)，這是目前最受業界支持的分散式物件的標準。

和 JavaRMI 一樣，CORBA 提供的是一種用於建構分散式物件系統的結構，JavaRMI 與 CORBA 是用於同一目的的不同框架。相對地，CORBA 處理的對象是包含 Java 在內的多種語言，如表 2.4 所示。因此，當需要對已存在的系統進行分散式處理，或者要建構用多種語言描述的大規模分散式系統時，採用 CORBA 是比較適合的。

表 2.4、CORBA 可支援的程式設計語言

Ada
C
C++
COBOL
CORBAScrip
Java
Lisp
PL/I
Python
Smalltalk

資料來源：CORBA

基於上述原因，我們經常透過 CORBA 以 Java 語言程式開發設計分散式物件系統。但與僅採用 JavaRMI 的系統相比，採用 CORBA 多少會使描述變得複雜些。此外，由於程式設計上的差異，採用 CORBA 有時會使程式碼變大一些

第三章 供應網絡生產規劃模式

記憶體模組廠之供應網絡生產規劃主要分為三個階層，如圖 3.1 所示。第一階層為供應商供給原物料至製造廠；第二階層為製造廠接收原物料，並將其製造成半成品；第三階層為製造廠運送半成品至配銷中心，並組裝成為完成品以滿足需求。

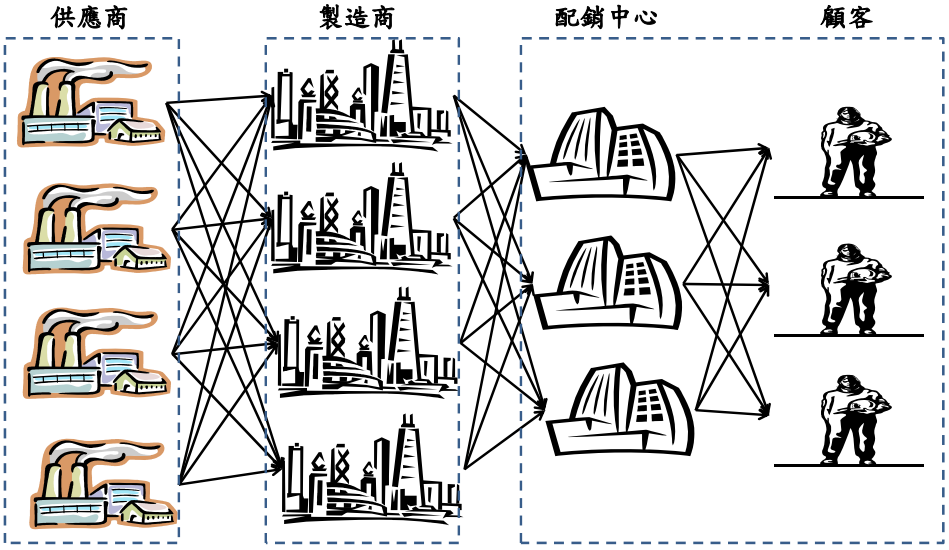


圖 3.1、記憶體模組廠供應網絡架構

針對此三個階層，每個階層在進行供應網絡生產規劃時，皆有其必須考量的特性及限制，接下來的部分將詳細進行說明。

第一階層：供應商供給原物料至製造廠

1. 物料供給限制：各個原物料在不同的週期，其可供給的數量亦有所異。因此，進行供應網絡生產規劃時，必須考量各個供應商能供給的數量。
2. 作業前置時間：進行供應網絡生產規劃時，必須要考量各個供應商的作業前置時間，包含由供應商運輸至製造廠的運輸前置時間及其生產的前置時間，避免規畫出不合理的交期。
3. 考量各項成本：考量供應商相關的各項成本因素，包含供應商之採購成本及供應商運輸至製造廠之運輸成本，以求最低之總體成本。

第二階層：製造廠接收原物料，並將其製造成半成品

1. 製造廠產能限制：各個製造廠皆有其產能限制，在此限制下，即使有足夠的物料，只要產能不足，亦無法利用此製造廠進行生產活動。

2. 作業前置時間：進行供應網絡生產規劃時，必須要考量各個製造廠的作業前置時間，包含由製造廠運輸至配銷中心的運輸前置時間及其生產的前置時間，避免規畫出不合理的交期。
3. 考量各項成本：考量製造廠相關的各項成本因素，包含製造廠之生產成本、製造廠運輸至配銷中心之運輸成本及製造廠之存貨持有成本，以求最低之總體成本。

第三階層：製造廠運送半成品至配銷中心，並組裝成完成品滿足需求

1. 訂單交期時間：各個配銷中心的訂單皆有其交期，在進行供應網絡生產規劃時，必須根據交期，決定由哪些製造廠滿足該筆訂單且於何時進行生產，進一步決定由哪些供應商於何時運送原物料，避免規畫出不合理的訂單交期。
2. 訂單售價：由於原物料供給數量有限，當需求大於元物料可供給之數量時，考量訂單的單位售價，決定如何將有限的資源分配給各筆訂單，以達最大獲利。
3. 組裝前置時間：考量各個配銷中心的組裝前置時間，避免規畫出不合理的訂單交期。
4. 考量各項成本：考量配銷中心相關的各項成本因素，包含配銷中心之組裝成本、配銷中心之存貨持有成本及缺貨成本，以求最低之總體成本。

承上述，記憶體模組產業在進行供應網絡生產規劃時，必須考量許多的生產特性及限制條件，然而在實務上，要考慮所有限制有其一定程度的困難，因此，記憶體模組產業之規劃人員並沒有考量所有特性及限制。接下來的部份，將先說明記憶體模組產業供應網絡生產規劃的現況方法，比較其與所有記憶體模組產業之特性及限制的差異，並以視覺化之圖形來說明此供應網絡，考量大部分生產特性及限制條件，解決記憶體模組產業之供應網絡生產規劃問題。

3.1 記憶體模組產業供應網絡生產規劃現況方法

記憶體模組產業供應網絡生產規劃的現況方法如圖 3.2 所示，包含有 9 個步驟，接下來將予以進行說明。

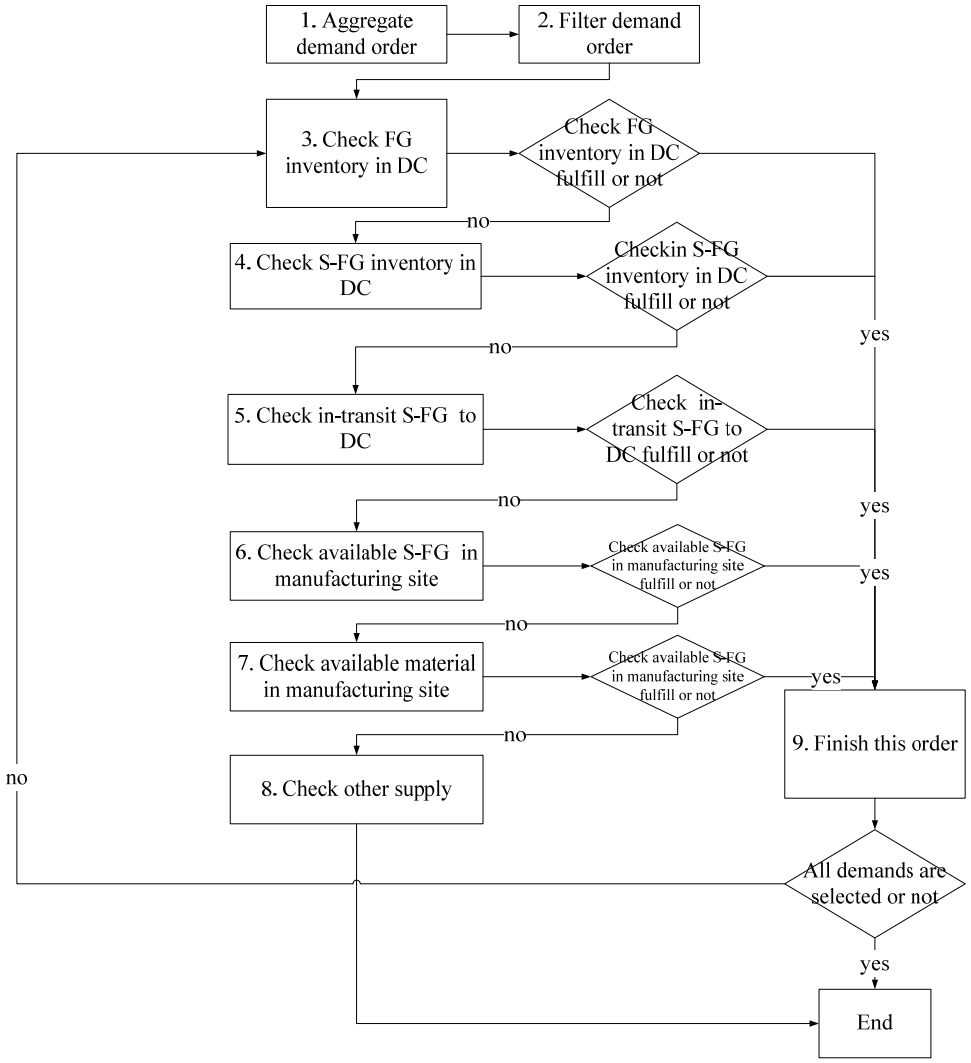


圖 3.2、記憶體模組廠之現行供應網絡生產規劃方法

1. 彙總各個配銷中心在規劃週期時間內的訂單需求
2. 根據交期、顧客重要性、需求數量及延遲處罰成本等決定訂單滿足的優先順序
3. 檢視目前配銷中心是否有足夠的完成品數量能滿足該筆訂單之需求，若能滿足則進入步驟 9；若未能滿足需求數量，則將配銷中心現有之完成品數量分配至該筆訂單，並計算尚未足夠之數量進入步驟 4。
4. 當配銷中心之完成品無法滿足訂單需求時，考慮產品組合結構，檢視配銷中心是否有足夠的可用半成品數量能生產為完成品以滿足該筆訂單之不足量，若能滿足，則依據可用半成品數量的多寡決定以哪些可用半成品生產為完成品滿足此訂單，進入步驟 9；若未能滿足不足數量，則將配銷中心現有之可用半成品數量分配至該筆訂單，並計算目前尚未滿足訂單之數量進入步驟 5。

5. 當配銷中心之半成品無法滿足訂單時，考慮產品組合結構，檢視從製造廠運輸至配銷中心的可用半成品在途量是否能滿足該筆訂單之不足量，若能滿足，則依據可用半成品數量的多寡決定以哪些可用半成品滿足此訂單，進入步驟 9；若未能滿足不足數量，則將在途量之半成品數量分配至該筆訂單，並計算目前尚未滿足訂單之數量進入步驟 6。
6. 當製造廠運輸至配銷中心的半成品在途量亦無法滿足訂單時，考慮產品組合結構，檢視製造廠是否由足夠的可用半成品存貨以滿足該筆訂單之不足量，若能滿足，則依據各製造廠至配銷中心的運輸前置時間及可用半成品數量的多寡，決定以哪些製造廠供給哪些可用半成品以滿足此訂單，進入步驟 9；若未能滿足不足數量，則將製造廠的可用半成品分配至該筆訂單，並計算目前尚未滿足訂單之數量進入步驟 7。
7. 當製造廠的可用半成品存貨亦無法滿足訂單時，考慮產品組合結構，檢視製造廠是否由足夠的可用原物料以滿足該筆訂單之不足量，若能滿足，則依據各製造廠至配銷中心的運輸前置時間及可用原物料數量的多寡，決定以哪些製造廠供給哪些可用原物料以滿足此訂單，進入步驟 9；若未能滿足不足數量，則將製造廠的可用原物料分配至該筆訂單，並計算目前尚未滿足訂單之數量進入步驟 8。
8. 當上述步驟皆無法滿足訂單需求時，考慮產品組合結構，向供應商採購可用之原物料，依據各製造廠至配銷中心的運輸前置時間決定以哪些製造廠進行生產以滿足此訂單，結束規劃。
9. 當該筆訂單被滿足時，判斷是否尚有訂單未滿足，若所有訂單皆以滿足則結束規劃；反之，若尚有訂單未滿足則回到步驟 3。

由現行的供應網絡生產規劃方式得知，規劃人員是以逐筆訂單進行規劃，無法得到整體的最佳規劃，此外，決定各個製造廠進行生產僅依據製造廠至配銷中心的運輸前置時間，並未考量各廠目前可用產能及各項成本因素。因此，本研究將以視覺化的圖形說明此供應網絡，建立一適用於記憶體模組產業之供應網絡生產規劃模式。考量大部分生產特性及限制條件，解決記憶體模組產業之供應網絡生產規劃問題。整體而言，針對記憶體模組產業之生產特性及限制條件，比較現況方法及本研究模式所考量的條件及限制，整理如表 3.1：

表 3.1、現況方法與本研究模式之供應網絡比較表

階層	生產特性及限制條件	現況方法	本研究方法
第一階層：供應商供給原物料至製造廠	物料供給限制	V	V
	作業前置時間(生產/運輸)		V
	考量各項成本		V
第二階層：製造廠接收原物料，並將其製造成半成品	製造廠產能限制		V
	作業前置時間(生產/運輸)	V(僅運輸)	V
	考量各項成本		V
第三階層：製造廠運送半成品至配銷中心，並組裝成完成品以滿足需求	訂單交期時間	V	V
	訂單售價		V
	組裝前置時間		V
	考量各項成本		V
其它	整體規劃最佳		V

3.2 供應網絡之數學模型

承如表 3.1 所示，記憶體模組產業供應網絡生產規劃的現行方法並未考量所有的生產特性及限制條件，且是以逐筆訂單進行規劃的方式，無法得到整體的最佳規劃，因此，本研究考量大部分的特性及限制，以視覺化之圖形說明此供應網絡，解決記憶體模組產業之供應網絡生產規劃問題。接下來的部分將先介紹本模式之假設條件及已知資訊，進一步將記體模組產業的供應網絡轉換成視覺化之圖形描述，並且以此圖形為基礎建立供應網絡之數學模型，最後利用分散式平行系統解決此供應網絡生產規劃時間上的問題，以供規劃人員進行訂單分配時之參考。

假設條件：

1. 配銷中心無產能上限
2. 廠區間不進行調撥
3. 供應商僅提供原物料
4. 製造廠僅將原物料製造為半成品

5. 配銷中心僅將半成品組裝成完成品
6. 不考慮各廠區安全庫存水準
7. 規劃週期內，各項單位成本不會改變
8. 不考慮生產良率或機台故障率
9. 訂單需求皆為完成品
10. 不考慮產品組成結構

已知資訊：

1. 需求面：
   (1) 規劃週期內訂單需求
   (2) 每筆訂單單位售價
   (3) 每筆訂單的交期時間
2. 供給面：
   (1) 規劃週期內原物料供給數量
   (2) 供應商生產原物料的前置時間
   (3) 供應商運輸至製造廠的前置時間
   (4) 製造廠運輸至配銷中心的前置時間
   (5) 製造廠由原物料生產至半成品的前置時間
   (6) 各製造廠對半成品的產能上限
3. 成本面：
   (1) 單位採購成本
   (2) 需求缺量懲罰成本
   (3) 單位製造成本
   (4) 單位組裝成本
   (5) 單位存貨成本
   (6) 單位運輸成本

3.2.1 以視覺化圖形描述供應網絡

以視覺化圖形描述記憶體模組產業供應網絡生產規劃模式。其中，節點 (nodes) 所包含的資訊有 1.節點編號 2.廠區 3.供應商和製造廠之出貨時間 4.最大產能/可供給量 5.作業成本 6.單位存貨持有成本 7.訂單節點之交期時間；而連接線 (arcs)所包含 1.運輸數量 2.作業前置時間 3.單位運

輸成本，如圖 3.3 所示：

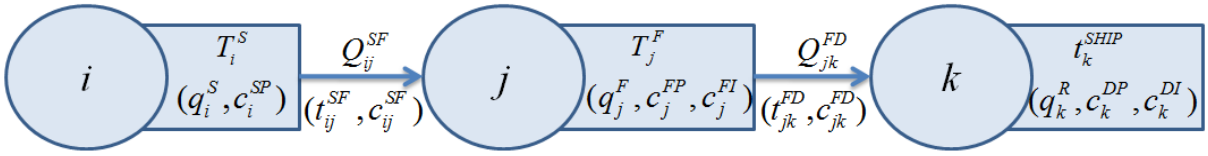


圖 3.3、視覺化圖形定義說明

下標說明

i　　供應商節點編號　　$(i = 1, 2, \ldots, I)$

j　　製造廠節點編號　　$(j = 1, 2, \ldots, J)$

k　　訂單節點編號　　$(k = 1, 2, \ldots, K)$

圖形符號定義

時間：

T_i^S　　供應商節點 i 的出貨時間

T_j^F　　製造廠節點 j 的出貨時間

t_k^{SHIP}　　訂單節點 k 的訂單交期時間

t_{ij}^{SF}　　供應商節點 i 運輸至製造廠節點 j 的作業前置時間 (包含製造及運輸)

t_{jk}^{FD}　　製造廠節點 j 運輸至訂單節點 k 的作業前置時間 (包含組裝及運輸)

成本：

c_i^{SP}　　供應商節點 i 的單位採購成本

c_j^{FP}　　製造廠節點 j 的單位生產成本

c_k^{DP}　　訂單節點 k 的單位組裝成本

c_j^{FI}　　製造廠節點 j 的單位存貨持有成本

c_k^{DI}　　訂單節點 k 的單位存貨持有成本

c_{ij}^{SF}　　供應商節點 i 運輸至製造廠節點 j 的單位運輸成本

c_{jk}^{FD}　　製造廠節點 j 運輸至訂單節點 k 的單位運輸成本

數量：

q_i^S　　供應商節點 i 的可供給量

q_j^F　　製造廠節點 j 的最大產能

q_k^R　　訂單節點 k 的需求數量

Q_{ij}^{SF}　　供應商節點 i 運輸至製造廠節點 j 的運送數量

Q_{jk}^{FD}　　製造廠節點 j 運輸至訂單節點 k 的運送數量

其他符號定義

t_i^{SP}　　供應商節點 i 之生產前置時間

t_i^{SI}　　供應商節點 i 之存貨週期時間

Q_k^{SH}　　訂單節點 k 的缺貨數量

e_k　　訂單節點 k 的單位售價

c_k^{SH}　　訂單節點 k 的單位缺貨成本

在上述圖形和其它符號的定義過程中，英文字母大寫的變數為決策變數。以視覺化圖形描述供應網絡生產規劃模式時，不再侷限於廠區本身，考慮以廠區及週期時間為維度，拓展成為不同的節點(包含供應商、製造廠及訂單節點)。在這樣以圖形為基礎之供應網絡下，以最大總淨利為目標，如何選取最佳的訂單滿足路徑及各節點的數量，決定向哪些供應商節點採購多少原物料，再由哪些製造廠節點由原物料生產為多少半成品，最後運送至哪些訂單節點組裝以滿足需求，如圖 3.4 所示。

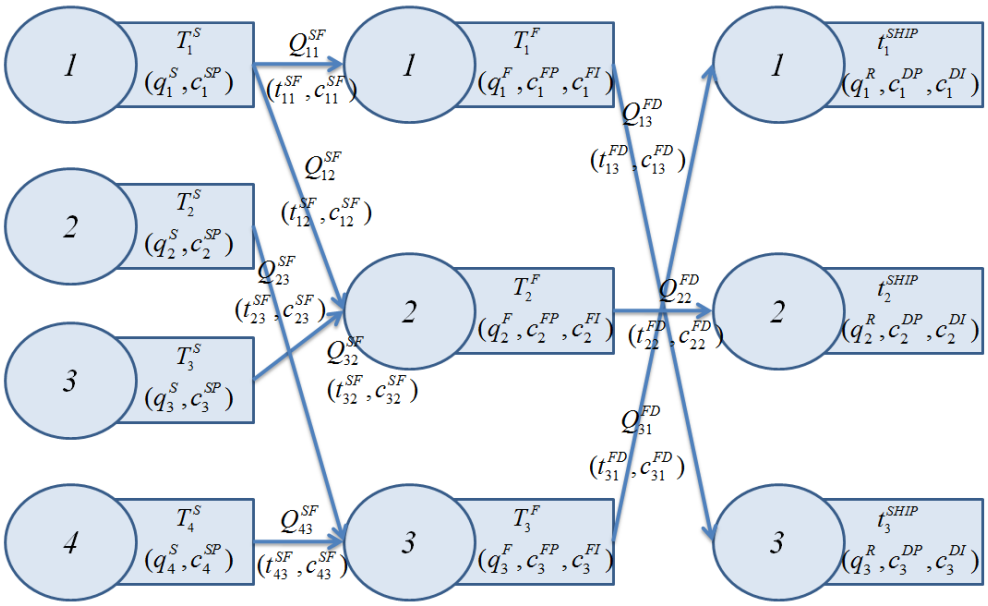


圖 3.4、案例：供應網絡生產規劃模式示意圖-以視覺化圖形描述

而影響總淨利的元素有：總盈收及總成本，目標式定義如下：

目標式

MAX = 規劃週期內總淨利(Gross Profit) =

規劃週期內總盈收(Order Revenue)－規劃週期內總成本(Total Cost)

以下將進一步針對規劃週期內總利潤及總成本兩部分，分別作詳細說明，假設今供應商節點有 I 個，製造廠節點有 J 個，訂單節點有 K 個：

1. **總盈收**

總盈收 = 訂單節點的需求數量 × 訂單節點單位售價

$$\text{總盈收} = \sum_{k=1}^{K}(q_k^R \times e_k) \quad (1)$$

q_k^R：訂單節點 k 的需求數量

e_k：訂單節點 k 的單位售價

2. **總成本**

在總成本部分，區分為採購成本(Purchase Cost)、生產成本(Production Cost)、存貨持有成本(Inventory Holding Costs)、運輸成本(Transportation Cost)及缺貨成本(Out of Stock Cost)，如下：

(1) 採購成本

採購成本 = 供應商節點運送原物料數量 × 供應商節點的單位採購成本

$$\text{採購成本} = \sum_{i=1}^{I}\sum_{j=1}^{J}(Q_{ij}^{SF} \times c_i^{SP}) \quad (2)$$

Q_{ij}^{SF}：供應商節點 i 運送原物料至製造廠節點 j 的數量

c_i^{SP}：供應商節點 i 之原物料單位採購價格

(2) 生產成本

生產成本分為製造廠製造成本及配銷中心組裝成本兩部分：

製造廠之製造成本 = 製造廠節點製造數量 × 製造廠節點單位生產成本

$$\text{製造廠之製造成本} = \sum_{i=1}^{I}\sum_{j=1}^{J}(Q_{ij}^{SF} \times c_j^{FP}) \quad (3)$$

Q_{ij}^{SF}：供應商節點 i 運送原物料至製造廠節點 j 的數量

c_j^{FP}：製造廠節點 j 之半成品單位生產成本

配銷中心的組裝成本 = 訂單節點組裝數量 × 訂單節點的單位組裝成本

(由於不考慮配銷中心之產能限制，因此訂單節點所生產之數量為製造廠節點所運送之數量)

$$\text{配銷中心的組裝成本} = \sum_{j=1}^{J}\sum_{k=1}^{K}(Q_{jk}^{FD} \times c_k^{DP}) \quad (4)$$

Q_{jk}^{FD}：製造廠節點 j 運送半成品至訂單節點 k 的數量

c_k^{DP}：訂單節點 k 之完成品單位組裝成本

(3)存貨持有成本

存貨持有成本分為製造廠的存貨持有成本及配銷中心的存貨持有成本兩部分：

製造廠的存貨持有成本 = 製造廠節點存貨數量 × 製造廠節點存貨週期 × 製造廠節點的單位存貨持有成本

$$\text{製造廠的存貨持有成本} = \sum_{i=1}^{I}\sum_{j=1}^{J}\left[Q_{ij}^{SF} \times \left(T_j^F - T_i^S - t_{ij}^{SF}\right) \times c_j^{FI}\right] \quad (5)$$

Q_{ij}^{SF}：供應商節點 i 運送原物料至製造廠節點 j 的數量

T_j^F：製造廠節點 j 的出貨時間

T_i^S：供應商節點 i 的出貨時間

t_{ij}^{SF}：供應商節點 i 運送至製造廠節點 j 的前置時間 (包含生產及運輸)

c_j^{FI}：製造廠節點 j 的單位存貨持有成本

配銷中心之存貨持有成本 = 訂單節點存貨數量 × 訂單節點存貨週期 × 訂單節點的單位存貨持有成本

$$\text{配銷中心的存貨持有成本} = \sum_{j=1}^{J}\sum_{k=1}^{K}\left[Q_{jk}^{FD} \times \left(t_k^{SHIP} - T_j^F - t_{jk}^{FD}\right) \times c_k^{DI}\right] \quad (6)$$

Q_{jk}^{FD}：製造廠節點 j 運送半成品至訂單節點 k 的數量

t_k^{SHIP}：訂單節點 k 之訂單交期時間

T_j^F：製造廠節點 j 的出貨時間

t_{jk}^{FD} ：製造廠節點 j 運送至訂單節點 k 的前置時間 (包含生產及運輸)

c_k^{DI} ：訂單節點 k 於配銷中心的單位存貨持有成本

(4)運輸成本

運輸成本分為供應商至製造廠之運輸成本及製造廠至配銷中心之運輸成本兩部分：

供應商至製造廠的運輸成本 = 供應商節點運送至製造廠節點之數量 × 單位供應商至製造廠的單位運輸成本

$$\text{供應商至製造廠的運輸成本} = \sum_{i=1}^{I}\sum_{j=1}^{J}(Q_{ij}^{SF} \times c_{ij}^{SF}) \quad (7)$$

Q_{ij}^{SF} ：供應商節點 i 運送原物料至製造廠節點 j 的數量

c_{ij}^{SF} ：供應商節點 i 運送原物料至製造廠節點 j 之單位運輸成本

製造廠至配銷中心之運輸成本 = 製造廠節點運送至訂單節點之數量 × 製造廠節點至訂單節點的單位運輸成本

$$\text{製造廠至配銷中心的運輸成本} = \sum_{j=1}^{J}\sum_{k=1}^{K}(Q_{jk}^{FD} \times c_{jk}^{FD}) \quad (8)$$

Q_{jk}^{FD} ：製造廠節點 j 運送半成品至訂單節點 k 的數量

c_{jk}^{FD} ：製造廠節點 j 運送半成品至訂單節點 k 的單位運輸成本

(5)缺貨處罰成本

缺貨處罰成本 = 訂單節點的缺貨數量 × 訂單節點的單位缺貨處罰成本

$$\text{缺貨處罰成本} = \sum_{k=1}^{K}(Q_k^{SH} \times c_k^{SH}) \quad (9)$$

Q_k^{SH} ：訂單節點 k 的缺貨數量

c_k^{SH} ：訂單節點 k 的單位缺貨成本

因此，結合式子(1) ~ (9)，記憶體模組產業之供應網絡生產規劃模式目標函數式如下所示：

$$
\begin{aligned}
MAX = & \\
& \sum_{k=1}^{K}(q_k^R \times e_k) - \\
& \sum_{i=1}^{I}\sum_{j=1}^{J}(Q_{ij}^{SF} \times c_i^{SP}) - \\
& \sum_{i=1}^{I}\sum_{j=1}^{J}(Q_{ij}^{SF} \times c_j^{FP}) - \sum_{j=1}^{J}\sum_{k=1}^{K}(Q_{jk}^{FD} \times c_k^{DP}) - \\
& \sum_{i=1}^{I}\sum_{j=1}^{J}\left[Q_{ij}^{SF} \times (T_j^F - T_i^S - t_{ij}^{SF}) \times c_j^{FI}\right] - \\
& \sum_{j=1}^{J}\sum_{k=1}^{K}\left[Q_{jk}^{FD} \times (t_k^{SHIP} - T_j^F - t_{jk}^{FD}) \times c_k^{DI}\right] - \\
& \sum_{i=1}^{I}\sum_{j=1}^{J}(Q_{ij}^{SF} \times c_{ij}^{SF}) - \sum_{j=1}^{J}\sum_{k=1}^{K}(Q_{jk}^{FD} \times c_{jk}^{FD}) - \\
& \sum_{k=1}^{K}(Q_k^{SH} \times c_k^{SH})
\end{aligned}
\tag{10}
$$

而記憶體模組產業之供應網絡生產規劃模式所需考量的限制式如下所示：

1. 限制式一：供需平衡

在規劃週期內，由於供應商的原物料供給數量有限且各製造廠亦有其產能限制，因此，各個製造廠節點供給至訂單節點的數量($\sum_{j=1}^{J} Q_{jk}^{FD}$)，可能無法滿足該訂單節點之需求($q_k^R$)，而產生缺貨數量($Q_k^{SH}$)。其限制式如下所示：

$$
\sum_{j=1}^{J} Q_{jk}^{FD} + Q_k^{SH} = q_k^R \quad \forall k = 1, 2, ..., K \tag{11}
$$

$\sum_{j=1}^{J} Q_{jk}^{FD}$ ：各個製造廠節點 j 供給至訂單節點 k 的數量總和

Q_k^{SH} ：訂單節點 k 之缺貨數量

q_k^R ：訂單節點 k 的需求數量

2. 限制式二：供應商原物料供給限制

在規劃週期內，供應商在各週期的原物料供給數量有其限制，因此，供應商節點供給至各個製造廠節點的數量($\sum_{j=1}^{J} Q_{ij}^{SF}$)必須小於等於該供應商

節點可供給的原物料最大限制(q_i^S)，其限制式如下所示：

$$\sum_{j=1}^{J} Q_{ij}^{SF} \le q_i^S \quad \forall i = 1,2,...,I \tag{12}$$

$\sum_{j=1}^{J} Q_{ij}^{SF}$ ：供應商節點 i 供給至各個製造廠節點 j 的數量總和

q_i^S ：供應商節點 i 原物料的供給上限

3. 限制式三：製造廠產能限制

在規劃週期內，製造廠在各週期的產能有其限制，因此，製造廠節點供給至各個訂單節點的數量($\sum_{k=1}^{K} Q_{jk}^{FD}$)必須小於等於該製造廠節點的最大產能限制供($q_j^F$)，其限制式如下所示：

$$\sum_{k=1}^{K} Q_{jk}^{FD} \le q_j^F \quad \forall j = 1,2,...,J \tag{13}$$

$\sum_{k=1}^{K} Q_{jk}^{FD}$ ：製造廠節點 j 供給至各個訂單節點 k 的數量總和

q_j^F ：製造廠節點 j 的產能上限

4. 限制式四：供應商供給至製造廠之總和等於製造廠供給至 DC 之總和

就記憶體模組產業而言，一個原物料能生產為一個半成品，而一個半成品能組裝為一個完成品，且本研究不探討各個廠區之安全庫存，因此，各供應商節點運送原物料至製造廠節點以生產為半成品的數量($\sum_{i=1}^{I} Q_{ij}^{SF}$)必須等於製造廠節點運送至訂單節點以生產為完成品的數量($\sum_{k=1}^{K} Q_{jk}^{FD}$)，其限制式如下所示：

$$\sum_{i=1}^{I} Q_{ij}^{SF} = \sum_{k=1}^{K} Q_{jk}^{FD} \quad \forall j = 1,2,...,J \tag{14}$$

$\sum_{i=1}^{I} Q_{ij}^{SF}$ ：各個供應商節點 i 運送原物料至製造廠節點 j 的數量總和

$\sum_{k=1}^{K} Q_{jk}^{FD}$ ：製造廠節點 j 運送半成品至各個訂單節點 k 的數量總和

5. 限制式五：路徑運輸量

運輸路徑分為兩段，供應商至製造廠之運輸及製造廠至配銷中心之運輸，以下將分別進行說明：

(1) 供應商至製造廠之運輸

當供應商節點 i 運輸至製造廠節點 j，若此兩節點間的運輸存在時，數量必須大於或等於 10，如果兩節點間沒有運輸存在時則數量為 0，因此限制式中必須要加入等於 0 或 1 的整數變數(P_{ij}^{SF})，其限制式如下所示：

$$\begin{aligned} &10\times P_{ij}^{SF} \le Q_{ij}^{SF} \\ &Q_{ij}^{SF} \le m\times P_{ij}^{SF} \\ &\forall i=1,2,...,I \quad \forall j=1,2,...,J \end{aligned} \tag{15}$$

Q_{ij}^{SF} ：供應商節點 i 供給至製造廠節點 j 的數量

P_{ij}^{SF} ：0 或 1 的整數變數，代表著供應商節點 i 至製造廠節點 j 的運輸是否存在 (0 為不存在，1 為存在)

m ：BigM 係數，當 P_{ij}^{SF} 等於 1 時，為一個限制不住 Q_{ij}^{SF} 的係數

(2) 製造廠至配銷中心之運輸

當製造廠節點 j 運輸至訂單節點 k，若此兩節點間的運輸存在時，數量必須大於或等於 10，如果兩節點間沒有運輸存在時則數量為 0，因此限制式中必須要加入等於 0 或 1 的整數變數(P_{jk}^{FD})，其限制式如下所示：

$$\begin{aligned} &10\times P_{jk}^{FD} \le Q_{jk}^{FD} \\ &Q_{jk}^{FD} \le m\times P_{jk}^{FD} \\ &\forall j=1,2,...,J \quad \forall k=1,2,...,K \end{aligned} \tag{16}$$

Q_{jk}^{FD} ：製造廠節點 j 供給至訂單節點 k 的數量

P_{jk}^{FD} ：0 或 1 的整數變數，代表著製造廠節點 j 至訂單節點 k 的運輸是否存在 (0 為不存在，1 為存在)

m ：BigM 係數，當 P_{jk}^{FD} 等於 1 時，為一個限制不住 Q_{jk}^{FD} 的係數

6. 限制式六：供應商出貨時間限制

供應商節點 i 的出貨時間(T_i^S)大於等於運輸存在的供應商作業前置時間($P_{ij}^{SF} \times t_i^{SP}$)，小於等於供應商結點 i 的生產前置時間(t_i^{SP})加上存貨週期時間(t_i^{SI})，如果供應商節點 i 的運輸完全不存在時($\sum_{j=1}^{J} P_{ij}^{SF} = 0$)，供應商節點 i 的出貨時間為 0($T_i^S = 0$)，其限制式如下所示：

$$\begin{aligned} & P_{ij}^{SF} \times t_i^{SP} \le T_i^S \quad \forall i = 1,2,...,I \quad \forall j = 1,2,...,J \\ & T_i^S \le t_i^{SP} + t_i^{SI} \quad \forall i = 1,2,...,I \\ & T_i^S \le m \times \sum_{j=1}^{J} P_{ij}^{SF} \quad \forall i = 1,2,...,I \end{aligned} \tag{17}$$

T_i^S：供應商節點 i 的出貨時間

t_i^{SP}：供應商節點 i 之生產前置時間

t_i^{SI}：供應商節點 i 之存貨週期時間

P_{ij}^{SF}：0 或 1 的整數變數，代表著供應商節點 i 至製造廠節點 j 的運輸是否存在 (0 為不存在，1 為存在)

m：BigM 係數，當 $\sum_{j=1}^{J} P_{ij}^{SF}$ 大於等於 1 時，為一個限制不住 T_i^S 的係數

7. 限制式七：製造廠出貨時間限制

製造廠節點 j 的出貨時間(T_j^F)大於等於運輸存在的供應商節點 i 出貨時間(T_i^S)加上運輸至製造廠節點 j 的作業前置時間(t_{ij}^{SF})；並且製造廠節點 j 的出貨時間(T_j^F)加上運輸至訂單節點 k 的作業前置時間(t_{jk}^{FD})小於等於訂單節點 k 的訂單交貨時間(t_k^{SHIP})，如果製造商節點 j 的運輸不存在時($\sum_{k=1}^{K} P_{jk}^{FD} = 0$)，製造商節點 j 的出貨時間為 0($T_j^F = 0$)，其限制式如下所示：

$$\begin{aligned} & T_j^F \ge (T_i^S + t_{ij}^{SF}) - m \times (1 - P_{ij}^{SF}) \quad \forall i = 1,2,...,I \quad \forall j = 1,2,...,J \\ & T_j^F \le m \times \sum_{k=1}^{K} P_{jk}^{FD} \quad \forall j = 1,2,...J \\ & t_k^{SHIP} \ge (T_j^F + t_{jk}^{FD}) - m \times (1 - P_{jk}^{FD}) \quad \forall j = 1,2,...,J \quad \forall k = 1,2,...K \end{aligned} \tag{18}$$

T_i^S：供應商節點 i 的出貨時間

T_j^F：製造廠節點 j 的出貨時間

t_k^{SHIP}：訂單節點 k 的交期時間

t_{ij}^{SF}：供應商節點 i 運輸至製造廠節點 j 的作業前置時間

t_{jk}^{FD}：製造廠節點 j 運輸至訂單節點 k 的作業前置時間

P_{ij}^{SF}：0 或 1 的整數變數，代表著供應商節點 i 至製造廠節點 j 的運輸是否存在 (0 為不存在，1 為存在)

P_{jk}^{FD}：0 或 1 的整數變數，代表著製造廠節點 j 至訂單節點 k 的運輸是否存在 (0 為不存在，1 為存在)

m：BigM 係數，當 $\sum_{k=1}^{K} P_{jk}^{FD}$ 大於等於 1 或者 P_{ij}^{SF} 、P_{jk}^{FD} 等於 0 時，為一個限制不住 T_j^F 的係數

此節所介紹的為記憶體模組產業供應網絡的數學模型，利用數理規畫工具針對此 INLP 模式的數學模型求解過程中，複雜度會隨著規模變大而大幅度的增加，而供應網絡會隨著時間的不同而各項成本會跟著改變，因此求解時間過長將會導致規劃結果無法使用，在下節中將介紹如何將循序程式平行化和分散式物件系統的建構，利用此分散式平行系統達到平行計算的目的，解決求解時間上的問題，並且達到整體的最佳規劃結果。

3.3 分散式平行系統之架構

在之前介紹了記憶體模組產業供應網絡生產規畫的現況方法，在採用這種生產規劃方法下，無法得到整體的最佳規畫。於是再將目前記憶體模組產業供應網絡之生產規劃模式以視覺化圖形的方式表達出來，並且以此建立供應網絡之數學模型，針對此數學模型求解將可以得到全域最佳解，但由於求解時間會隨著規模變大而造成 *NP-Complete* 的問題。因此在此節將會介紹如何針對之前的數學模型做平行化的動作，以及在平行化後會產生哪些問題而必須要加入可行的演算法，然後再透過分散式物件的方法建構出分散式平行系統。

本研究在系統的建構是採用 JAVA 和 LINGO 的結合，在 LINGO 線性規劃軟體中提供了動態聯結資料庫(Dynamic Link Library, DLL)的功能，讓使用者可以動態的聯結至不同應用程式之資料庫以存取其資料。

3.3.1 循序程式的平行化

在分散式平行系統中，為了讓各個計算節點(Computation Node)能夠對於不同的資料進行平行計算，伺服端必須要將資料分割然後分配給每一個計算結點進行平行計算，在每個計算結點分別算完所有資料後匯整，計算出在此種資料分割方式下所求得的最佳規劃結果，以此結果來供規劃人員參考。本研究所採用的資料分割方式為將全域供應網絡的規模分割為多個規模較小的區域供應網絡之組合，其分割方式如圖 3.5 所示：

全域	
供應商節點編號	製造廠節點編號
1, 2, 3, 4, 5, 6	1, 2, 3, 4, 5, 6

資料分割

	區域	
No.	供應商節點編號	製造廠節點編號
1	1, 2, 3, 4, 5, 6	1, 2, 3, 4
2	1, 2, 3, 4, 5, 6	1, 2, 4, 5
3	1, 2, 3, 4, 5, 6	1, 2, 5, 6
4	1, 2, 3, 4, 5, 6	1, 3, 4, 5
5	1, 2, 3, 4, 5, 6	1, 3, 4, 6
…	…	…
15	1, 2, 3, 4, 5, 6	3, 4, 5, 6

圖 3.5、資料的平行處理

上圖中假設供應網絡中第一階和第二階的供應商及製造廠分別有 6 家，而資料分割的方式採用供應商選 6 家和製造廠選 4 家，因此會產生出 15($C_6^6 \times C_4^6 = 15$)種組合方式的區域供應網絡，伺服端在分割完資料後是採用一次傳送一組資料給計算結點。

伺服端透過遠端程序呼叫的方式，透過 JavaRMI 將分割完的資料，以物件的方式分配給計算節點端做計算的動作，計算節點端在接收伺服端所傳送的資料後進行計算，並且將計算完的結果回傳給伺服端，再進行下一組資料的計算，其架構如圖 3.6 所示：

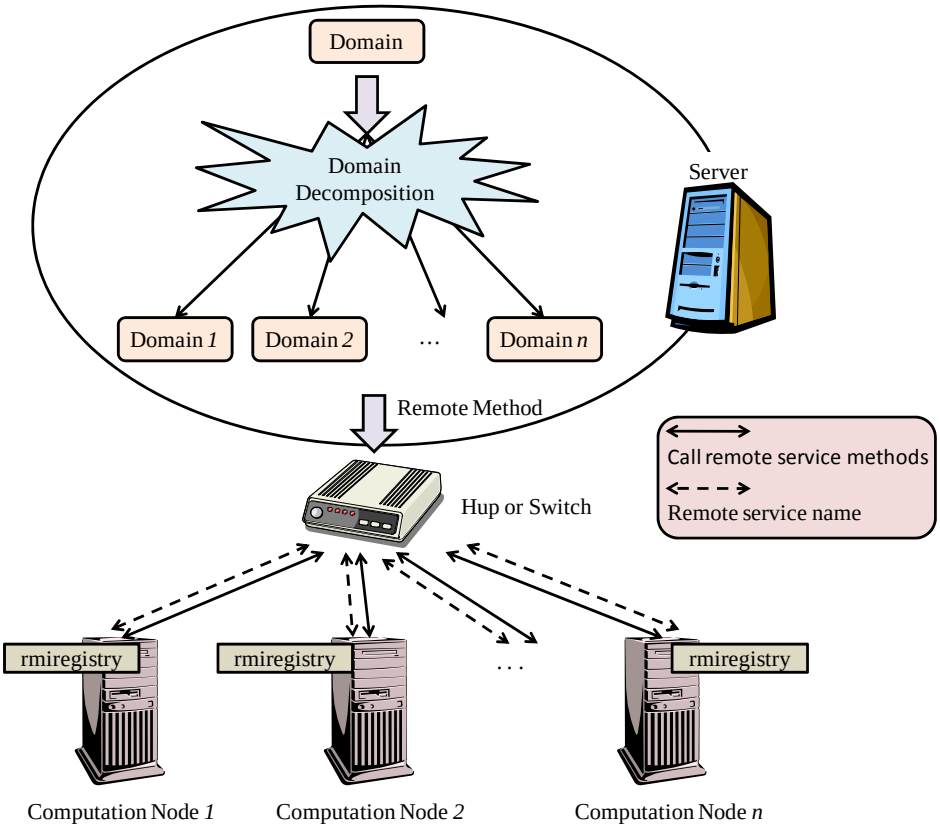


圖 3.6、分散式平行系統之架構示意圖

在伺服端中，呼叫計算節點端物件的方式是採用多執行緒(Multithreading)的設計，如此可以不用等待前一個計算節點端計算完畢後才能呼叫第二個計算節點端進行計算，而執行緒中判斷各個計算節點斷是否計算完畢，則是採用此類別中的 *isAlive()*方法，此方法回傳的值為布林值(boolean)；當 *isAlive()* 方法物件回傳 *true* 時，代表著計算節點端物件存在並且在計算中，因此伺服端不會進行分配下一組資料給計算節點進行計算；當 *isAlive()*方法物件回傳 *false* 時，代表著計算節點已經計算完畢並且把計算結果回傳給伺服端，目前物件已經結束不存在，此時伺服端就開始判別是否還有資料要給計算節點計算的，如果有的話將會分配資料給計算節點計算。

如果伺服端已經沒有資料要分配給計算節端點進行計算時，伺服端此時也只需要等待所有的計算節點端計算完畢(所有的 *isAlive()*方法物件都回傳 *false*)並且回傳計算結果給伺服端，在接收所有的計算結果後針對目標值進行排序並且取出目標值最佳的規劃結果，其伺服端及計算節點端的計算流程如圖 3.7 和圖 3.8 所示。

伺服端計算流程中存在著許與計算節點端間物件的互動，在 JavaRMI

必須在計算節點端必須事先定義介面，並且類別檔必須執行介面，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。參數 *nI* 為供應商的數量，*nJ* 為製造廠的數量。
3. *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。
4. *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。

計算節點端的類別檔執行介面後，伺服端的程式才能利用 RMI 的方式進行物件上的呼叫，在下述流程中所列出來的函數(Function)，即為是先前在介面中所定義的方法物件，而計算完的結果是以位元陣列的方式回傳給伺服端。

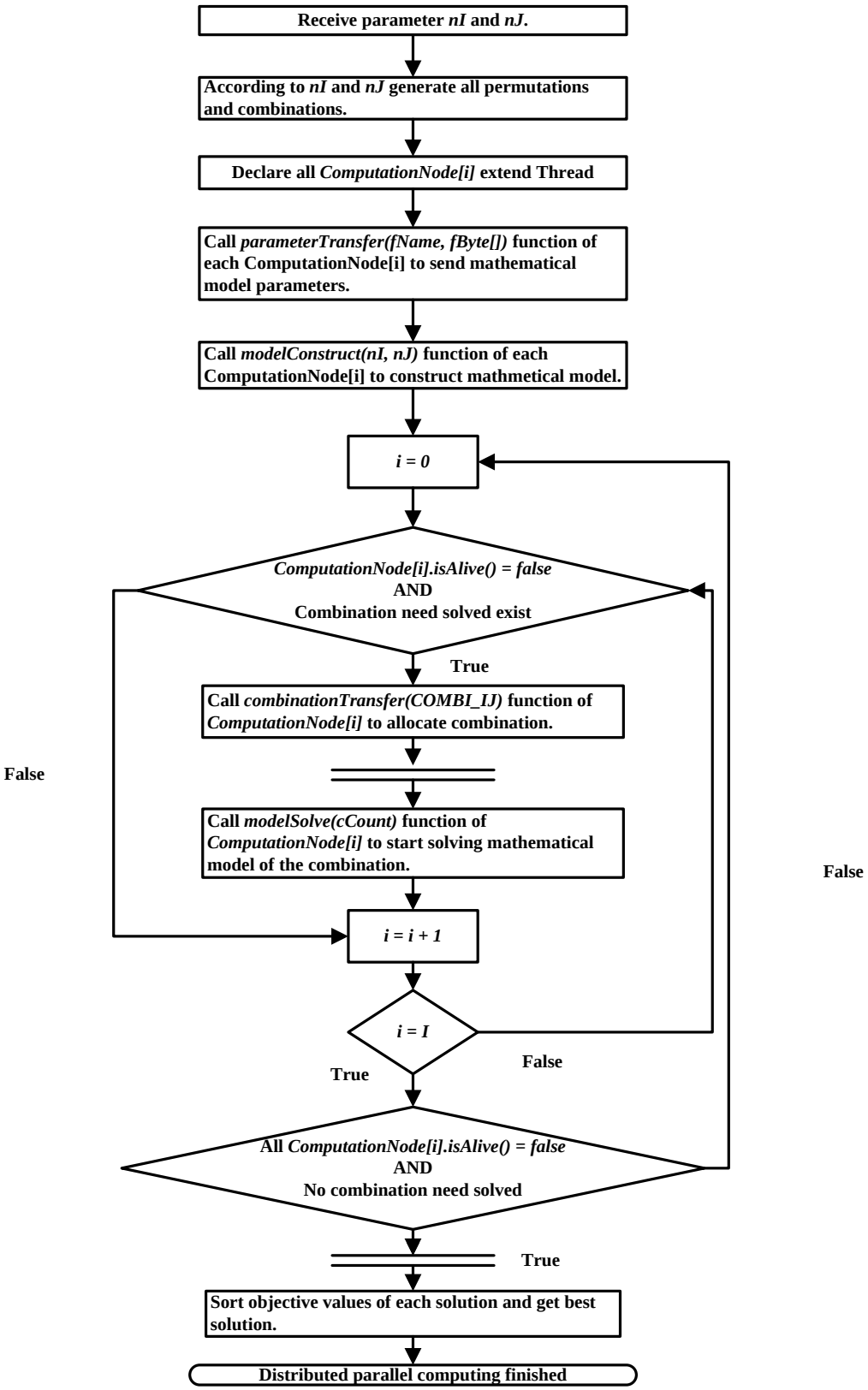


圖 3.7、伺服端的計算流程

接下來將予以說明伺服端的計算流程：

1. 輸入參數 *nI* 及 *nJ*。
2. 根據供應商數量(*nI*)和製造廠數量(*nJ*)產生所有的排列組合。
3. 計算節點 *i(I = 0,1,…,(I – 1))*繼承執行緒類別物件。
4. 呼叫每一個計算節點 *i* 的 *parameterTransfer(fName, fByte[])*方法物件，傳送數學模型的參數給每一個計算節點 *i*，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。
5. 呼叫每一個計算節點 *i* 的 *modelConstruct(nI, nJ)*方法物件，傳送參數給每一個計算節點 *i* 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。參數 *nI* 為供應商的數量，*nJ* 為製造廠的數量。
6. 設定 *i = 0*。
7. 計算節點 *i* 的物件是否不存在且還有組合未計算，判別結果為是的話，進入步驟 8，反之則進入步驟 10。
8. 呼叫計算節點 *i* 的 *combinationTransfer(COMBI_IJ)*方法物件，分配未計算的組合給計算節點 *i*。參數 *COMBI_IJ* 為未計算的組合所轉換成的字串。
9. 呼叫計算節點 *i* 的 *modelSolve(cCount)*方法物件，並針對步驟 8 所接收到的組合進行分散式平行計算。參數 *cCount* 為組合的編號，為用來建立計算結果的檔案名稱的參數。
10. 對 *i* 進行累加(*i = i + 1*)。
11. 是否 *i = I*，判別結果為是的話，進入步驟 12，反之則進入步驟 7。
12. 所有的計算節點 *i* 物件是否都不存在且所有組合都計算完畢了，判別結果為是的話，進入步驟 13，反之進入步驟 6。
13. 對所有組合的計算結果進行排序並取出最大值。
14. 伺服端的計算流程結束。

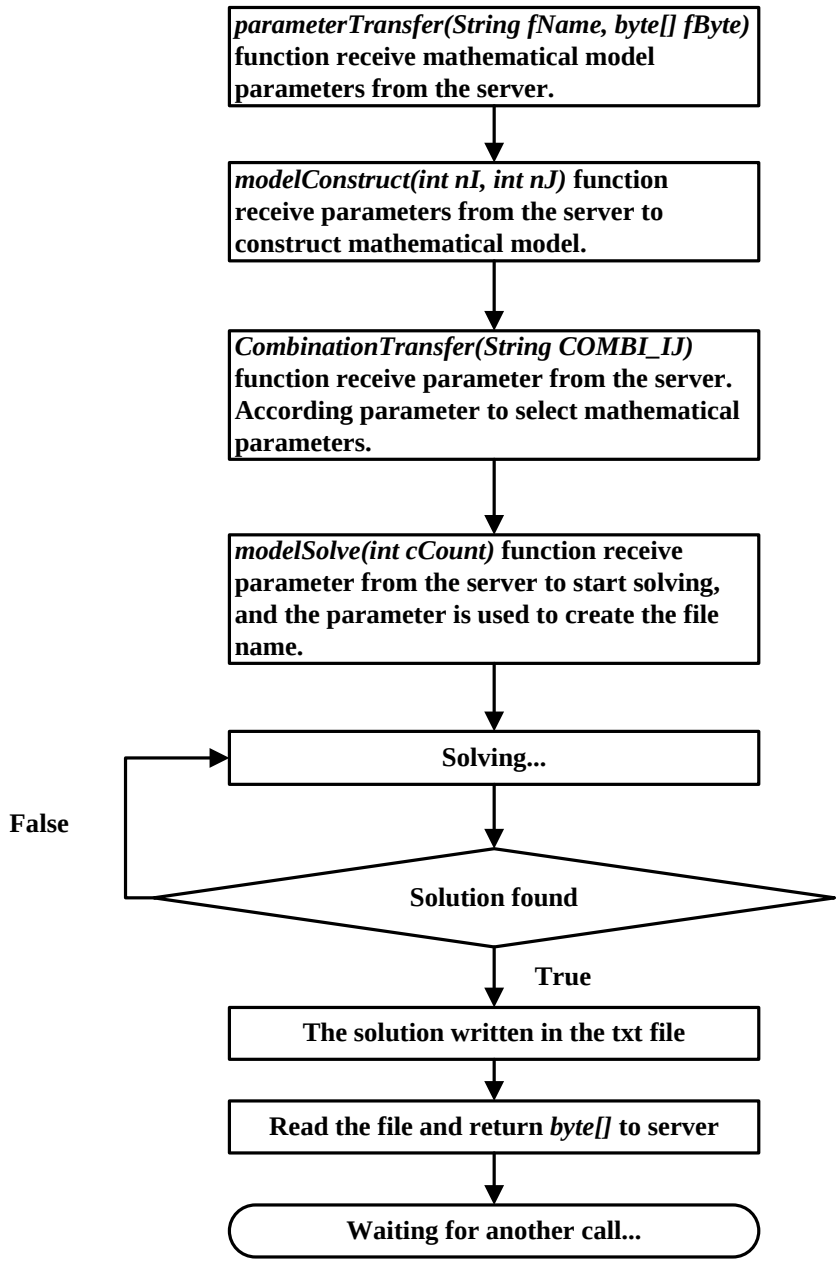


圖 3.8、計算節點端的計算流程

接下來將予以說明計算節點端的計算流程：

1. *parameterTransfer(String fName, byte[] fByte)*方法物件接收伺服端傳送數學模型的參數，本研究所採用的方式為數學模型的參數寫在文字檔內。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. *modelConstruct(int nI, int nJ)*方法物件接收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。參數 *nI* 為供應商的數量，*nJ* 為製造廠的數量。
3. *combinationTransfer(String COMBI_IJ)* 方法物件接收伺服端分配的組

合，計算節點端將會依照此組合讀取數學模型的參數。參數 *COMBI_IJ* 依照組合方式所轉換成的字串。

4. *modelSolve(int cCount)*方法物件接收伺服端傳送的參數後，計算節點端針對步驟 3 所接收的組合進行數學模型的求解。參數 *cCount* 為組合的編號，用來建立文字檔的檔案名稱。
5. LINGO 增加疊代(Iteration)對數學模型求解。
6. LINGO 是否已經計算完畢，判別結果為是的話，進入步驟 7，反之則進入步驟 5。
7. *modelSolve(int cCount)*方法物件將計算結果寫入文字檔。
8. *modelSolve(int cCount)*方法物件將文字檔轉換成位元陣列並且回傳給伺服端。
9. 計算節點端結束計算程序，等待伺服端的呼叫。

根據上述的平行計算的流程，採用 6 家供應商取 6 家、6 家製造廠取 4 家進行平行計算，總計算時間為 10435 秒，其 15 組計算結果如下表：

表 3.2、平行計算的結果

No.	Runtime(s)	Objective value
1	1420	43173
2	254	44457
3	1101	43173
4	76	44457
5	253	42871
6	34	44457
7	5562	43495
8	10434	42211
9	6249	43226
10	1954	43495
11	182	44457
12	2512	42623
13	119	44457
14	30	44457
15	2202	43495

由上表的第 8 個組合方式的計算結果可知，單純的採用全域供應網絡分割成規模較小的區域供應網絡的方式來進行計算時，可能會分割出更多複雜的數學模型，而這些數學模型計算出來的結果意義也不大，因此在進

行平行計算的過程中，必須要存在一個演算法來控制這些複雜且沒有任何意義的數學模型，讓各個資料之間彼此有互動的關係存在，如此才能讓平行計算在實務上變得有意義。下一節中將會介紹此 INLP 模型在 LINGO 中的運作方式，並以此方式推演出一個可行的演算法，讓平行計算顯得更有效率，更能解決全域最佳解計算時間上的問題。

3.3.2 平行計算之演算法

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 在計算過程中上界(Upper Bound)和下界(Lower Bound)的資訊，根據種方法把這兩種資訊隨著時間改變而記錄下來，並且作一些資料上的整理，其結果如下圖(橫軸為時間(秒)、縱軸為值)：

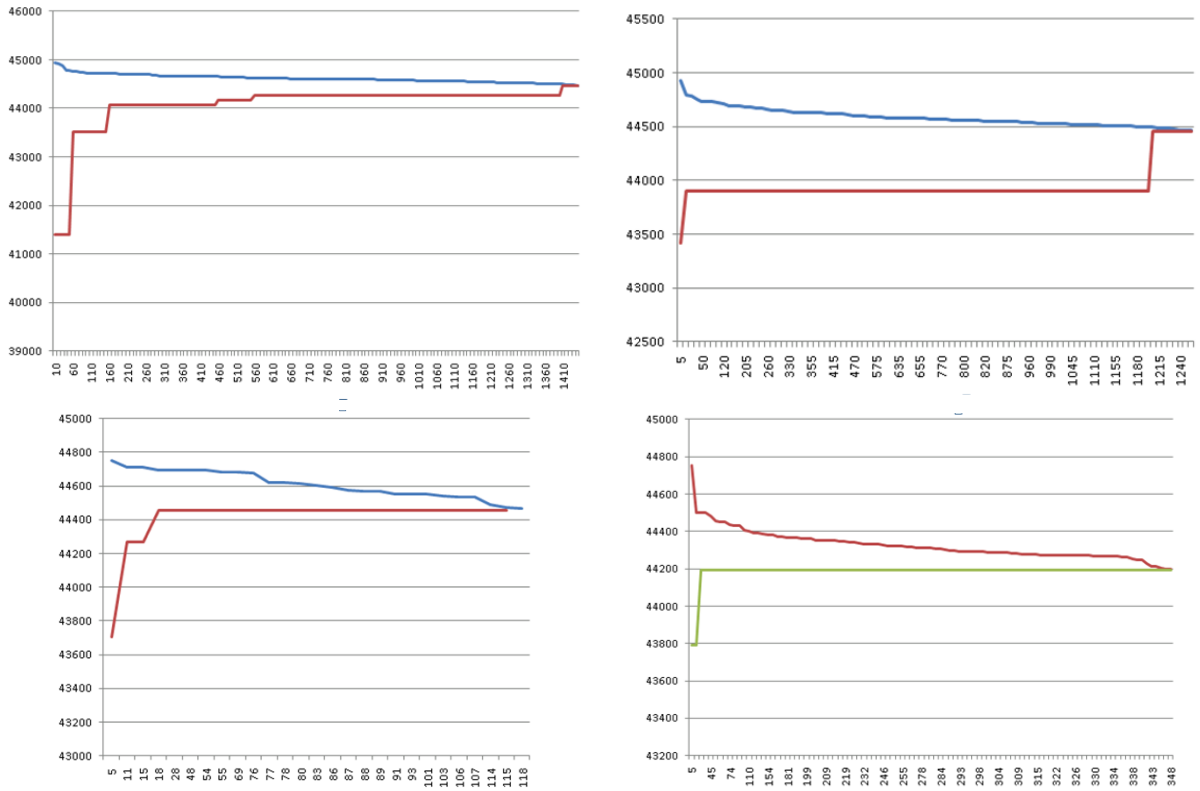


圖 3.9、LINGO 計算過程中的上界和下界

本研究挑選了四種模式分別說明上界和下界的變化，第一種模式為圖左上方的模式，其下界會隨著計算過程中呈階梯狀不斷的上升，而上界呈現緩慢的收斂；第二種模式為圖右上方的模式，其下界在計算的過程中並沒有任何的改變，而是在上界到達一定的值以後快速的上升，上界的部分

也呈現出緩慢的收斂；第三種模式為圖左下方的模式，其下界在剛開始就快速上升到一定的值，而上界也呈現出快速的收斂；第四種模式為圖右下方的模式，下界在剛開始就快速上升到一定的值，而上界呈現緩慢的收斂。

上圖中上界和下界交會的地方即為針對數學模型所求得的目標值，如果要從上界的收斂程度來停止計算的話，那可能會拒絕掉很多有意義的解，假使要從上界和下界的差距來停止計算時亦是如此，因此本研究提出了集合的概念來讓平行計算的效率更佳，至於集合的概念如下圖：

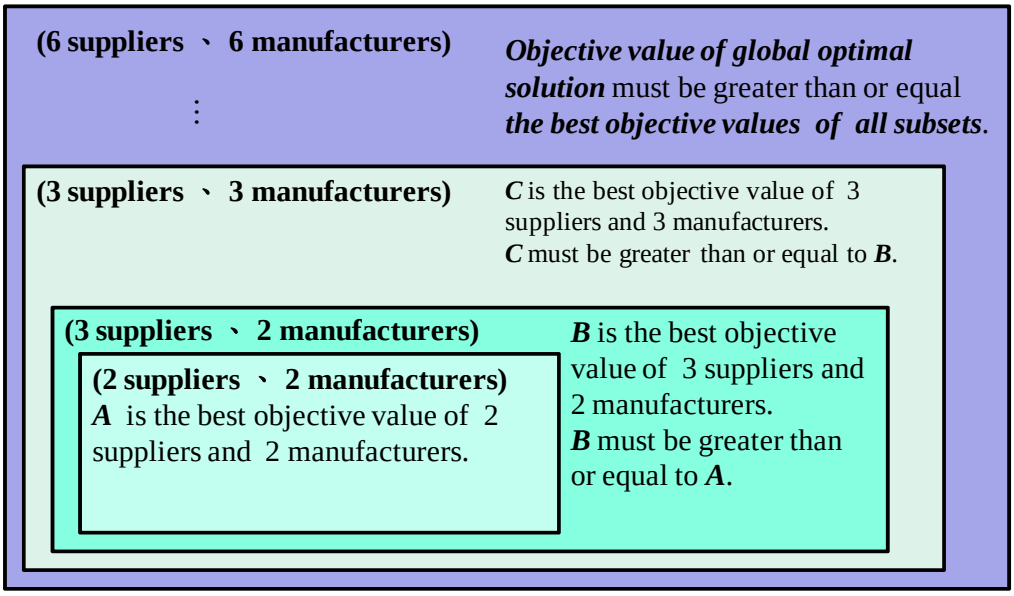


圖 3.10、數學模型之集合的概念

上圖中說明了對於數學模型求解時，供應商 3 家和製造廠 2 家所產生的 300 組計算結果中，其計算結果中的最佳規劃結果之目標值一定會優於或等於供應商 2 家和製造廠 2 家所產生的 225 組計算結果中的最佳規劃結果之目標值，以此類推，規模比較大的集合所求出來的最佳規劃結果之目標值一定會優於或等於子集合的最佳規劃結果之目標值。用此種方式，可以先對子集合求出最佳規劃結果之目標值，再接著對規模比較大的集合求解時，我們可以設定當規模比較大的數學模型在進行平行計算時，其中有幾組數學模型計算出來的上界已經小於子集合的最佳規劃結果之目標值時，這些數學模型的目標值也一定小於子集合的最佳規劃結果之目標值，因此這些組合在實質上就沒有任何意義存在，計算節點端將停止對於這些組合的計算，以此讓平行計算提高計算上的效率。

因此針對此邏輯所推衍出來的演算法，為一剛開始伺服端就傳送子集

合的最佳規劃結果之目標值給計算節點端，計算節點端在計算各個組合的過程中，判別上界已經小於子集合的最佳規劃結果之目標值時，就會終止計算並且再取得下一組資料進行下一個組合的計算，其概念如圖 3.11 所示：

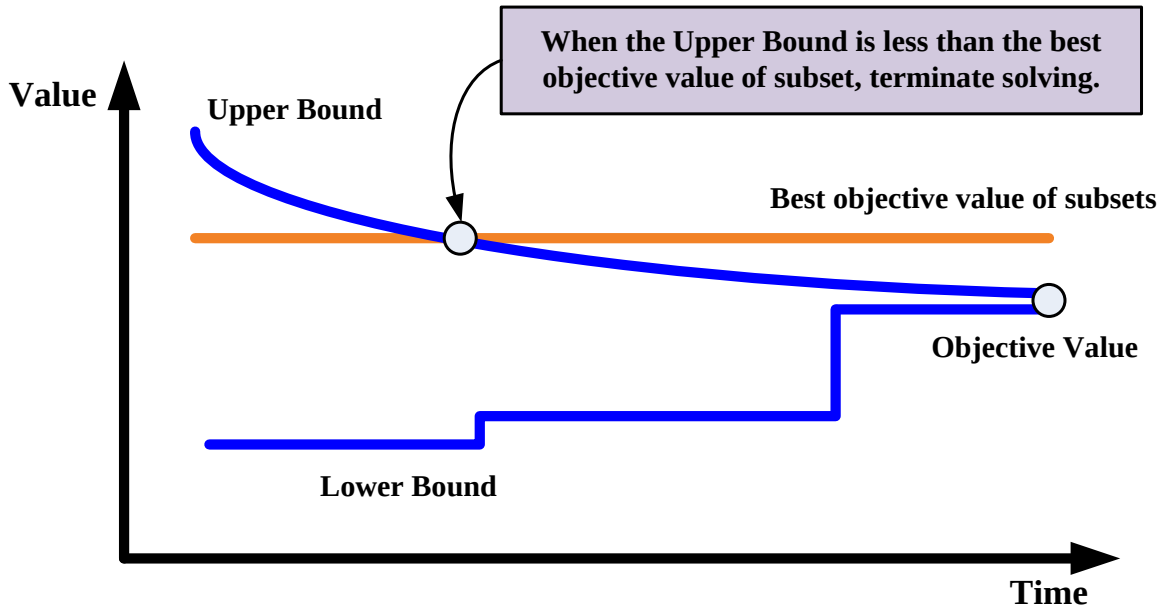


圖 3.11、子集合的最佳規劃結果之目標值來控制上界

根據上述所加入的控制上界的概念，在計算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計算節點端的程式中，其伺服端與計算節點端的流程如圖 3.12 與圖 3.13 所示。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。參數 *nI* 為供應商的數量，*nJ* 為製造廠的數量。
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。

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。

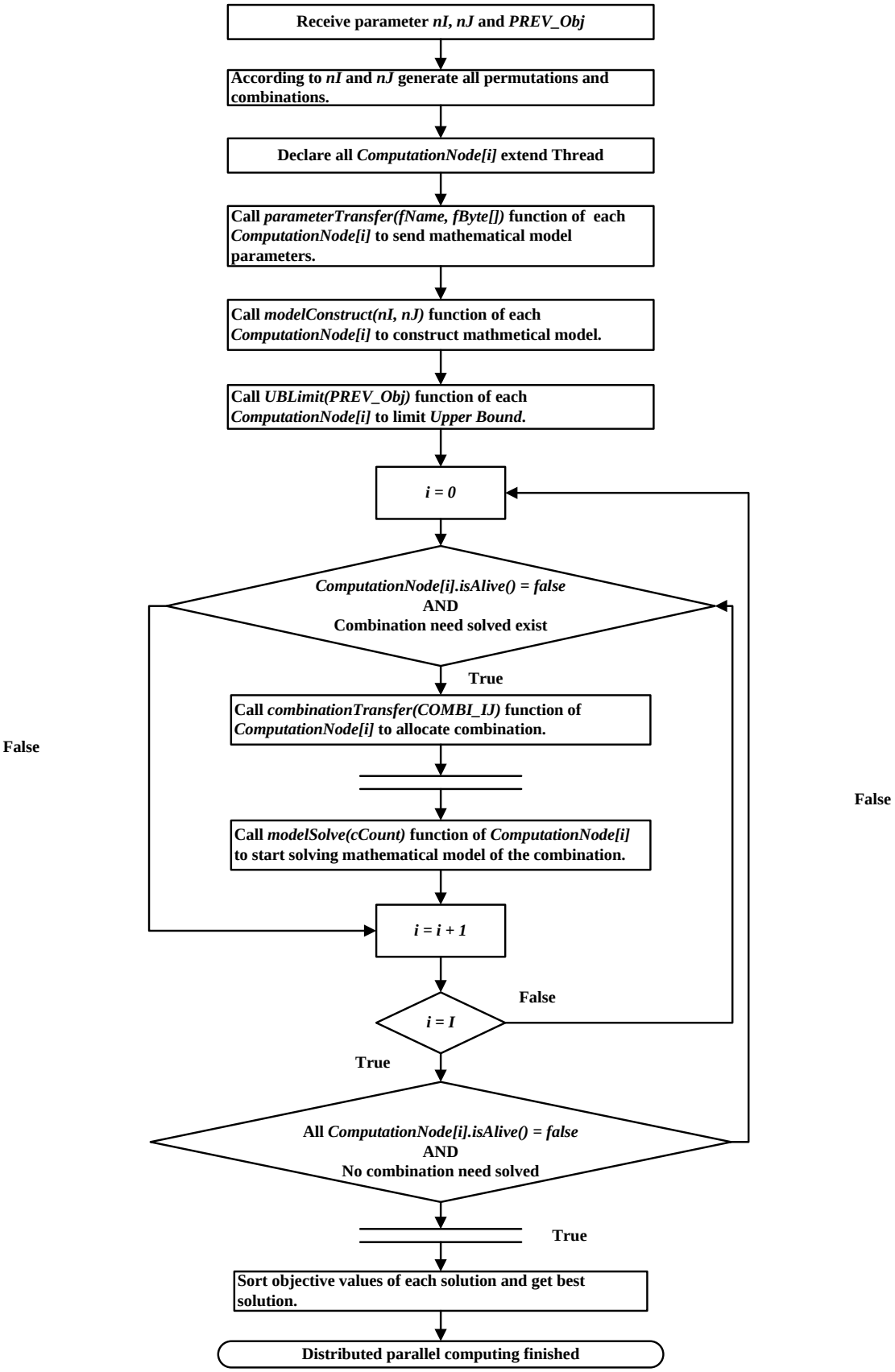


圖 3.12、伺服端的計算流程(子集合之最佳規劃結果控制上界)

接下來將予以說明伺服端的計算流程：

1. 輸入參數 *nI*、*nJ* 和 *PREV_Obj*。
2. 根據供應商數量(*nI*)和製造廠數量(*nJ*)產生所有的排列組合。
3. 計算節點 *i(I = 0,1,…,(I – 1))*繼承執行緒類別物件。
4. 呼叫每一個計算節點 *i* 的 *parameterTransfer(fName, fByte[])*方法物件，傳送數學模型的參數給每一個計算節點 *i*，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。
5. 呼叫每一個計算節點 *i* 的 *modelConstruct(nI, nJ)*方法物件，傳送參數給每一個計算節點 *i* 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。參數 *nI* 為供應商的數量，*nJ* 為製造廠的數量。
6. 呼叫每一個計算節點 *i* 的 *UBLimit(PREV_Obj)*，傳送子集合的最佳規劃結果之目標值給各個計算節點 *i*，以此 *PREV_Obj* 的值來限制住上界的值。參數 *PREV_Obj* 為子集合的最佳規劃結果之目標值。
7. 設定 *i = 0*。
8. 計算節點 *i* 的物件是否不存在且還有組合未計算，判別結果為是的話，進入步驟 9，反之則進入步驟 11。
9. 呼叫計算節點 *i* 的 *combinationTransfer(COMBI_IJ)*方法物件，分配未計算的組合給計算節點 *i*。參數 *COMBI_IJ* 為未計算的組合所轉換成的字串。
10. 呼叫計算節點 *i* 的 *modelSolve(cCount)*方法物件，並針對步驟 9 所接收到的組合進行分散式平行計算。參數 *cCount* 為組合的編號，為用來建立計算結果的檔案名稱的參數。
11. 對 *i* 進行累加(*i = i + 1*)。
12. 是否 *i = I*，判別結果為是的話，進入步驟 13，反之則進入步驟 8。
13. 所有的計算節點 *i* 物件是否都不存在且所有組合都計算完畢了，判別結果為是的話，進入步驟 14，反之進入步驟 7。
14. 對所有組合的計算結果進行排序並取出最大值。
15. 伺服端的計算流程結束。

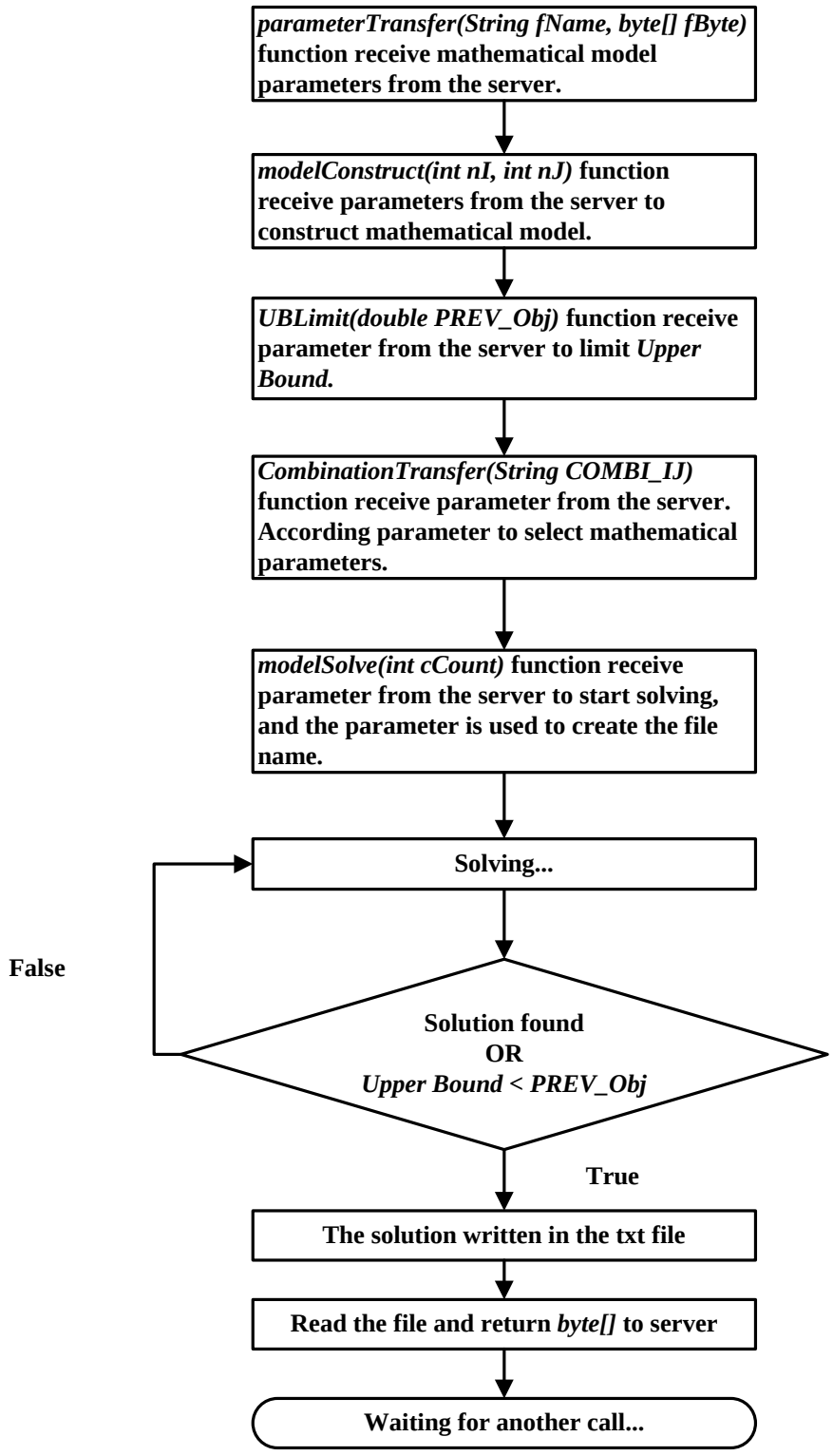


圖 3.13、計算節點端的計算流程(子集合之最佳規劃結果控制上界)

接下來將予以說明計算節點端的計算流程：

1. *parameterTransfer(String fName, byte[] fByte)*方法物件接收伺服端傳送數學模型的參數，本研究所採用的方式為數學模型的參數寫在文字檔內。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. *modelConstruct(int nI, int nJ)*方法物件接收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。參數 *nI* 為供應商的數量，*nJ* 為製造廠的數量。
3. *UBLimit(double PREV_Obj)*方法物件接收伺服端傳送的參數限制 *Upper Bound*。參數 *PREV_Obj* 為子集合的最佳規劃結果之目標值。
4. *combinationTransfer(String COMBI_IJ)* 方法物件接收伺服端分配的組合，計算節點端將會依照此組合讀取數學模型的參數。參數 *COMBI_IJ* 依照組合方式所轉換成的字串。
5. *modelSolve(int cCount)*方法物件接收伺服端傳送的參數後，計算節點端針對步驟 4 所接收的組合進行數學模型的求解。參數 *cCount* 為組合的編號，用來建立文字檔的檔案名稱。
6. LINGO 增加疊代(Iteration)對數學模型求解。
7. LINGO 是否已經計算完畢或者上界的值小於 *PREV_Obj* 的值時，判別結果為是的話，進入步驟 8，反之則進入步驟 6。
8. *modelSolve(int cCount)*方法物件將計算結果寫入文字檔。
9. *modelSolve(int cCount)*方法物件將文字檔轉換成位元陣列並且回傳給伺服端。
10. 計算節點端結束計算程序，等待伺服端的呼叫。

本研究繼續擴大應用此演算法，因為在平行計算的過程中，伺服端會不斷的接收計算節點計算完的結果，假如已經計算完的結果中，存在著目標值大於子集合的最佳規劃結果之目標值時，伺服端將會採取此更佳的目標值來控制計算節點的上界，因此伺服端在計算過程中會一直不斷的判別是否有更佳的目標值存在，計算節點端也會在計算中接收到伺服端傳送的更佳目標值，上界也可能因為限制的改變，而突然中止計算節點端對於此組合的計算，其概念如圖 3.14 所示：

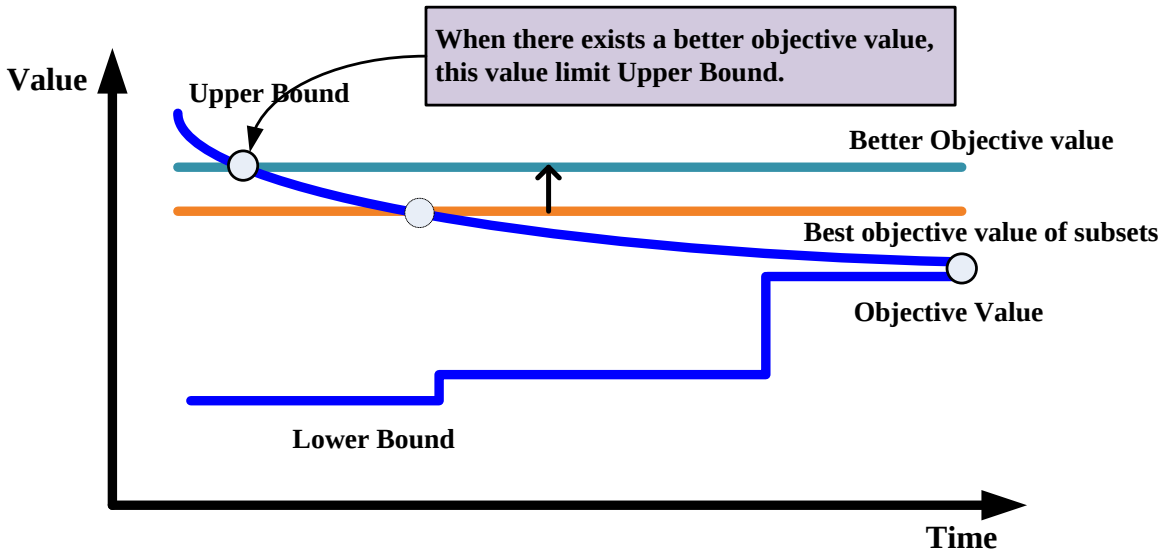


圖 3.14、平行計算中得到的更佳目標值來控制上界

根據上述以子集合的最佳規劃結果之目標值，以及平行計算過程中所求得的更佳目標值來控制上界的概念，將這種控制機制寫入伺服端與計算節點端的程式中，其流程如下圖 3.15 與圖 3.16 所示。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。參數 *nI* 為供應商的數量，*nJ* 為製造廠的數量。
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更佳目標值。
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。

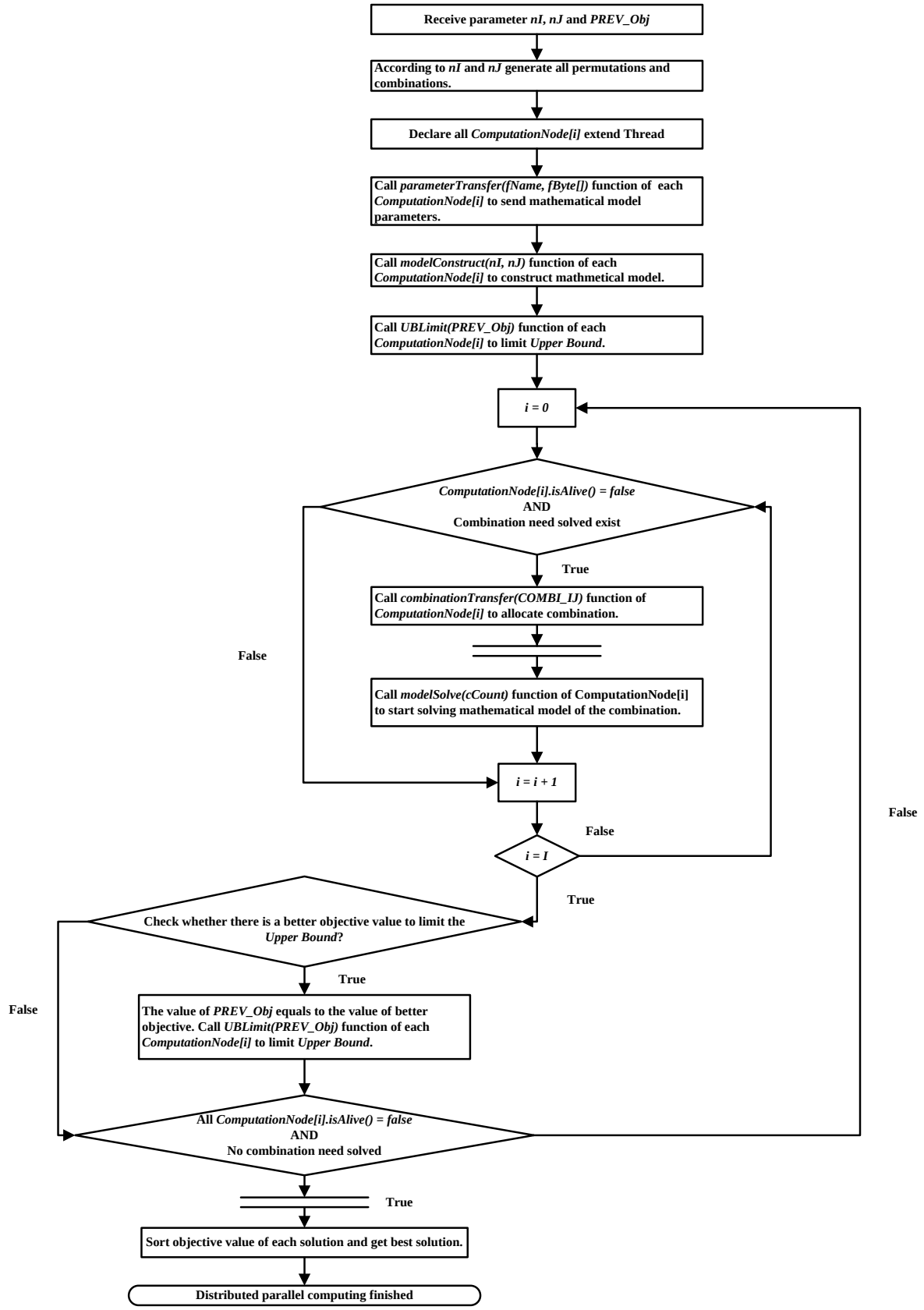


圖 3.15、伺服端的計算流程(以更佳的目標值控制上界)

接下來將予以說明伺服端的計算流程：

1. 輸入參數 *nI*、*nJ* 和 *PREV_Obj*。
2. 根據供應商數量(*nI*)和製造廠數量(*nJ*)產生所有的排列組合。
3. 計算節點 *i(I = 0,1,…,(I – 1))*繼承執行緒類別物件。
4. 呼叫每一個計算節點 *i* 的 *parameterTransfer(fName, fByte[])*方法物件，傳送數學模型的參數給每一個計算節點 *i*，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。
5. 呼叫每一個計算節點 *i* 的 *modelConstruct(nI, nJ)*方法物件，傳送參數給每一個計算節點 *i* 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。參數 *nI* 為供應商的數量，*nJ* 為製造廠的數量。
6. 呼叫每一個計算節點 *i* 的 *UBLimit(PREV_Obj)*，傳送子集合的最佳規劃結果之目標值給各個計算節點 *i*，以此 *PREV_Obj* 的值來限制住上界的值。參數 *PREV_Obj* 為子集合的最佳規劃結果之目標值。
7. 設定 *i = 0*。
8. 計算節點 *i* 的物件是否不存在且還有組合未計算，判別結果為是的話，進入步驟 9，反之則進入步驟 11。
9. 呼叫計算節點 *i* 的 *combinationTransfer(COMBI_IJ)*方法物件，分配未計算的組合給計算節點 *i*。參數 *COMBI_IJ* 為未計算的組合所轉換成的字串。
10. 呼叫計算節點 *i* 的 *modelSolve(cCount)*方法物件，並針對步驟 9 所接收到的組合進行分散式平行計算。參數 *cCount* 為組合的編號，為用來建立計算結果的檔案名稱的參數。
11. 對 *i* 進行累加(*i = i + 1*)。
12. 是否 *i = I*，判別結果為是的話，進入步驟 13，反之則進入步驟 8。
13. 伺服端對目前已經計算完的結果，判斷是否有更好的目標值來限制上界，判別結果為是的話，進入步驟 14，反之進入步驟 15。
14. *PREV_Obj* 的值為計算中所得到的更佳目標值，呼叫每一個計算節點 *i* 的 *UBLimit(PREV_Obj)*方法物件，傳送更佳的目標值給各個計算節點 *i*，用來限制住上界。
15. 所有的計算節點 *i* 物件是否都不存在且所有組合都計算完畢了，判別結

果為是的話，進入步驟 16，反之進入步驟 7。

16. 對所有組合的計算結果進行排序並取出最大值。

17. 伺服端的計算流程結束。

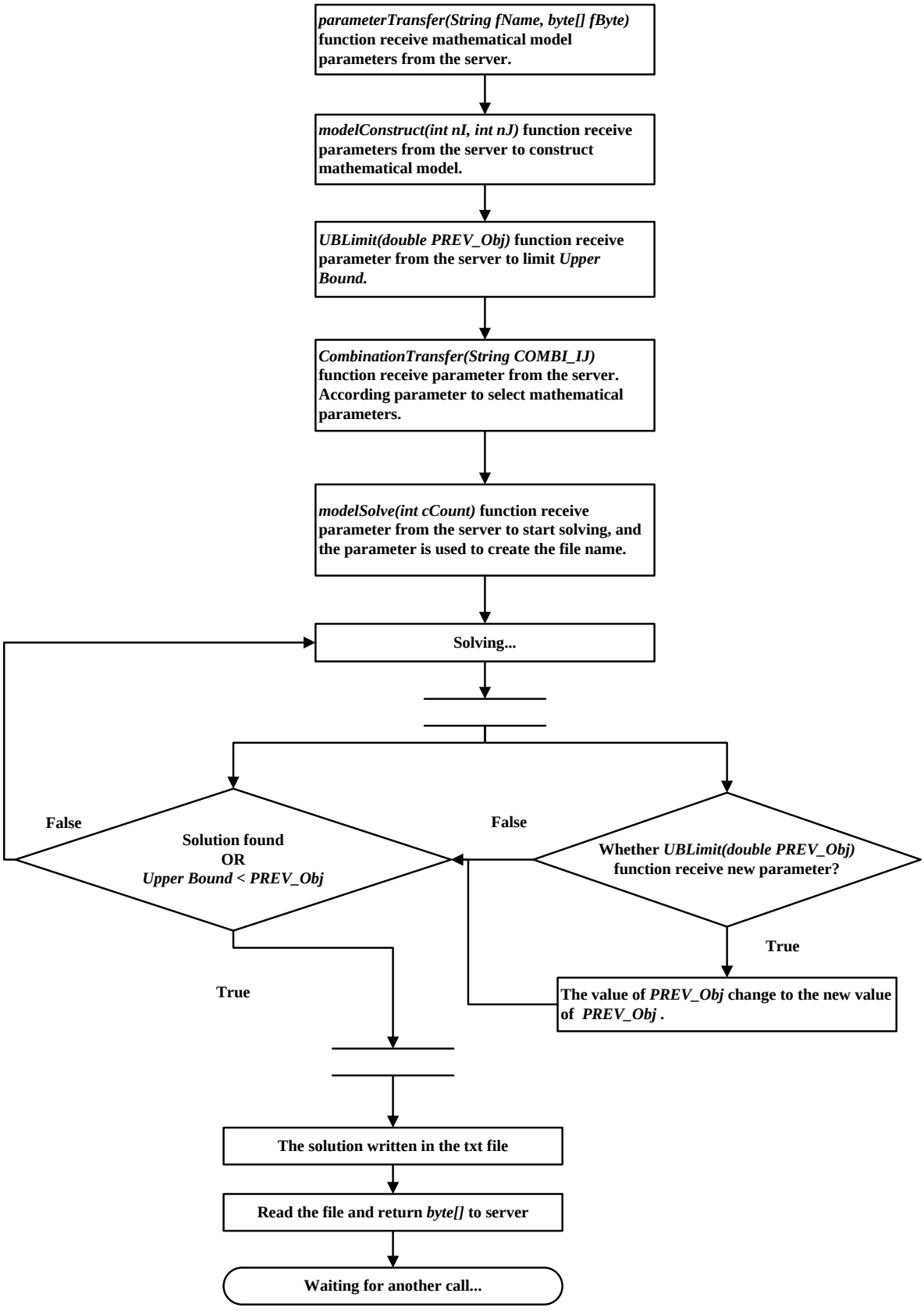


圖 3.16、計算節點端的計算流程(以更佳的目標值控制上界)

接下來將予以說明計算節點端的計算流程：

1. *parameterTransfer(String fName, byte[] fByte)*方法物件接收伺服端傳送數學模型的參數，本研究所採用的方式為數學模型的參數寫在文字檔內。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. *modelConstruct(int nI, int nJ)*方法物件接收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。參數 *nI* 為供應商的數量，*nJ* 為製造廠的數量。
3. *UBLimit(double PREV_Obj)*方法物件接收伺服端傳送的參數限制 *Upper Bound*。參數 *PREV_Obj* 為子集合的最佳規劃結果之目標值。
4. *combinationTransfer(String COMBI_IJ)* 方法物件接收伺服端分配的組合，計算節點端將會依照此組合讀取數學模型的參數。參數 *COMBI_IJ* 依照組合方式所轉換成的字串。
5. *modelSolve(int cCount)*方法物件接收伺服端傳送的參數後，計算節點端針對步驟 4 所接收的組合進行數學模型的求解。參數 *cCount* 為組合的編號，用來建立文字檔的檔案名稱。
6. LINGO 增加疊代(Iteration)對數學模型求解，接下來的步驟 7 和步驟 8 為平行的程序。
7. LINGO 是否已經計算完畢或者上界的值小於 *PREV_Obj* 的值時，判別結果為是的話，進入步驟 10，反之則進入步驟 6。
8. *UBLimit(double PREV_Obj)*方法物件是否有接收伺服端傳送更好的目標值，判別結果為是的話，進入步驟 9，反之則進入步驟 7。參數 *PREV_Obj* 為平行計算中所得到的更佳目標值。
9. *PREV_Obj* 的值改變為步驟 8 所接收的更佳目標值，進入步驟 7。
10. *modelSolve(int cCount)*方法物件將計算結果寫入文字檔。
11. *modelSolve(int cCount)*方法物件將文字檔轉換成位元陣列並且回傳給伺服端。
12. 計算節點端結束計算程序，等待伺服端的呼叫。

接著針對演算法之決策流程予以說明，在此流程中主要是針對子集合的選擇，以及當供應商和製造廠增加時，其目標值增加幅度小於某個限度，或者規劃中的供應商和製造廠數量已達規劃人員設定的上限時，決策流程

停止計算並且以目前所得到的最佳解為最佳規劃結果，演算法之決策流程如圖 3.17 所示：

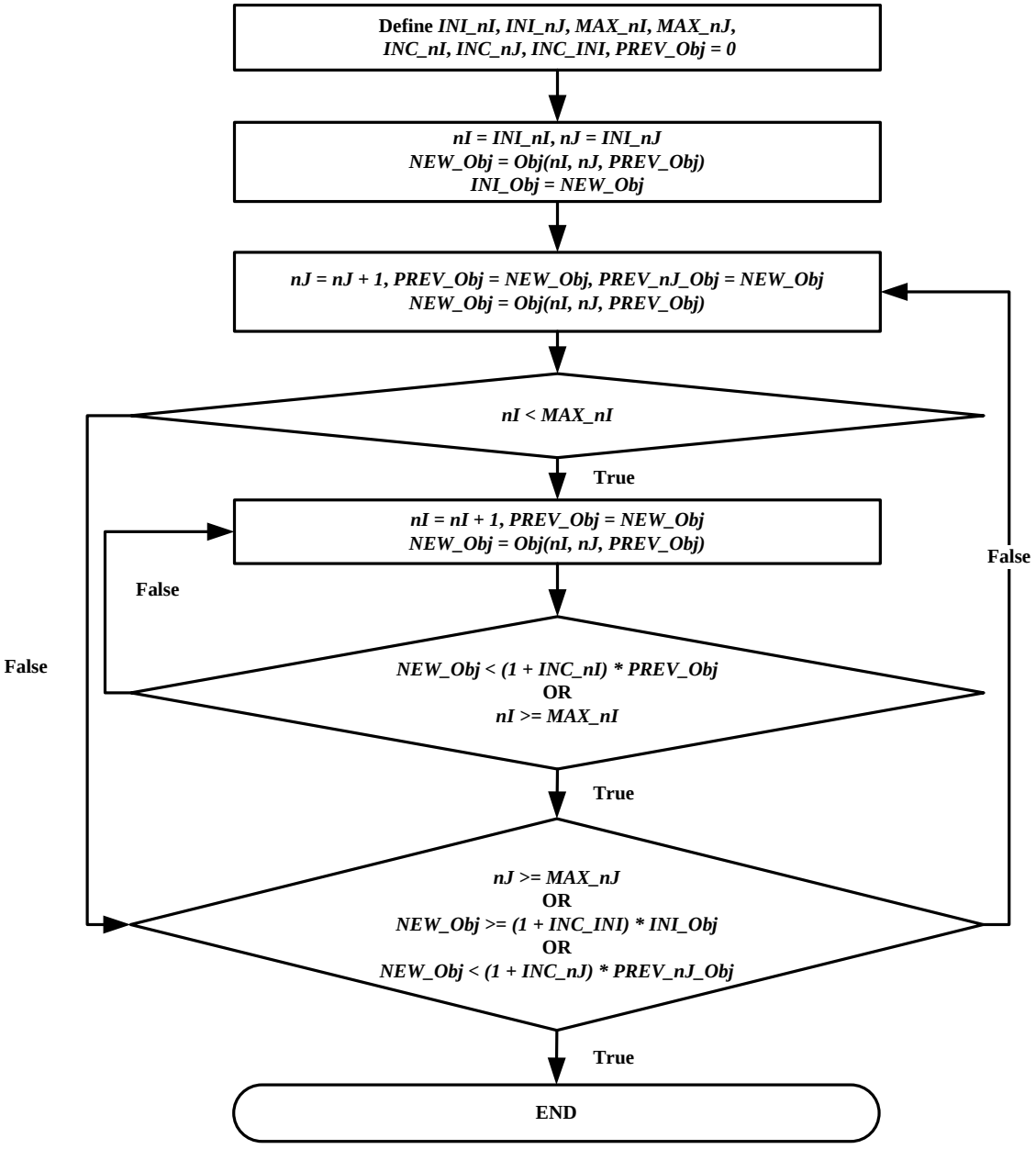


圖 3.17、演算法之決策流程圖

變數說明

INI_nI　　供應商數量的起始值，本研究選擇產能剛好滿足需求的供應商數量

INI_nJ　　製造廠數量的起始值，本研究選擇產能剛好滿足需求的製造廠數量

MAX_nI　　供應商數量的最大值，判別供應商數量是否達到此值

MAX_nJ	製造廠數量的最大值，判別製造廠數量是否達到此值
INC_nI	判別供應商數量增加時，目標值改變百分比的限制
INC_nJ	判別製造廠數量增加時，目標值改變百分比的限制
INC_INI	判別目前所得到的目標值跟最初始得到得到的目標值改變百分比的限制
PREV_Obj	子集合的最佳規劃結果之目標值
NEW_Obj	目前得到的最佳規劃結果之目標值
INI_Obj	最初始的最佳規劃結果之目標值
PREV_nJ_Obj	為目前規劃中的製造廠數量少一家的最佳規劃結果之目標值

接下來將予以說明上圖中的計算流程：

1. 輸入供應商和製造廠數量的起始值(*INI_nI*、*INI_nJ*)，並且設定供應商和製造廠數量的上限(*MAX_nI* 、*MAX_nJ*)、供應商和製造廠增加時，目標標值改變百分比限制的值(*INC_nI*、*INC_nJ*)、跟最初始得到得到的目標值改變百分比限制的值(*INC_INI*)和子集合的最佳規劃結果之目標值(*PREV_Obj*)為 0。
2. 以供應商和製造廠數量的起始值(*nI = INI_nI* 、*nJ = INI_nJ*)和上界的限制為 *PREV_Obj = 0* 進行規劃。*NEW_Obj* 和 *INI_Obj* 的值為最佳規劃結果之目標值。
3. *PREV_Obj* 和 *PREV_nJ_Obj* 的值為目前 *NEW_Obj* 的值。供應商數量(*nI*)不改變、製造廠的數量增加一家(*nJ = nJ + 1*)和上界的限制為 *PREV_Obj* 進行規劃。*NEW_Obj* 的值為最佳規劃結果之目標值。
4. 判別供應商的數量(*nI*)是否小於供應商數量的上限(*MAX_nI*)，如果為是的話進入步驟 5，反之則進入步驟 7。
5. *PREV_Obj* 的值為目前 *NEW_Obj* 的值。供應商數量增加一家(*nI = nI + 1*)、製造廠的數量不改變(*nJ*) 和上界的限制為 *PREV_Obj* 進行規劃。*NEW_Obj* 的值為最佳規劃結果之目標值。
6. *NEW_Obj* 跟 *PREV_Obj* 的差異小於 *PREV_Obj 的 INC_nI* 個百分比，或者供應商數量(*nI*)已經大於或等於供應商數量上限(*MAX_nI*)時，進入步驟 7，反之則進入步驟 5。
7. 製造廠數量(*nJ*)已經大於或等於製造廠數量上限(*MAX_nJ*)時，或者，

NEW_Obj 跟 *INI_Obj* 的差異大於或等於 *INI_Obj 的 INC_INI* 個百分比，或者，*NEW_Obj* 跟 *PREV_nJ_Obj* 的差異小於 *PREV_nJ_Obj* 的 *INC_nJ* 個百分比時，進入步驟 8，反之則進入步驟 3。

8. 決策流程結束。

上述流程為演算法的決策流程，此決策流程以及流程中的參數設定，是依照本研究所建構出來及設定的，針對不同的產業也應該有所不同，而此決策流程的主要目的並非求得全域最佳解，而是以計算時間以及目標值兩項目的中做抉擇；而上述流程中存在著 *Obj(int nI, int nJ, double PREV_Obj)*，此即為本研究所假設的一種函數，代表著在每次執行伺服端的程式時所需輸入的參數(*nI*、*nJ*、*PREV_Obj*)。

本節中說明了分散式平行系統集演算法的建構過程，為了說明此系統之可行性以及處理器數目對於平行計算時間上的影響，在下一章節中將做完整的實驗設計及分析，並以此分散式平行計算解決大規模之數學模型計算時間上的問題。

第四章 實驗設計

本章節利用實驗設計之方法，先針對實驗方式與環境建構做出說明，此說明中包含了實驗環境的說明，以及實驗參數的設定。之後將進行實驗設計及實驗結果進行分析，分析的結果包含了分散式平行系統的計算結果與全域最佳解進行比較，以及處理器數量改變對於計算時間的影響。本章可分為兩節，4.1 為實驗方式與環境建構，4.2 為實驗設計及分析。

4.1 實驗方式與環境建構

4.1.1 實驗環境

1. 訂單資訊：總需求數量為總原物料可供給數量的 30%、60%和 80%。
2. 規劃時間：每個訂單節點的訂單交期時間不同，每期生產規劃時間長度為 1 個週期時間。
3. 多階多廠生產環境：本研究探討之多階多廠生產環境如圖 4.1 所示，包含多個製造廠、多個配銷中心及多個供應商 (將於稍後小節詳細定義供應鏈規模之因子水準)，其中，每個製造廠僅接收原物料並生產為半成品，而每個配銷中心僅接收半成品並組裝為完成品。就原物料而言，記憶體模組產業最重要之原物料為晶圓顆粒，因此，本實驗僅考量此一關鍵原物料之供給狀況。

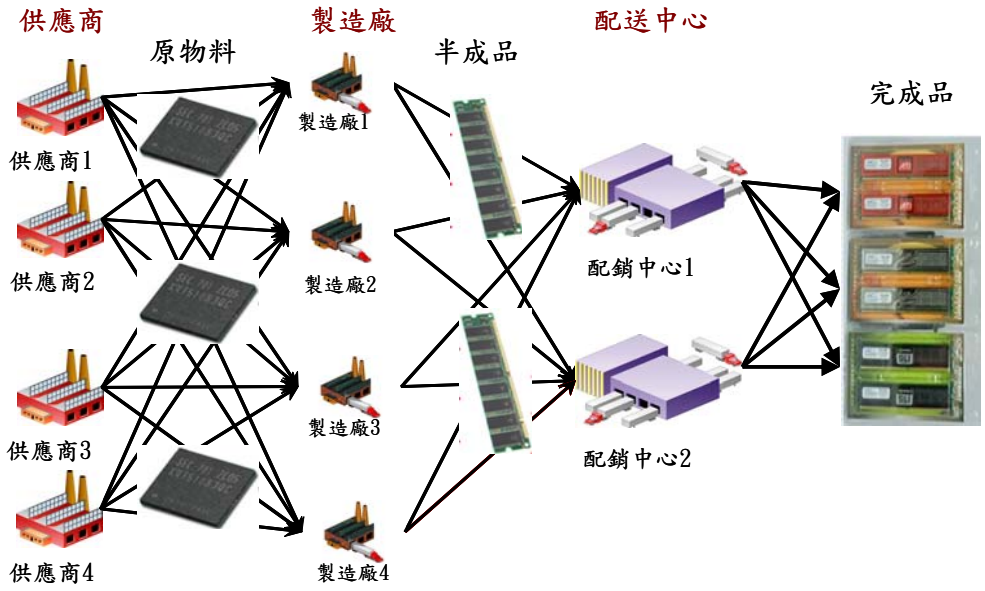


圖 4.1、多階多廠生產環境

4. 進行實驗之作業系統環境：本實驗之作業系統為 Windows XP

Professional SP2，CPU 為 Pentium 4 2.8 G，512MB RAM，求解工具為 JAVA 和 LINGO 10 EXTENDED 版本。

4.1.2 實驗因子及數學模型的參數設定

實驗因子可區分為環境因子及控制因子，其中，環境因子為供應鏈規模大小；而控制因子包含需求水準和處理器數目；就分散式平行系統而言，其參數設定為伺服端主流程一開始及數學模型所需輸入的參數。以下將詳細說明各種因子及參數。

1. 環境因子：供應鏈規模因子區分為規模大及規模小兩個水準。大供應鏈規模中有 8 個供應商、6 個製造廠以及 6 個配銷中心；而小供應鏈規模中則有 6 個供應商、6 個製造廠以及 3 個配銷中心。
2. 控制因子：
   (1) 需求水準因子：需求水準可區分為需求為供給的 30%、60%和 80%。
   (2) 處理器數目因子：小規模的處理器數目可以分為 4 顆、6 顆、8 顆。大規模的處理器數目可以分為 8 顆、12 顆、16 顆。

各個環境因子及其水準整理如表 4.1：

表 4.1、控制因子

控制因子	因子水準	說明
需求水準	高	總需求量為總供給量之 80%
	中	總需求量為總供給量之 60%
	低	總需求量為總供給量之 30%
處理器數目	高	小規模為 8 顆、大規模為 16 顆
	中	小規模為 6 顆、大規模為 12 顆
	低	小規模為 4 顆、大規模為 8 顆

3. 數學模型參數：數學模型參數為 LINGO 中數學模型所需讀取的已知變數，包含了各個節點的基本資訊以及訂單資訊，其參數設定分為大規模和小規模，如下表 4.2 和表 4.3：

表 4.2、數學模型之參數設定(大規模)

參數代號	參數說明	數值設定							
q_i^S	供應商節點 i 的產能	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
		106	106	117	136	140	143	146	143
c_i^{SP}	供應商節點 i 的單位採購成本	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
		61	61	68	60	64	70	61	68
t_i^{SP}	供應商節點 i 的作業前置時間	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
		4	3	9	3	10	7	4	8
t_i^{SI}	供應商節點 i 的庫存週期時間	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
		5	0	4	5	2	0	2	6
q_j^F	製造廠節點 j 的產能	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$		
		281	281	294	285	210	208		
c_j^{FP}	製造廠節點 j 的單位生產成本	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$		
		45	49	47	52	45	52		
c_j^{FI}	製造廠節點 j 的單位庫存成本	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$		
		3	9	3	9	10	1		
c_k^{DP}	訂單節點 k 的單位組裝成本	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$		
		48	40	44	49	50	51		
c_k^{DI}	訂單節點 k 的單位庫存成本	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$		
		4	7	2	7	8	4		
q_k^R	訂單節點 k 的需求數量	*Degree \ k*	*1*	*2*	*3*	*4*	*5*	*6*	
		30%	85	29	25	31	86	56	
		60%	101	82	78	51	203	108	
		80%	56	201	84	316	38	135	
t_k^{SHIP}	訂單節點 k 的訂單交期時間	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$		
		24	23	27	27	23	27		
e_k	訂單節點 k 的單位售價	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$		
		388	371	376	369	384	362		
c_k^{SH}	訂單節點 k 的單位缺貨成本	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$		
		517	494	501	491	511	481		
c_{ij}^{SF}	供應商節點 i 運輸至製造廠節點 j 的單位運輸成本	*i \ j*	*1*	*2*	*3*	*4*	*5*	*6*	
		1	26	22	14	25	12	30	
		2	24	29	13	21	12	20	
		3	15	15	27	10	26	15	
		4	15	14	22	28	17	30	
		5	11	14	29	13	29	25	
		6	28	14	11	17	23	13	
		7	11	17	11	23	28	30	
		8	12	10	12	14	12	20	
c_{jk}^{FD}	製造廠節點 j 運輸至訂單節點 k 的單位運輸成本	*j \ k*	*1*	*2*	*3*	*4*	*5*	*6*	
		1	20	26	21	18	17	20	
		2	13	17	27	20	28	16	
		3	11	29	15	20	17	10	
		4	10	20	17	15	17	19	
		5	11	24	12	16	25	29	
		6	24	27	23	27	18	30	
t_{ij}^{SF}	供應商節點 i 運輸至製造廠節點 j 的作業前置時間 (包含製造及運輸)	*i \ j*	*1*	*2*	*3*	*4*	*5*	*6*	
		1	9	10	8	5	15	10	
		2	7	10	14	5	12	9	
		3	10	6	15	14	12	11	
		4	12	7	12	12	11	6	
		5	6	14	13	8	13	7	
		6	5	11	14	15	14	5	
		7	13	15	7	6	8	6	
		8	10	14	7	14	15	12	
t_{jk}^{FD}	製造廠節點 j 運輸至訂單節點 k 的作業前置時間 (包含組裝及運輸)	*j \ k*	*1*	*2*	*3*	*4*	*5*	*6*	
		1	12	5	10	5	8	10	
		2	13	10	11	11	15	5	
		3	10	9	12	14	14	10	
		4	15	6	8	10	8	9	
		5	13	10	11	12	9	13	
		6	13	7	14	13	12	13	

表 4.3、數學模型之參數設定(小規模)

	參數說明	數值設定					
q_i^S	供應商節點 i 的產能	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
		106	106	117	136	140	143
c_i^{SP}	供應商節點 i 的單位採購成本	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
		61	61	68	60	64	70
t_i^{SP}	供應商節點 i 的作業前置時間	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
		4	3	9	3	10	7
t_i^{SI}	供應商節點 i 的庫存週期時間	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
		5	0	4	5	2	0
q_j^F	製造廠節點 j 的產能	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
		281	281	294	285	210	208
c_j^{FP}	製造廠節點 j 的單位生產成本	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
		45	49	47	52	45	52
c_j^{FI}	製造廠節點 j 的單位庫存成本	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
		3	9	3	9	10	1
c_k^{DP}	訂單節點 k 的單位組裝成本	$k=1$		$k=2$		$k=3$	
		48		40		44	
c_k^{DI}	訂單節點 k 的單位庫存成本	$k=1$		$k=2$		$k=3$	
		4		7		2	
q_k^R	訂單節點 k 的需求數量	Degree \ k	1	2	3		
		30%	55	74	95		
		60%	244	103	101		
		80%	189	129	280		
t_k^{SHIP}	訂單節點 k 的訂單交期時間	$k=1$		$k=2$		$k=3$	
		32		34		30	
e_k	訂單節點 k 的單位售價	$k=1$		$k=2$		$k=3$	
		388		371		376	
c_k^{SH}	訂單節點 k 的單位缺貨成本	$k=1$		$k=2$		$k=3$	
		517		494		501	
c_{ij}^{SF}	供應商節點 i 運輸至製造廠節點 j 的單位運輸成本	i \ j	1	2	3	4	5	6
		1	26	22	14	25	12	30
		2	24	29	13	21	12	20
		3	15	15	27	10	26	15
		4	15	14	22	28	17	30
		5	11	14	29	13	29	25
		6	28	14	11	17	23	13
c_{jk}^{FD}	製造廠節點 j 運輸至訂單節點 k 的單位運輸成本	j \ k	1	2	3			
		1	20	26	21			
		2	13	17	27			
		3	11	29	15			
		4	10	20	17			
		5	11	24	12			
		6	24	27	23			
t_{ij}^{SF}	供應商節點 i 運輸至製造廠節點 j 的作業前置時間 (包含製造及運輸)	i \ j	1	2	3	4	5	6
		1	9	10	8	5	15	10
		2	7	10	14	5	12	9
		3	10	6	15	14	12	11
		4	12	7	12	12	11	6
		5	6	14	13	8	13	7
		6	5	11	14	15	14	5
t_{jk}^{FD}	製造廠節點 j 運輸至訂單節點 k 的作業前置時間 (包含組裝及運輸)	j \ k	1	2	3			
		1	12	5	10			
		2	13	10	11			
		3	10	9	12			
		4	15	6	8			
		5	13	10	11			
		6	13	7	14			

5. 演算法之參數：此參數設定為演算法之決策流程所需設定的參數，本研究所採用的參數設定如表 4.4：

表 4.4、演算法之決策流程的參數設定

參數代號	參數說明	數值設定	
INI_nI	供應商數量的起始值	小規模 30%	2
		小規模 60%	4
		小規模 80%	5
		大規模 30%	3
		大規模 60%	5
		大規模 80%	7
INI_nJ	製造廠數量的起始值	小規模 30%	1
		小規模 60%	2
		小規模 80%	3
		大規模 30%	2
		大規模 60%	3
		大規模 80%	4
MAX_nI	供應商數量的最大值	小規模 30%	4
		小規模 60%	5
		小規模 80%	6
		大規模 30%	5
		大規模 60%	6
		大規模 80%	7
MAX_nJ	製造廠數量的最大值	小規模 30%	3
		小規模 60%	4
		小規模 80%	5
		大規模 30%	4
		大規模 60%	5
		大規模 80%	5
INC_nI	供應商數量增加時，目標值改變百分比的限制	0.03	
INC_nJ	製造廠數量增加時，目標值改變百分比的限制	0.03	
INC_INI	與最初始得到得到的目標值改變百分比的限制	0.10	

4.2 實驗設計及分析

4.2.1 小規模數學模型的全域最佳解

本研究先針對小規模的數學模型求解，依照 LINGO 求出來的解和時間以利之後分散式平行系統的規劃結果做比較，其全域最佳解的結果在此節中會以視覺化之圖形的方式表達出決策變數的值，並說明其規畫出來的總淨利、各項成本以及計算時間。

1. 小規模 30%需求水準

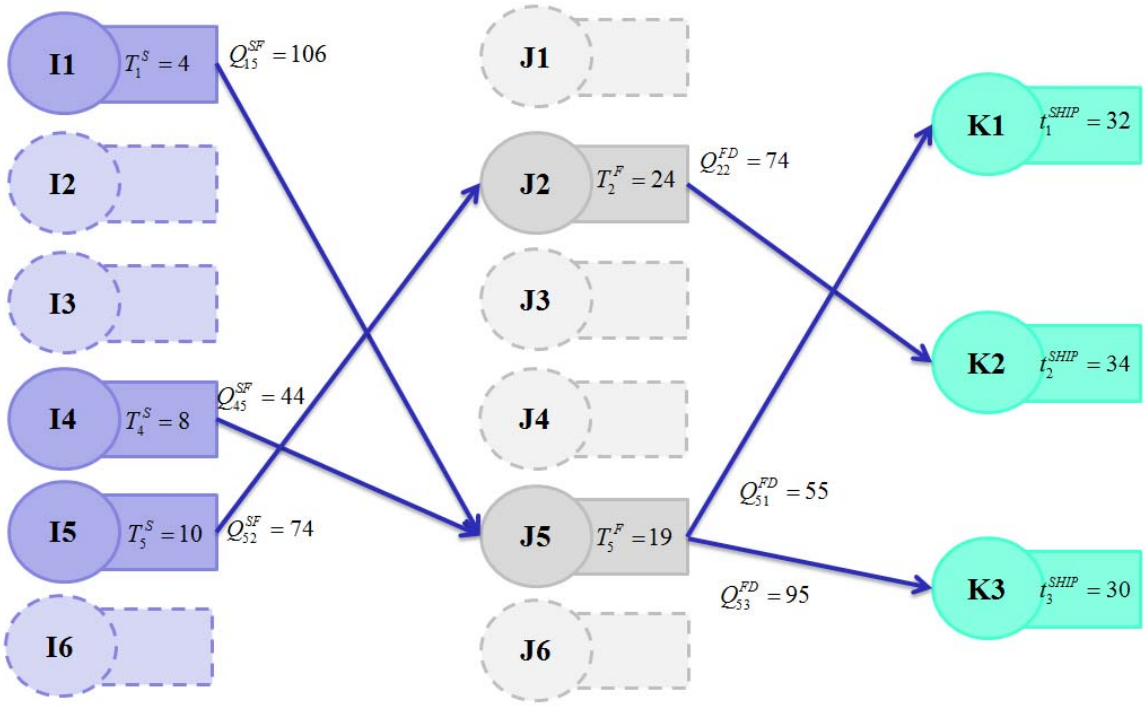


圖 4.2、小規模 30%需求水準的全域最佳規劃結果

規劃結果的總淨利、各項成本及計算時間如下：

(1) 總淨利：44457
(2) 總盈收：84514
(3) 採購成本：13842
(4) 製造成本：20156
(5) 製造廠之存貨持有成本：0
(6) 配銷中心之存貨持有成本：0
(7) 運輸成本：6059
(8) 缺貨處罰成本：0
(9) 計算時間：1649(s)

2. 小規模 60%需求水準

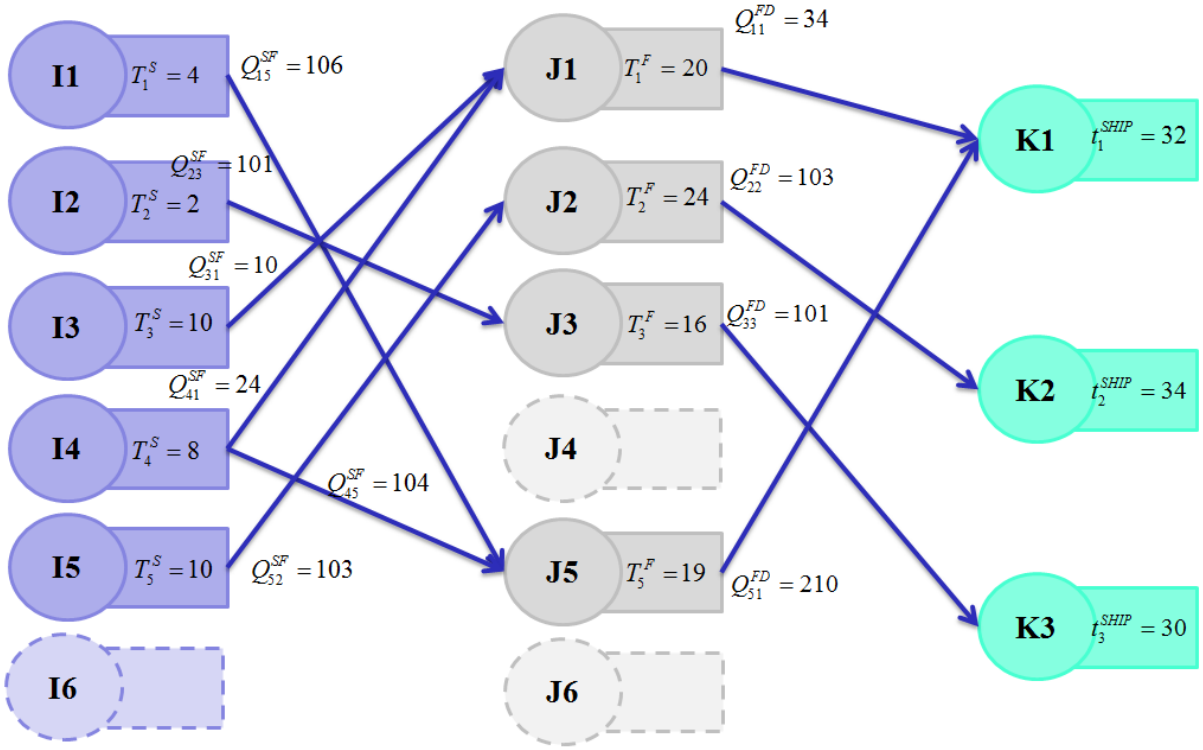


圖 4.3、小規模 60%需求水準的全域最佳規劃結果

規劃結果的總淨利、各項成本及計算時間如下：

(1) 總淨利：89267

(2) 總盈收：170861

(3) 採購成本：27579

(4) 製造成本：41050

(5) 製造廠之存貨持有成本：0

(6) 配銷中心之存貨持有成本：404

(7) 運輸成本：12561

(8) 缺貨處罰成本：0

(9) 計算時間：33429(s)

3. 小規模 80%需求水準

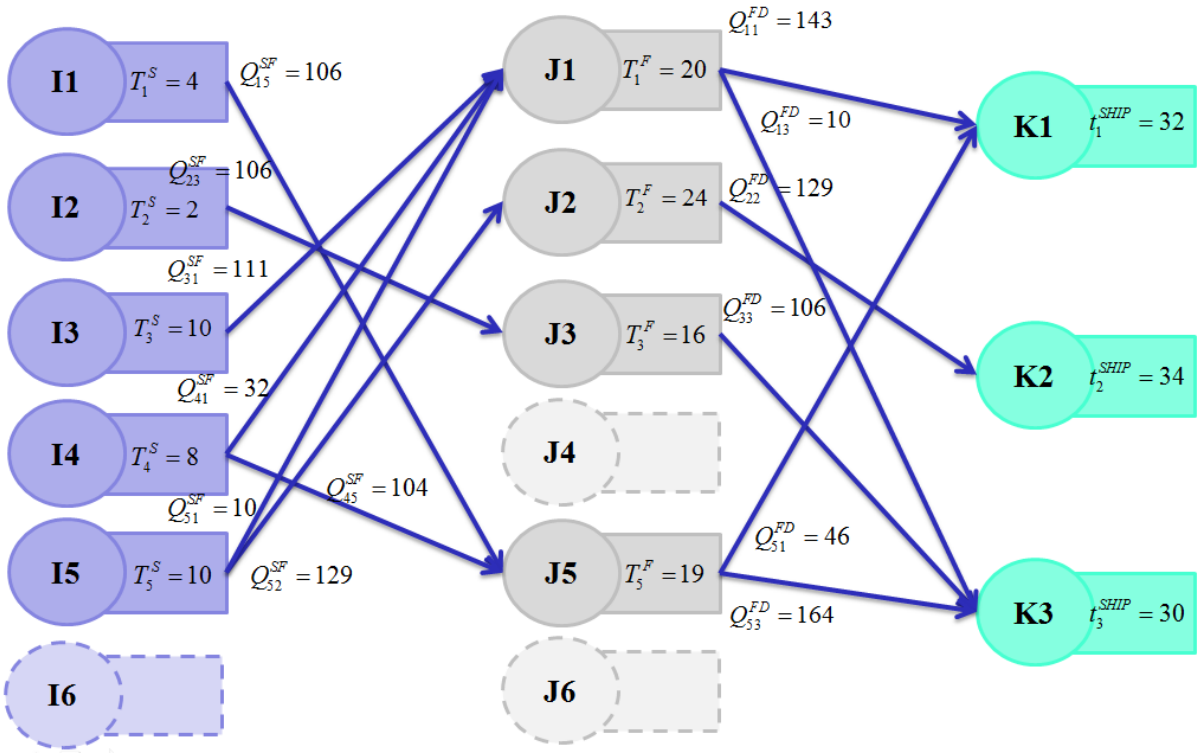


圖 4.4、小規模 80%需求水準的全域最佳規劃結果

規劃結果的總淨利、各項成本及計算時間如下：

(1) 總淨利：116395

(2) 總盈收：226471

(3) 採購成本：37536

(4) 製造成本：54190

(5) 製造廠之存貨持有成本：120

(6) 配銷中心之存貨持有成本：424

(7) 運輸成本：17806

(8) 缺貨處罰成本：0

(9) 計算時間：47090(s)

因為在演算法的決策流程中涉及了供應商和製造廠的改變，因此本小節中值得注意的部分分別為全域最佳解的供應商數量、製造廠數量、總淨利和計算時間，如此在執行分散式平行系統所產生的規劃結果中，有個明確的比較依據。

4.2.2 小規模之平行計算的最佳規劃結果

本節中將先提出各個環境因子組合，再以分散式平行系統依照這些組合進行計算，求出最佳規劃結果以及總計算時間，以利之後與全域最佳解的比較和大規模數學模型的求解。首先將先說明所有的環境因子組合，其組合如表 4.5 所示：

表 4.5、小規模之環境因子組合表

環境因子組合	供應鏈規模	需求水準	處理器數量
環境組合 A	小規模	30%	4
環境組合 B	小規模	30%	6
環境組合 C	小規模	30%	8
環境組合 D	小規模	60%	4
環境組合 E	小規模	60%	6
環境組合 F	小規模	60%	8
環境組合 G	小規模	80%	4
環境組合 H	小規模	80%	6
環境組合 I	小規模	80%	8

分散式平行系統依照上述的環境因子組合進行計算，並依此求得總淨利和規劃時間，其計算結果將在下面的表格中完整列出。

1. 環境組合 A 到 C 的求解過程和規劃結果如下表：

表 4.6、環境組合 A 到 C 之規劃結果

	環境組合 A	環境組合 B	環境組合 C
供應商數量	4		
製造廠數量	2		
總淨利	44457	44457	44457
計算時間(s)	473	287	211

由上表可知，處理器數目的改變對於目標值和計算結果並沒有任何的影響；在計算時間方面，處理器數目的增加下，明顯的降低平行計算所需要的計算時間。

2. 環境組合 D 到 F 的求解過程和規劃結果如下表：

表 4.7、環境組合 D 到 F 之規劃結果

	環境組合 D	**環境組合 E**	**環境組合 F**
供應商數量	5		
製造廠數量	4		
總淨利	89267	89267	89267
計算時間(s)	6338	4184	3477

由上表可知，處理器數目的改變對於目標值和計算結果並沒有任何的影響；在計算時間方面，處理器數目的增加下，明顯的降低平行計算所需要的計算時間。

3. 環境組合 G 到 I 的求解過程和規劃結果如下表：

表 4.8、環境組合 G 到 I 之規劃結果

	環境組合 G	**環境組合 H**	**環境組合 I**
供應商數量	6		
製造廠數量	4		
總淨利	116395	116395	116395
計算時間(s)	5917	5236	4435

由上表可知，處理器數目的改變對於目標值和計算結果並沒有任何的影響；在計算時間方面，處理器數目的增加下，只能小幅度的降低平行計算所需要的計算時間。

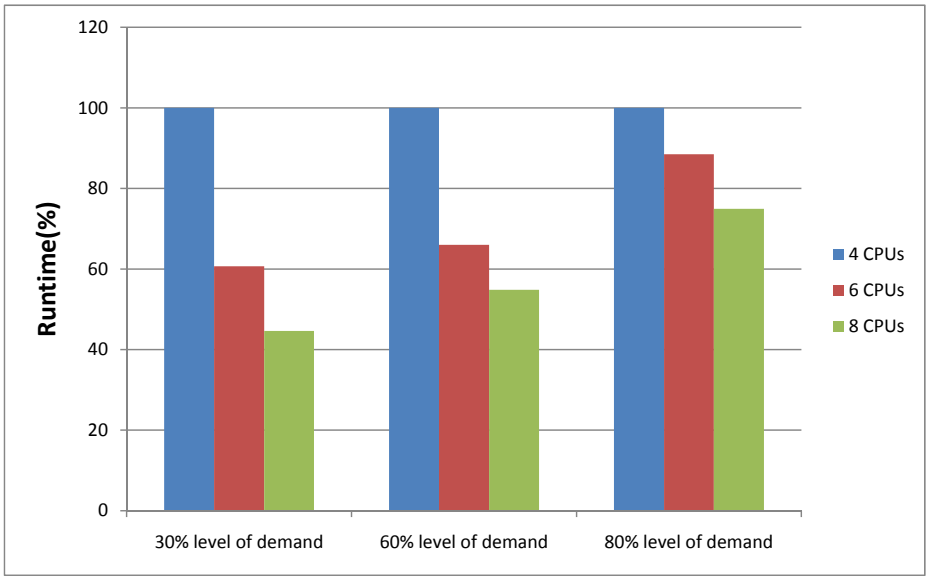


圖 4.5、小規模平行計算之處理器數目和計算時間的比較

總結上述實驗結果的計算時間如圖 4.5，圖 4.5 中以 4 顆處理器數目的計算時間為 100%的情況下，比較處理器數目增加下的計算時間所佔得百分比，此圖說明小規模 30%與 60%在處理器數目 6 顆到 8 顆中，對於計算時間的改變有趨緩的趨勢，因此根據此實驗能夠預期，處理器數目增加到一定的數目時，對於計算時間的改變並不會太大，甚至可能會到一定的極限值而無法再下降；而在 4 顆處理器與 8 顆處理器的差異中，以 80%需求水準的差異幅度為最小，由此可知，針對小規模 80%需求水準是否需要更多的處理器來節省下計算時間，必須要看各產業對於規劃時間的需求以及處理器的成本來考量。

在此小節中針對小規模進行平行計算，並且以演算法流程中所設定的參數為終止條件；在上述過程中說明了平行計算中，處理器的數量對於目標值並無任何影響，而在計算時間方面，會因為處理器數目的不同而造成計算時間上的差異，在下節中將以本節的結果與全域最佳解來作目標值與計算時間上的比較。

4.2.3 小規模之平行計算與全域最佳解比較

上一小節依據演算法的參數設定，求得了環境組合 A 到 I 的平行計算的結果以及時間，在此一小節中將以小規模之環境(環境組合 A 到 I)，比較全域最佳解規劃方式與本研究所提出之分散式平行系統之規劃結果和計算時間，並在此以圖形比較處理器數目的改變對計算時間的影響與全域最佳解運上時間的差距。

1. 小規模 30%需求水準之總淨利和計算時間的比較：

表 4.9、小規模 30%需求水準之總淨利和計算時間的比較

	全域最佳解	環境組合 A	環境組合 B	環境組合 C
處理器數目	1	4	6	8
總淨利	44457	44457	44457	44457
計算時間	1649	473	287	211
差異百分比(淨利)	-	0%	0%	0%
差異百分比(時間)	-	71.32%	82.60%	87.20%

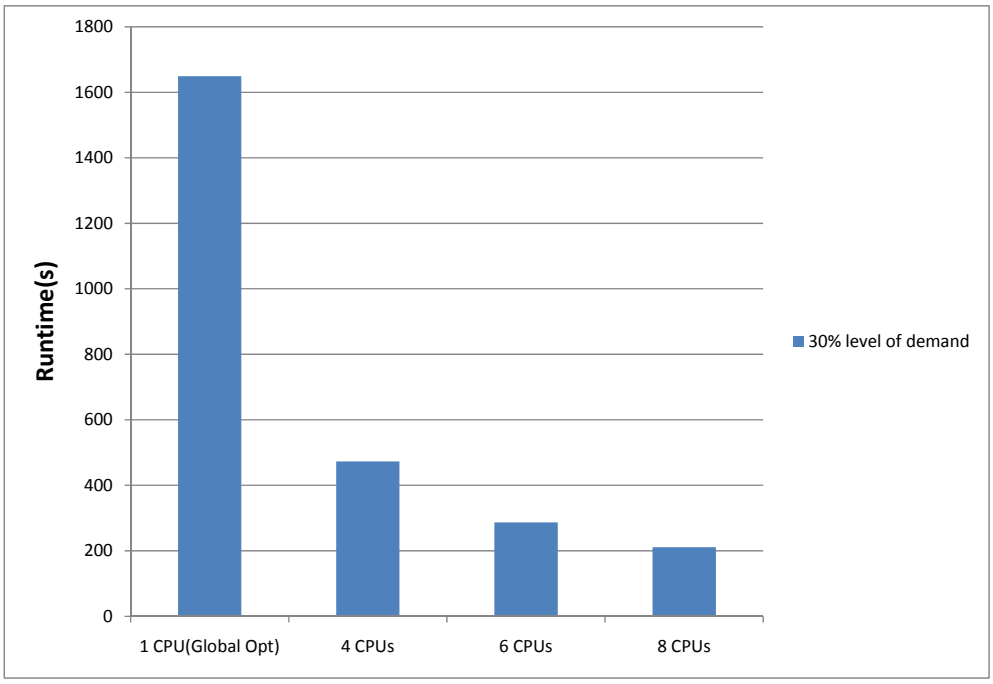


圖 4.6、小規模 30%需求水準之處理器數目和計算時間的比較

環境組合 A 到 C 與全域最佳的比較可知，依照演算法的決策流程所進行的平行計算，其計算結果在供應商與製造廠數量達到全域最佳解所規畫

出來的結果時，平行計算所計算出來的目標值與規劃結果皆與全域最佳解相同；而平行計算與全域最佳解的計算時間上，平行計算明顯的大幅降低計算時間，在採用 4 顆處理器時省下了 71%的計算時間，在處理器數目增加到 8 顆時，甚至與全域最佳解的求解時間差異到 87%，由此可知平行計算在求解上的可行性與時間上的差異性。

2. 小規模 60%需求水準之總淨利和計算時間的比較：

表 4.10、小規模 60%需求水準之總淨利和計算時間的比較

	全域最佳解	環境組合 D	環境組合 E	環境組合 F
處理器數目	1	4	6	8
總淨利	89267	89267	89267	89267
計算時間	33429	6338	4184	3477
差異百分比(淨利)	-	0%	0%	0%
差異百分比(時間)	-	81.04%	87.48%	89.60%

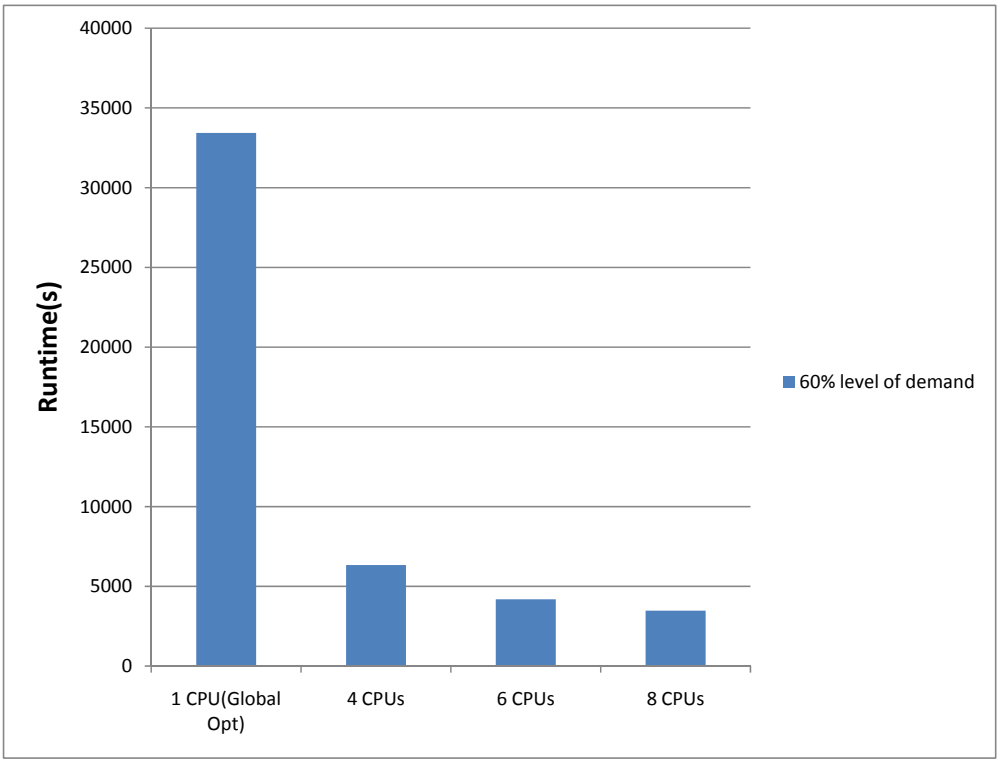


圖 4.7、小規模 60%需求水準之處理器數目和計算時間的比較

環境組合 D 到 F 與全域最佳的比較可知，依照演算法的決策流程所進

行的平行計算，其計算結果在供應商與製造廠數目達到全域最佳解所規畫出來的結果時，平行計算所計算出來的目標值與規劃結果皆與全域最佳解相同；而平行計算與全域最佳解的計算時間上，平行計算明顯的大幅降低計算時間，在採用 4 顆處理器時省下了 81%的計算時間，在處理器數目增加到 8 顆時，甚至與全域最佳解的求解時間差異到 89%，由此可知平行計算在求解上的可行性與時間上的差異性。

3. 小規模 80%需求水準之總淨利和計算時間的比較：

表 4.11、小規模 80%需求水準之總淨利和計算時間的比較

	全域最佳解	環境組合 G	環境組合 H	環境組合 I
處理器數目	1	4	6	8
總淨利	116395	116395	116395	116395
計算時間	47090	5917	5236	4435
差異百分比(淨利)	-	0%	0%	0%
差異百分比(時間)	-	87.43%	88.88%	90.58%

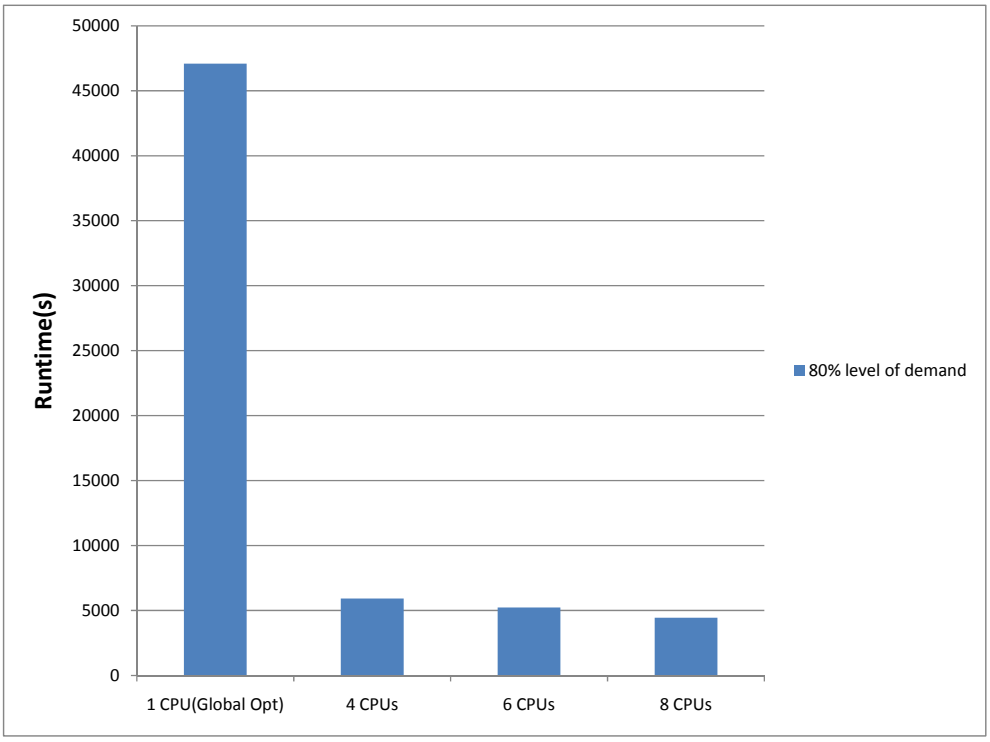


圖 4.8、小規模 80%需求水準之處理器數目和計算時間的比較

環境組合 G 到 I 與全域最佳的比較可知，依照演算法的決策流程所進行的平行計算，其計算結果在供應商與製造廠數目達到全域最佳解所規畫出來的結果時，平行計算所計算出來的目標值與規劃結果皆與全域最佳解相同；而平行計算與全域最佳解的計算時間上，平行計算明顯的大幅降低計算時間，在採用 4 顆處理器時省下了 87%的計算時間，在處理器數目增加到 8 顆時，甚至與全域最佳解的求解時間差異到 90%，由此可知平行計算在求解上的可行性與時間上的差異性。

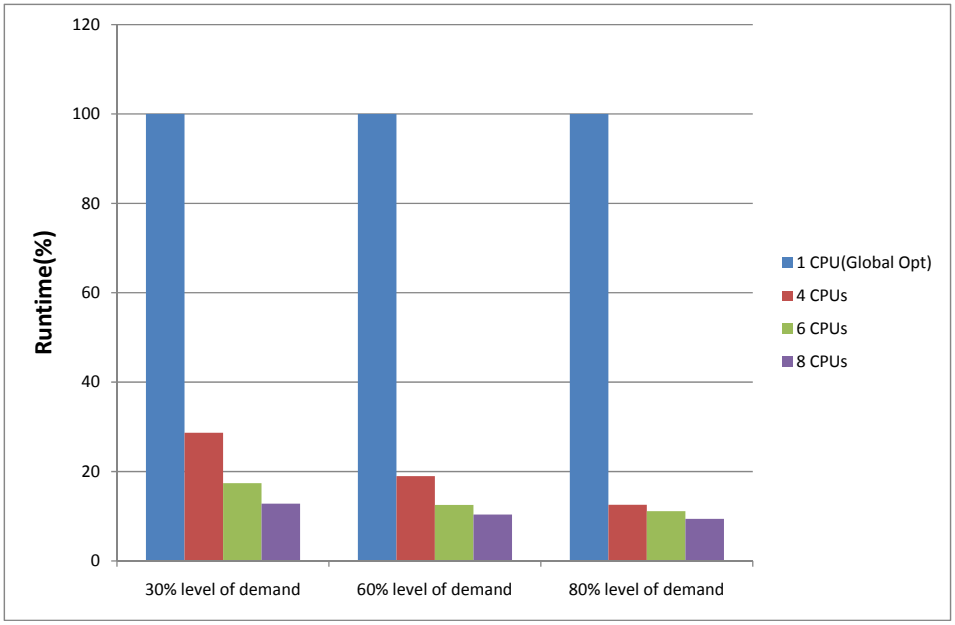


圖 4.9、小規模之處理器數目和計算時間的比較

圖 4.9 中整理了小規模各需求水準計算時間上的比較，以全域之數學模型的計算時間為 100%的情況下，由全域最佳解與平行計算時間上的比較，可以看出 60%與 80%採用平行計算的方式下，所降低的計算時間幅度最大，由此可知，平行計算對於 60%與 80%需求準準計計算時間最為有效率；而在 30%需求水準中，雖然全域最佳解的求解時間並不是很長，但是由上圖中可看出對於計算時間的下降也是非常有效率的。因此本小節中說明了分散式平行系統之最佳規劃結果的可行性，在大規模中對於不了解全域最佳解的狀態下，是可以採用分散式平行系統在短時間內推導出接近全域最佳解的可行解。

4.2.4 大規模之平行計算的最佳規劃結果

對大規模的數學模型求全域最佳解時，規畫時間會隨著規模的增加而造成計算時間呈現 *NP-Complete* 的狀態，在實務上實在是不方便。因此在本小節中將採用分散式平行系統在短時間內求出最佳規劃結果，並比較處理器數目的改變對於計算時間的影響，而全域最佳解的部分，本研究將採用 LINGO 在計算 11 個小時後，所求得的上界和下界與平行計算的結果做比較。大規模的環境組合如表 4.12 所示：

表 4.12、大規模之環境因子組合表

環境因子組合	供應鏈規模	需求水準	處理器數量
環境組合 J	大規模	30%	8
環境組合 K	大規模	30%	12
環境組合 L	大規模	30%	16
環境組合 M	大規模	60%	8
環境組合 N	大規模	60%	12
環境組合 O	大規模	60%	16
環境組合 P	大規模	80%	8
環境組合 Q	大規模	80%	12
環境組合 R	大規模	80%	16

1. 環境組合 J 到 L 的求解過程和規劃結果如表 4.13，以及處理器的數量對於計算時間的改變如圖 4.10：

表 4.13、環境組合 J 到 L 之規劃結果

	環境組合 J	環境組合 K	環境組合 L
供應商數量	4		
製造廠數量	3		
總淨利	60168	60168	60168
總計算時間(s)	3397	2301	1797

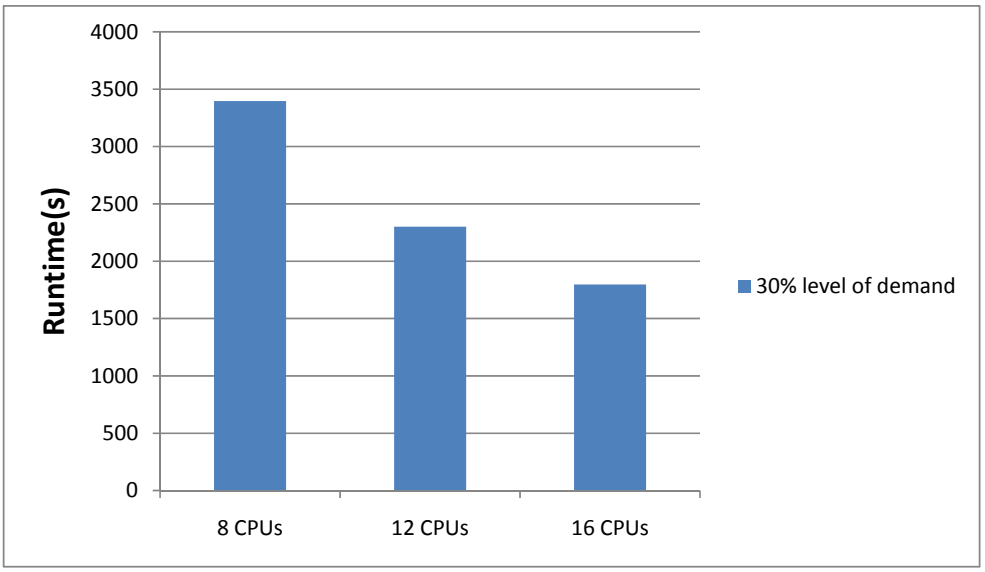


圖 4.10、大規模 30%需求水準的處理器數目和計算時間的比較

由上表可知，處理器數目的改變對於目標值和計算結果並沒有任何的影響；在計算時間方面，處理器數目的增加下，明顯的降低平行計算所需要的計算時間。

2. 環境組合 M 到 O 的求解過程和規劃結果如表 4.14，以及處理器的數量對於計算時間的改變如圖 4.11：

表 4.14、環境組合 M 到 O 之規劃結果

	環境組合 M	環境組合 N	環境組合 O
供應商數量	6		
製造廠數量	4		
總淨利	118152	118152	118152
總計算時間(s)	7565	6318	5599

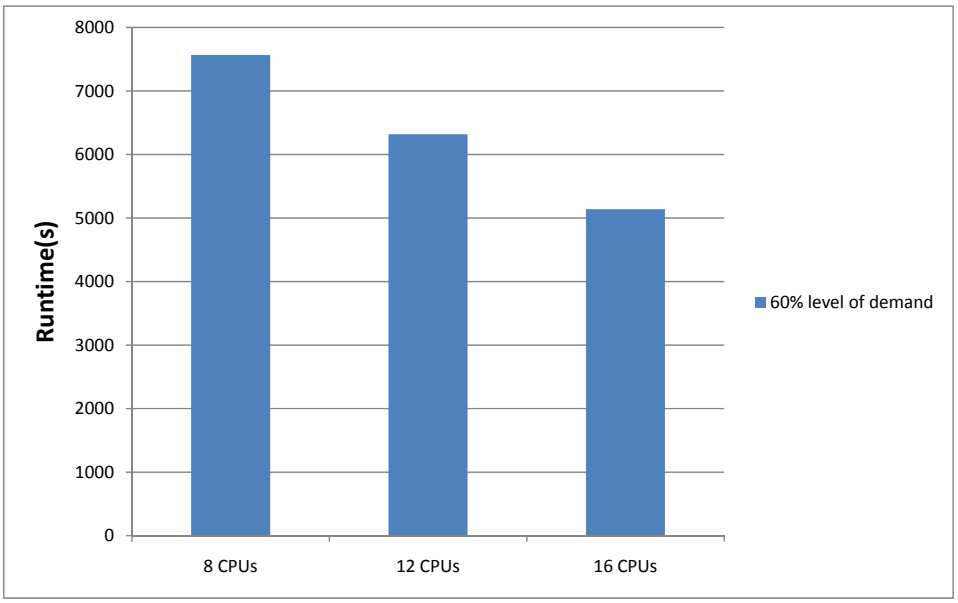


圖 4.11、大規模 60%需求水準的處理器數目和計算時間的比較

由上表可知，處理器數目的改變對於目標值和計算結果並沒有任何的影響；在計算時間方面，處理器數目的增加下，計算時間呈現穩定性下降。

3. 環境組合 P 到 R 的求解過程和規劃結果如表 4.15，以及處理器的數量對於計算時間的改變如圖 4.12：

表 4.15、環境組合 P 到 R 之規劃結果

	環境組合 P	環境組合 Q	環境組合 R
供應商數量	7		
製造廠數量	5		
總淨利	152124	152124	152124
總計算時間(s)	60431	48462	43458

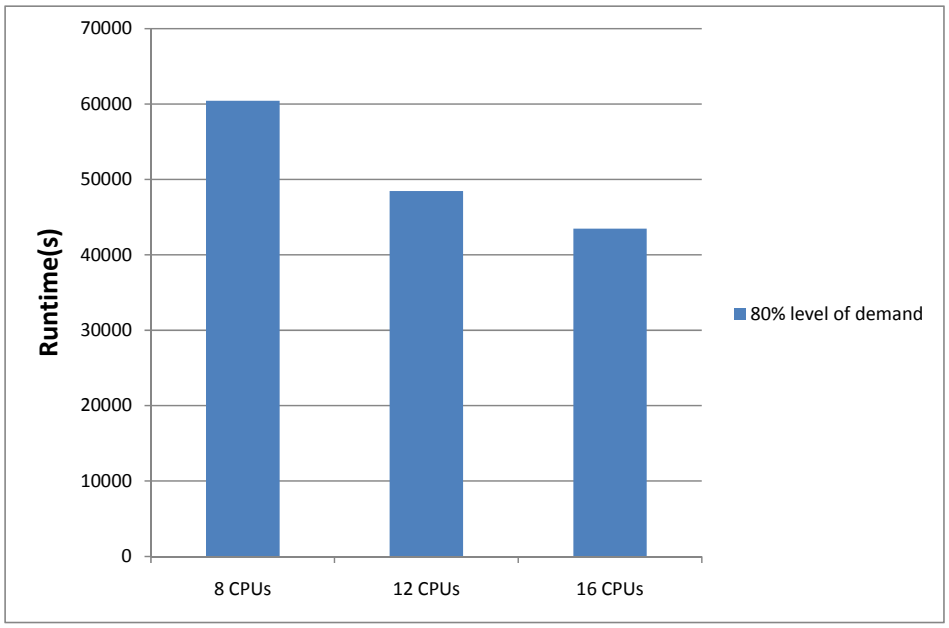


圖 4.12、大規模 80%需求水準的處理器數目和計算時間的比較

由上表可知，處理器數目的改變對於目標值和計算結果並沒有任何的影響；在計算時間方面，處理器數目的增加下，明顯的降低平行計算所需要的計算時間；而在此需求水準之下，計算時間普遍大於 12 小時，因此此實驗說明了平行計算會因為數學模型的複雜度增加，而大幅的增加平行計算的計算時間，在此時應該針對演算法之決策流程中做參數上的設定，以更寬鬆的限制來進行平行計算。

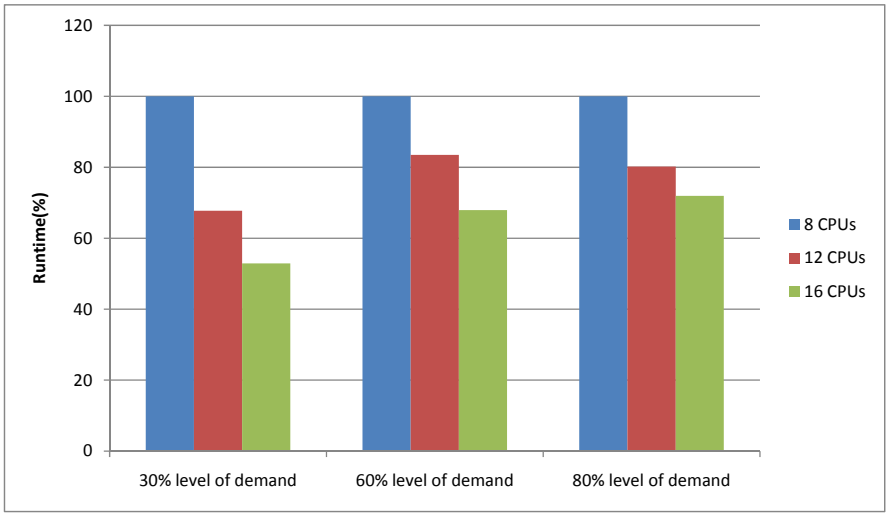


圖 4.13、大規模平行計算之處理器數目和計算時間的比較

總結本小節的計算時間如圖 4.13 可知，以 8 顆處理器的計算時間為 100%的情況下，由上圖可知 30%和 80%需求水準對於處理器數目的改變，從 8 顆到 12 顆時明顯的下降平行計算的時間，但是在 12 顆到 16 顆時，計算時間有趨緩的趨勢，因此能夠預其當處理器數目到一定的數量時會無法下降計算時間；而 60%需求水準對於處理器數目的改變，計算時間呈現穩定性的下降，因此本研究預期此需求水準的平行計算是可以繼續增加處理器數目而明顯降低計算時間。

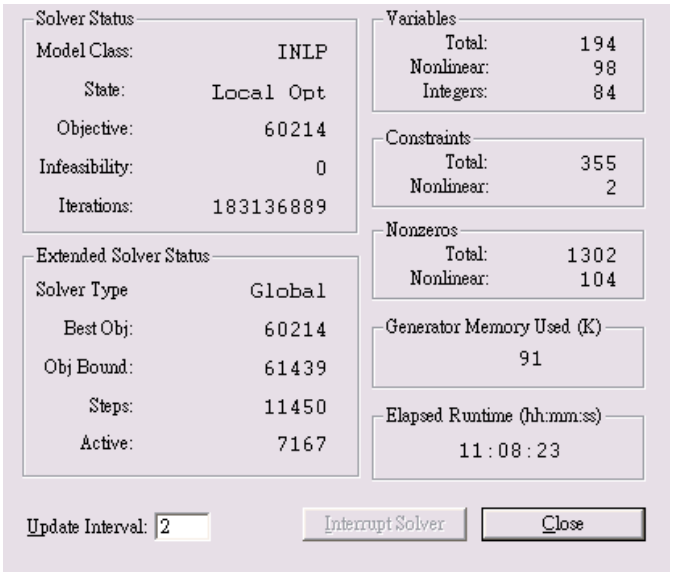


圖 4.14、大規模 30%需求水準之全域規劃結果

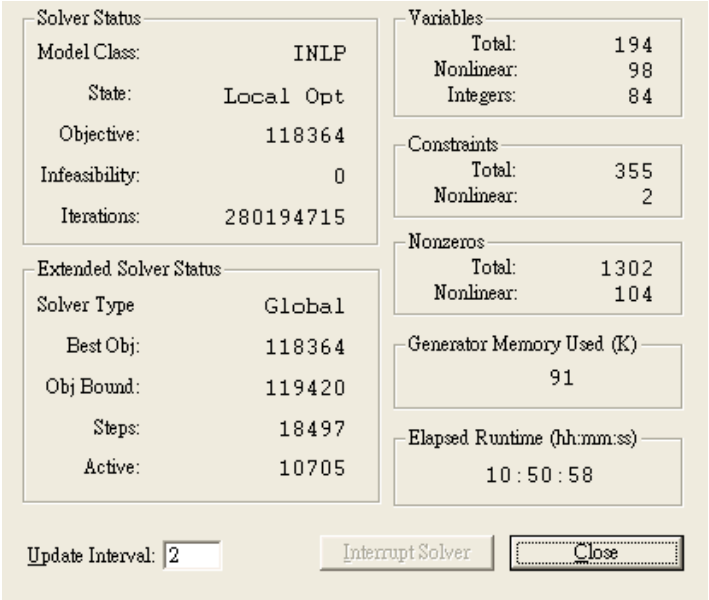


圖 4.15、大規模 60%需求水準之全域規劃結果

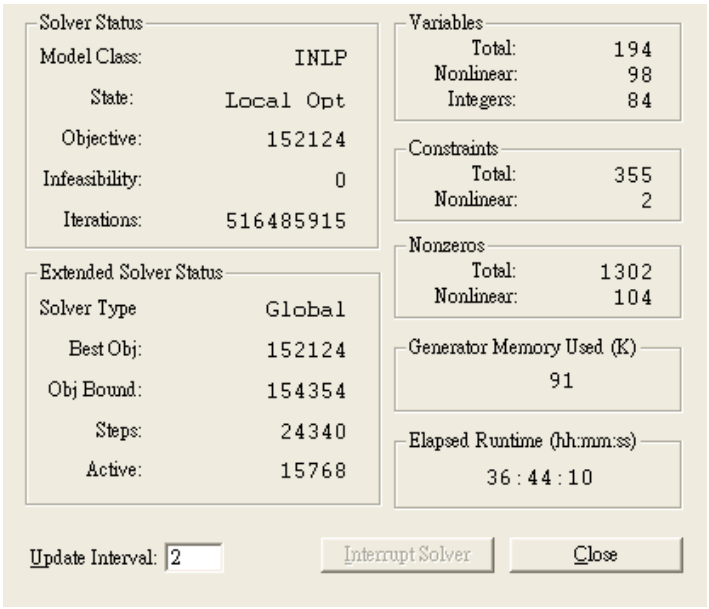


圖 4.16、大規模 80%需求水準之全域規劃結果

上圖 4.14 到 4.16 為 LINGO 針對全域的數學模型求解，在計算 11 小時後所求得的結果，由圖中所看到的 Best Obj 即為下界，為 LINGO 針對此數學模型目前所求得的解，而 Obj Bound 即為本研究之前所提到的上界，由上面三張圖可以看出，在此三種需求水準計算 11 個小時後仍然無法收斂到一個確切的目標值，而本研究將採用此三種模式的下界與平行計算的最佳規劃結果作比較，其比較結果將在接下來說明。

表 4.16、大規模 30%需求水準之總淨利和計算時間的比較

	全域規劃	環境組合 J	環境組合 K	環境組合 L
處理器數目	1	8	12	16
總淨利	60214*	60168	60168	60168
計算時間	40103	3397	2301	1797
計算完畢	否	是	是	是
差異百分比(淨利)	-	0.07%	0.07%	0.07%
差異百分比(時間)	-	91.53%	94.26%	95.52%

*為 LINGO 在此計算時間下所求得的下界，並非全域最佳解。

表 4.17、大規模 60%需求水準之總淨利和計算時間的比較

	全域規劃	環境組合 M	環境組合 N	環境組合 O
處理器數目	1	8	12	16
總淨利	118364*	118152	118152	118152
計算時間	39058	7565	6318	5599
計算完畢	否	是	是	是
差異百分比(淨利)	-	0.18%	0.18%	0.18%
差異百分比(時間)	-	80.63%	83.82%	85.66%

*為 LINGO 在此計算時間下所求得的下界，並非全域最佳解。

表 4.18、大規模 80%需求水準之總淨利和計算時間的比較

	全域規劃	環境組合 P	環境組合 Q	環境組合 R
處理器數目	1	8	12	16
總淨利	152124*	152124	152124	152124
計算時間	132250	60431	48462	43458
計算完畢	否	是	是	是
差異百分比(淨利)	-	0%	0%	0%
差異百分比(時間)	-	54.31%	63.36%	67.14%

*為 LINGO 在此計算時間下所求得的下界，並非全域最佳解。

由上面的三張表的比較可知，在 30%和 60%需求水準下，採用平行計算明顯的下降了計算時間，而所求得的目標值與全域規劃計算 11 小時的解非常接近，由此可知平行計算求解的可行性與計算時間的差異性；在 80%需求準下，全域規劃在 36 小時所求得的目標值等於平行計算所求得的最佳規劃結果之目標值，由此可之平行計算式可以在短時間內，能求出與全域規劃之結果差異不大的解；而在採用此種資料分割模式之下，上界與下界的收斂到一個相同的值之速度，與全域規劃比較起來是快很多的。

本章節針對平行計算與全域規劃的比較，說明了處理器數目對於平行計算之計算時間上的影響，以及平行計算在計算時間上所節省的時間；並以此推導出大規模的全域規劃結果，然而在大規模的規劃結果雖然不能完全與全域規劃結果相等，但與全域規劃在一定時間內所求得的解比較可知，平行計算在短時間內所求得的解也是非常接近的，如此說明了平行計算針對複雜的供應鏈網絡之數學模型，在實務上也是非常實用的。

第五章 結論與未來發展方向

5.1 結論

本研究針對記憶體模組產業特性及現況方法，以視覺化圖形描述供應網絡的特性及限制，建立 INLP 的數學模型並進行求解，然而針對數學模型求解的計算時間會隨著規模和複雜度的增加呈現 *NP-Complete* 的現象，在記憶體模組產業的環境改變下，此規劃結果並無法使用，因此針對記憶體模處產業的供應網絡進行生產規劃時，並須要以其它的演算法或者式計算方式來短時間求出規劃結果，來提供規劃人員進行生產規劃上的參考。

針對過去對於供應網絡之數學模型 *NP-Complete* 的研究進行探討，過去的研究針對此現象以各種的方式進行求解，然而對於經常應用於其他領域的平行計算卻非常的稀少，因此本研究以目前既有的技術 JavaRMI 和 LINGO 的結合，建構出分散式平行計算的系統以解決 *NP-Complete* 的問題。而本研究對於資料的切割方式是採用將數學模型拆解成規模較小的數學模型進行分散式平行計算，並針對此平行計算提出合理且提高計算效率的演算法，在系統建構完後進行實驗設計及分析，驗證此分散式平行計算的可行性及合理性。

在本研究模式與全域的數學模型求解的比較上，透過實驗結果可知，本研究模式對於計算時間以及規劃結果，都具有可行性與合理性，因此本研究模式證實了平行計算應用於供應網絡的生產規劃是相當實用的，以此模式提供記憶體模組產的規劃人員做為生產規劃上的參考。

5.2 未來發展方向

本研究對於供應網絡的數學模型是以一些假設為前提下所建構的，因此在未來可望能夠增加數學模型的複雜度，讓此數學模型更能夠表現出記憶體模組產業的特性；然而在分散式平行計算對於資料的切割方式，對於不同複雜度的模型也應該採用不同的方式進行資料的切割，因此在未來可改進的方向為提供更好的資料切割方式。

在本研究的實驗設計中也提到平行計算會因為數學模型的複雜度增加而造成本研究模式計算時間上的增加，然而在過去許多研究針對數學模型的求解問題，提供了許多的求解方法來降低運算時間，因此在未來的研究領域中，將以不同的求解方法進行平行計算，比較出計算時間的可行性以及規劃結果的合理性，如此才能應用不同求解方式，來提高平行計算對於計算時間上的效率。

參考文獻

中文部分

[1] 郭尚君，2008。*Java 2 入門進階*，文魁。

[2] 陳權，2009。*平行分子動力學應用於個人電腦叢集系統分析與效能研究*，工程科學系，國立成功大學。

[3] 許良政(譯)，2008。*TCP/IP Java網路程式設計*，新文京。

[4] 顏春煌，2008。*Java網路程式設計*，旗標。

英文部分

[5] Arntzen, B.C., Brown, G.G., Harrison, T.P., Trafton, L.L., 1995. Global Supply Chain Management at Digital Equipment Corporation. *Interfaces*, 25 69-93.

[6] Altiparmak, F., Gen, M., Lin, L., Paksoy, T., 2006. A Genetic Algorithm Approach for Multi-objective Optimization of Supply Chain Networks. *Computers & Industrial Engineering*, 51 197-216.

[7] Aliev, R.A., Fazlollahi, B., Guirimov, B.G., Aliev, R.R., 2007. Fuzzy-genetic approach to aggregate production-distribution planning in supply chain management. *Information Sciences*, 177(20) 4241-4255.

[8] Altiparmak, F., Gen, M., Lin, L., Karaoglan, I., 2009. A steady-state genetic algorithm for multi-product supply chain network design. *Computers & Industrial Engineering*, 56(2) 521-537.

[9] Bell, G., 1992. Ultracomputers: A teraflop before its time. *Communications of the ACM* , 35(8) 27-47.

[10] Bhatnagar, R., Chandra, P., Goyal, S.K., 1993. Models for multi-plant coordination. *European Journal of Operational Research*, 67 141-160.

[11] Chen, S.Y., Chern, C.C., 1999. Shortest Path for A Supply Chain Network. *The 4th International Conference, Asia-Pacific Region of Decision Sciences Institute*.

[12] Chan, F.T.S., Chung, S.H., Wadhwa, S., 2005. A hybrid genetic algorithm for production and distribution. *Omega*, 33(4) 345-355.

[13] Chern, C.C., Hsieh, J.S., 2007. A heuristic Algorithm for Master Planning that Satisfies Multiple Objectives. *Computers & Operations Research*, 34(11) 3419-3513.

[14] DRAMeXchange. http://www.dramexchange.com

[15] Flynn, M.J., 1976. Some Computer Organizations and their effectiveness. *IEEE Transactions on Computers*, 21(9) 948-960.

[16] Flynn, M.J., Rudd, K.W., 1996. Parallel Architectures. *ACM Computing Surveys*, 28(1) 67-70.

[17] Guinet, A., 2001. Multi-site planning: a transshipment problem. *International Journal of Production Economics*, 74(1-3) 21-32.

[18] Giglio, D., Minciardi, R., 2003. Modelling and optimization of multi-site production systems in supply chain networks. *IEEE International Conference*, 3(5-8) 2678-2683.

[19] Gen, M., Syarif, A., 2005. Hybrid genetic algorithm for multi-time period

production/distribution planning. *Computers & Industrial Engineering*, 48(4) 799-809.
[20] Hwang, K., Briggs, F.A., 1984. Computer Architecture and Parallel Processing. *The McGraw Companies, Inc.*
[21] Harland, C., 1997. Supply chain operational performance roles. *Integer Manufacturing* System, 8(2) 70-78.
[22] Kanyalkar, A.P., Adil, G.K., 2010. A robust optimisation model for aggregate and detailed planning of a multi-site procurement-production-distribution system. *International Journal of Production Research*, 48(3) 635-656.
[23] Lendermann, P., Gan, B.P., McGinnis, L.F., 2001. Distributed Simulation with Incorporated APS Procedures for High-Fidelity Supply Chain Optimization. *Winter Sumulation Conference*.
[24] Lim, S.J., Jeong, S.J., Kim, K.S., Park, M.W., 2006. A simulation approach for production-distribution planning with consideration given to replenishment policies. *The International Journal of Advanced Manufacturing Technology*, 27(5) 593-603.
[25] Lin, J.T., Chen, Y.Y., 2007. A multi-site supply network planning problem considering variable time buckets– A TFT-LCD industry case. *The International Journal of Advanced Manufacturing Technology*, 33(9-10) 1031-1044.
[26] LANSHUN, N., XIAOFEI, X., DECHEN Z., 2008. Collaborative Planning in Supply Chains by Lagrangian Relaxation and Genetic Algorithms. *International Journal of Information Technology & Decision Making*, 7(1) 7258-7262.
[27] Massege Passing Interface Forum. http://www.mpi-forum.org/
[28] Moon, C., Kim, J., Hur, S., 2002. Integrated process planning and scheduling with minimizing total tardiness in multi plants supply chain. *Computers & Industrial Engineering*, 43(1-2) 331-349.
[29] Orfali, R., Harkey, D., Edwards, J., 1996. The Essential Distributed Objects Survival Guide. *John Willy & Sons*.
[30] PVM:Parallel Virtual Machine. http://www.csm.ornl.gov/pvm/
[31] Shapiro, J.F., 2001. Modeling the Supply Chain. *Thomson Learning*.
[32] Stock, J.R., Lambert, D.M., 2001. Strategic logistic management. *The McGraw Companies, Inc*.
[33] Samxsivan, M., Schmidt, C.P., 2002. A heuristic procedure for solving multi-plant, multi-item, multi-period capacitated lot-sizing problems. *Asia-Pacific Journal of Operational Research*, 19 87-105.
[34] Shen, W., Kremer, R., Ulieru, M., Norrie, D., 2003. A collaborative agent-based infrastructure for Internet-enabled collaborative enterprises. *International Journal of Production Research*, 41(8) 1621-1638.
[35] Santoso, T., Ahmed, S., Goetschalckx, M., Shapiro, A., 2005. A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, 167(1) 96-115.
[36] The OMG's CORBA Website. http://www.corba.org/
[37] Timpe, C.H., Kallrath, J., 2000. Optimal Planning In Large Multi-site Production Networks. *European Journal of Operational Research*, 126(2) 422-435.

[38] Wilkinson, B., Allen, M., 1999. Parallel Programming: Techniques and Applications using Networked Workstations and Parallel Computers. *Prentice Hall*.

[39] Watson, K., Polito, T., 2003. Comparison of DRP and TOC financial performance within a multi-product, multi-echelon physical distribution environment. *International Journal of Production Research*, 41(4) 741-765.

[40] Wu, D., 2004. Multi-Item, Multi-Facility Supply Chain Planning : Models, Complexities, and Algorithms. *Computational Optimization and Applications*, 28(3) 325-356.

[41] Yeh, W.C., 2005. A hybrid heuristic algorithm for the multistage supply chain network problem. *The International Journal of Advanced Manufacturing Technology*, 26(5-6) 675-685.

[42] Yeh, W.C., 2006. An efficient memetic algorithm for the multi-stage supply chain network problem. *The International Journal of Advanced Manufacturing*, 29(7): 803-813.