國 立 交 通 大 學
工 業 工 程 與 管 理 學 系

碩 士 論 文

以動態規劃法進行 TFT-LCD 排序機派工之研究

A Dynamic Programming Method for the Dispatching of TFT-LCD Sorting Process

研究生：陳德珊

指導教授：巫木誠 博士

中華民國九十五年五月

以動態規劃法進行 TFT-LCD 排序機派工之研究

A Dynamic Programming Method for the Dispatching of TFT-LCD Sorting Process

研究生：陳德珊　　Student：Te-Shan Chen

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

國立交通大學

工業工程與管理學系

碩士論文

A Thesis

Submitted to Department of Industrial Engineering and Management

College of Management

National Chiao Tung University

In Partial Fulfillment of the Requirements

For the Degree of Master

In

Industrial Engineering and Management

May 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年五月

以動態規劃法進行 TFT-LCD 排序機派工之研究

研究生：陳德珊　　指導教授：巫木誠 博士

國立交通大學工業工程與管理研究所

中文摘要

薄膜液晶顯示器(TFT-LCD)的生產主要有四個製程：陣列製程、彩色濾光片(CF)製程、組立製程及模組製程。其中組立製程生產 TFT 基板，而彩色濾光片製程生產的 CF 基板，將一片 TFT 基板和 CF 基板壓合後便形成 TFT-LCD 基板；為了方便生產現場的運送，TFT 基板和 CF 基板是由卡匣承載，一個卡匣約可承載 10 至 20 片基板。在重視良率的 TFT-LCD 產業，TFT-LCD 廠在生產時會面臨兩個決策問題，分別是基板配對決策及排序機派工決策，基板配對決策是將 TFT 基板和 CF 基板進行適當的配對來最大化產出良率，此決策問題在過去的研究中已有最佳演算法；在進行基板配對決策後，一組配對的 TFT/CF 基板可能分散在不同的卡匣中，因此需利用排序機對基板進行抽取重置(pick-and-replace)作業，以形成一個新卡匣。在生產環境中，有效率的排序機派工法則應能最大化下游組立製程的產能利用率，因此本研究將排序機派工決策模組化成動態規劃問題進行求解，在實例驗証後得到的結果指出，本研究所發展的動態規劃演算法顯著優於過去研究的啟發式解法。

關鍵字：動態規劃、排序機、派工、薄膜液晶顯示器

A Dynamic Programming Method for the Dispatching of TFT-LCD Sorting Process

Student: Te-Shan Chen Advisor: Dr. Muh-Cherng Wu

Department of Industrial Engineering and Management

National Chiao Tung University

Abstract

The manufacturing of TFT-LCD involves four main processes: array, color filter (CF), cell, and assembly processes. The array process produces TFT plates while the CF process produces CF plates, where the two processes transport plates in a container is called a cassette that contains about 10-20 plates. A TFT-LCD plate is the combination of a TFT plate and a CF plate. In a low yield environment, a TFT-LCD factory faces two decisions: *plate matching* and *sorter dispatching*. The plate mapping decision, which has been well solved in literature, examines how to map TFT plates to CF plates to maximize the resulting yield. Given an optimal plate mapping, a pair of TFT/CF plates may be located in different cassettes and requires a sorter to pick-and-replace the plates to form a new cassette. Practitioners report that an effective sorter-dispatching algorithm is needed to enhance the utilization of the downstream cell process. Other than using heuristic rules, this research uniquely models and solves the sorter-dispatching problem as a dynamic program. Numerical experiments indicate that the DP model significantly outperform the prior heuristic method.

Keywords: dynamic programming, dispatching, sorter, TFT-LCD

誌 謝

本論文得以順利完成，首先要感謝巫木誠教授在這兩年來的細心指導，耐心地訓練我在組織分析、邏輯思考與論文寫作方面的能力，除了學術領域的指引之外，於做事態度及人際相處方面，亦使學生獲益良多，在此致上最誠摯的謝意。同時，十分感謝口試老師許錫美博士、彭德保博士及陳文智博士，針對論文提供了許多寶貴的意見，使學生的論文更臻完善，為我兩年的研究生涯劃上了完美的休止符。

研究所兩年中，感謝同門的盧威豪、陳詠進、吳政翰、林劭函及顏豪君的互相激勵與安慰。此外，感謝同研究室的蘇泰盛學長、施昌甫學長，你們的出現，豐富了我的人生，也讓我在緊張、忙碌的生活中，在這研究之路上留下了許多美好的回憶。另外，特別感謝林聲宇學長在服役之餘，仍能抽空給我很多在進行研究方面的意見，讓學弟能獲得許多寶貴的經驗。

最後要深深感謝我的父母親陳安金先生與莊美華女士和弟弟伯齊，由於你們不斷的鼓勵與支持，讓我能夠埋首於論文學業上，衷心感謝你們所給予我的精神支持。僅以此論文獻給最敬愛的家人以及所有關心我的師長、朋友及學弟妹。

德珊

于 風城交大

2006-06-17

目 錄

表 目 錄

圖 目 錄

第一章　緒論

1.1 研究背景

薄膜電晶體顯示器(thin film transistor liquid crystal display，簡稱 TFT-LCD)製程可分為三個部分，分別是陣列(array)製程、組立(cell)製程和模組(module)製程。陣列製程主要是製造薄膜電晶體基板(TFT plate，簡稱 TFT 基板)；組立製程是將製造完成的薄膜電晶體基板，配合上自製或外購的彩色濾光片(color filter plate，簡稱 CF 基板)進行壓合作業，再注入液晶，形成 TFT-LCD 基板(TFT-LCD plate)，再將 TFT-LCD 基板切割成固定尺寸的數個面板(panel)。而模組製程則類似一般組裝作業，是將面板與背光模組、驅動 IC、印刷電路板等其他附件進行組立。

在組立製程中，TFT 基板和 CF 基板配對的結果會影響製程的良率。在組立製程中，一塊 TFT-LCD 基板(plate)可被分割成很多面板(panel)；若要得到一良品 TFT-LCD 面板，則在進行壓合時的 TFT 和 CF 面板都必須為良品才行；若 TFT 或 CF 面板其中有一為不良品，則壓合作業後的 TFT-LCD 面板便為不良品。如圖 1.1 所示，例 1 中 TFT 基板和 CF 基板的個別良率均為 75%，但在進行壓合作業後，因為配對關係，僅得到良率為 50%的 TFT-LCD 基板，而在例 2 中，TFT 基板和 CF 基板的個別良率同樣為 75%，因為較好的配對，經壓合後可得到良率為 75%的 TFT-LCD 基板。本研究稱此配對後的良率為配對良率(mapping yield)。

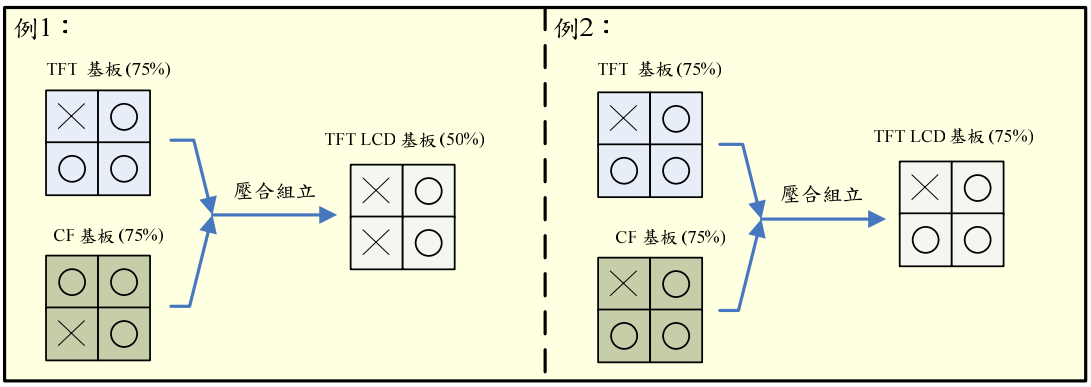


圖 1.1 TFT 基板和 CF 基板的配對良率

在生產現場，基板是以卡匣(cassette)為單位進行承載，通常一個卡匣可以承載約 10-20 片基板。基板製造的良率具有隨機特性，在一卡匣內，每一片基板的良率未必相同，即使良率相同，良品面板位置的分佈也未必一致。為了提高配對良率，工廠通常會將多個 CF 和 TFT 卡匣內的基板整合配對，一配對成組的 CF 和 TFT 基板互稱為配對基板(mapping plate)。假設一個卡匣可以承載 n 個基板，當 TFT 和 CF 各有 N 個卡匣時，則基板配對方式共有$(N\times n)!$種組合。在這些組合中，如何找到最佳良率的配對組合，我們稱之為基板配對決策 (plate mapping decision)，在過去的研究中，已有最佳演算法能找到最大良率之基板配對決策 。

基板配對決策之後，一 CF 卡匣內的基板所對應的 n 個 TFT 基板可能雜散在不同的 TFT 卡匣內，為了方便 Cell 製程的組立作業，工廠通常會先進行基板抽換及重置作業(pick-and-replace)，亦即將一 CF 卡匣所配對的 TFT 基板從原卡匣內全部抽出，置放在一個新的 TFT 卡匣內，以便壓合組立，此新的 TFT 卡匣稱為目標卡匣(target cassette)。如圖 1.2 所示，CF 卡匣 X 內有三個基板(x_1, x_2, x_3)，其配對基板(q_1, r_2, s_3)分別存放在 TFT 卡匣 Q, R, S 內。抽換作業將 TFT 基板 q_1, r_2, s_3 從其原卡匣抽出，置放在新的 TFT 卡匣 Y 內，使卡匣 X 內的所有基板，其配對基板均在卡匣 Y 內，抽換作業完成後的卡匣 Y 為目標卡匣。

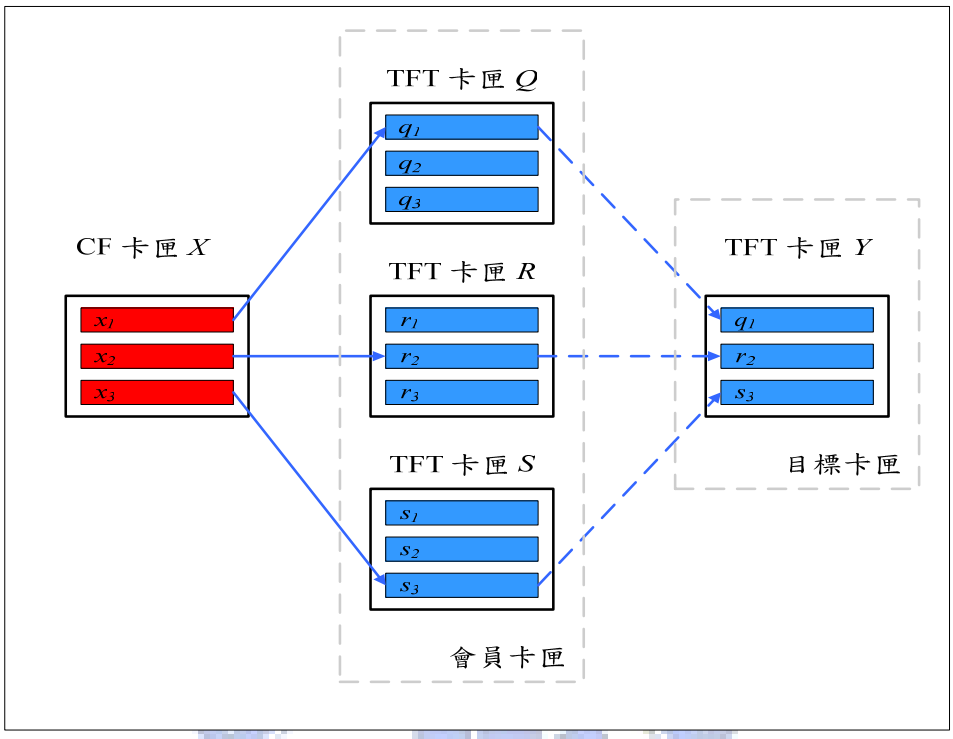


圖 1.2 抽換作業示意圖

進行抽換作業的設備稱為排序機(sorter)。如圖 1.3 所示，一個典型排序機的組成包括一個機器手臂(robot)，一個輸出埠(output port)，數個輸入埠(input port)。進行抽換作業時，輸出埠置放一空卡匣，機器手臂的功能是從輸入埠所置放的卡匣中抽出 TFT 基板，將其放在輸出埠的卡匣，以形成目標卡匣。

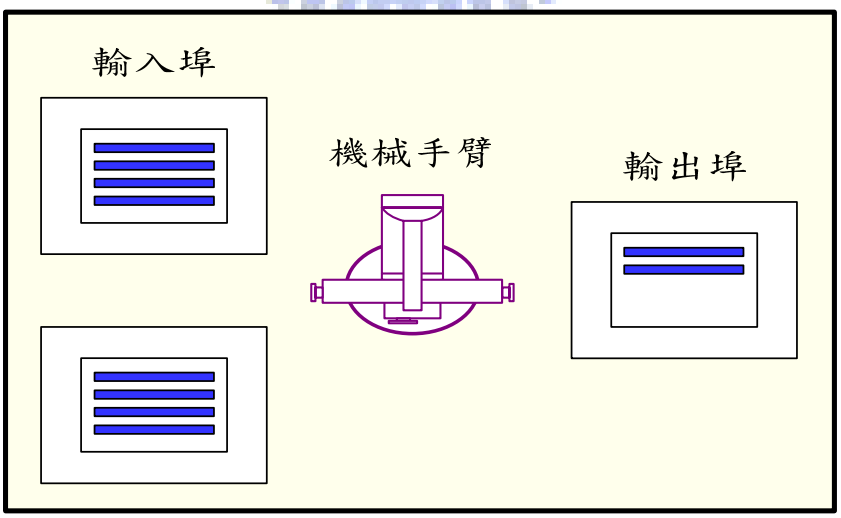


圖 1.3 排序機設備

經基板配對決策後，一 CF 卡匣內的基板所對應的 n 個 TFT 基板可能會分散在多個 TFT 卡匣中，這些 TFT 卡匣稱為該 CF 卡匣的會員卡匣(member cassette)，本研究利用卡匣配對矩陣(cassette mapping matrix, 簡稱配對矩陣)整合各 CF 卡匣與其會員卡匣間的卡匣對應關係(cassette-to-cassette relationship)，配對矩陣可用 $M_{N*N} = [m_{ij}]$, $1 \leqq i \leqq N$, $1 \leqq j \leqq N$ 表示，若 TFT 卡匣 j 為 CF 卡匣 i 之會員卡匣，則 m_{ij}=1，反之則 m_{ij}=0。圖 1.4 為 4 個 CF 卡匣和 4 個 TFT 卡匣進行基板配對決策後所得的卡匣配對矩陣 M_{4*4}，如圖所示，CF 2 卡匣的配對基板分散在 TFT 1, TFT 2 及 TFT 3 卡匣中，因此 TFT1, TFT 2 及 TFT 3 三個卡匣為目標卡匣 CF2 的會員卡匣，抽換作業需將配對基板從三個會員卡匣中抽出，放置在一新的 TFT 卡匣內，才能完成目標卡匣的產出。

	TFT 1	TFT 2	TFT 3	TFT 4
CF 3	1	0	0	0
CF 4	0	1	0	1
CF 2	1	1	1	0
CF 1	1	1	1	1

圖 1.4 配對矩陣

在排序機的派工過程中，會衍生出兩個問題，分別是目標卡匣的產出順序決策(output sequence decision)及會員卡匣上下排序機的順序決策(input sequence decision)，這兩個問題為排序機派工的主要決策；決策一為目標卡匣的產出順序(output sequence, 簡稱產出順序)，如圖 1.5 中所示，產出順序可為 C1→C2→C3→C4，或 C3→C4→C2→C1 等，共 4!種情形；決策二為會員卡匣上下排序機的順序(input sequence, 簡稱投入順序)，如圖 1.5 中所示，以 CF2 目標卡匣為例，其會員卡匣為 TFT1, TFT2, TFT3，在排序機只有一個輸入埠的情形下，投入順序為 T1→T2→T3 或 T1→T3→T2 等，共 3!種情形。

決策一例：

目標卡匣產出順序：C1→ C2→ C3→ C4

產出順序		TFT 1	TFT 2	TFT 3	TFT 4
1	CF 1	1	1	1	1
2	CF 2	1	1	1	0
3	CF 3	1	0	0	0
4	CF 4	0	1	0	1

目標卡匣產出順序：C3→ C4→ C2→ C1

產出順序		TFT 1	TFT 2	TFT 3	TFT 4
1	CF 3	1	0	0	0
2	CF 4	0	1	0	1
3	CF 2	1	1	1	0
4	CF 1	1	1	1	1

決策二例：

產出順序		TFT 1	TFT 2	TFT 3	TFT 4
1	CF 3	1	0	0	0
2	CF 4	0	1	0	1
3	CF 2	1	1	1	0
4	CF 1	1	1	1	1

CF2會員卡匣為TFT1, TFT2, TFT3

會員卡匣上下排序機順序：

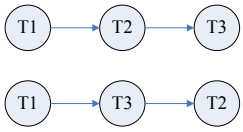


圖 1.5 排序機派工決策

組立製程為流線型生產(flow line)，排序機下游為組立壓合機台(assembly machine)，作業內容是將配對好的 TFT 基板和 CF 基板進行壓合。其生產特性為，在不缺料的情形下，每隔固定時間 T 可以產出一個壓合卡匣，反之在缺料的情形下，壓合機的產能則會閒置；因此壓合機對排序機的要求是每隔固定時間 T 需產出一目標卡匣，若目標卡匣的產出時間變異過大時，壓合機的產能可能會閒置；因此本研究著焦於排序機應該如何派工，以最大化壓合機台利用率，以期能提高 TFT-LCD 廠的最終產出。

1.2 研究目的

根據上述的背景，本研究以卡匣配對矩陣作為輸入，發展一演算法來進行排序機派工決策，妥善規劃產出順序和投入順序，以最大化下游組立壓合機台利用率。

1.3 論文組織

本論文後續章節安排如下，第二章將更深入介紹 TFT-LCD 製程及 TFT-LCD 製造管理相關文獻，第三章介紹研究構想，第四章探討研究方法－動態規劃演算模型，第五章是實例驗證，第六章則是結論及未來研究的方向。

第二章　文獻回顧

2.1 TFT-LCD 製程簡介

薄膜液晶顯示器(TFT-LCD)是利用液晶材料的特性，透過外加電路驅動液晶轉向，以控制外部光源穿透與否，進而達到明暗不同的效果，若在結構中加上彩色濾光片，便能顯現色彩化的畫面。

TFT-LCD 之生產製造技術結合半導體、化學、材料、光電等產業之製造技術，主要製程可分為三個部份：(1) 陣列製程(Array Process)，(2) 組立製程(Cell Process)，(3) 模組製程(Module Process)，如圖 2.1 所示。

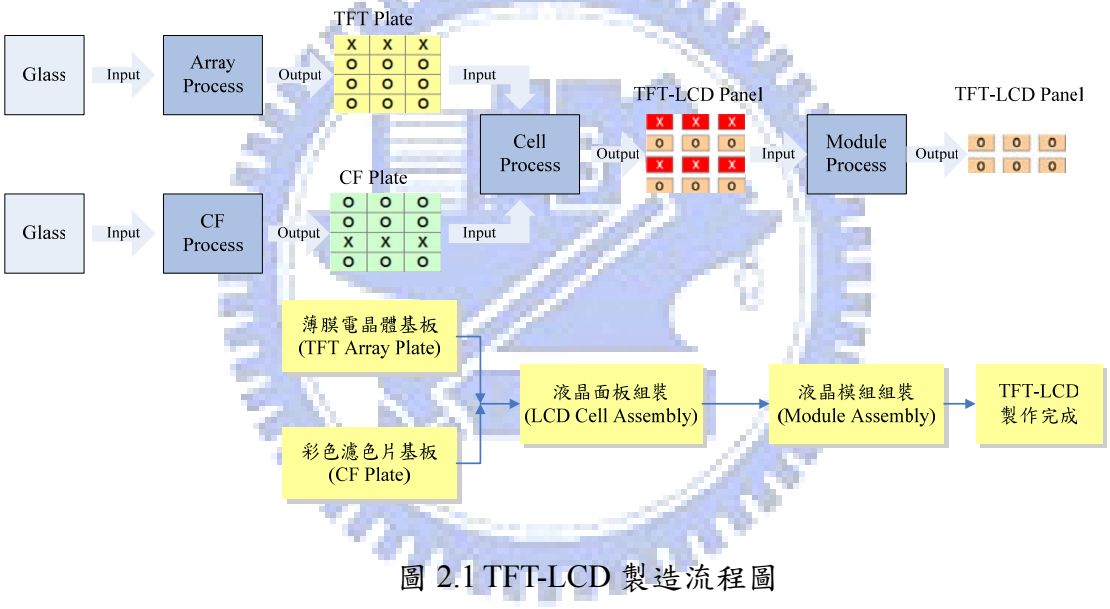


圖 2.1 TFT-LCD 製造流程圖

- 薄膜電晶體陣列(TFT Array)製造技術主要是將玻璃基板透過類似半導體製造技術(鍍膜、曝光、顯影、蝕刻等技術)，在基板上形成為數眾多的電晶體之陣列基板；這方面國內技術成熟，一般良率均達 90%以上。
- 液晶面板組立製造技術主要是利用滴注式液晶灌注製程(ODF，One-Drop Fill)，將組立製程完成的基板與彩色濾光片基板分別作配對對準，再進行框膠燒成，並注入液晶，並透過機械對位完成組立，最後切割成預定尺寸的面板(Panel)，並將偏光板貼附，最後做檢測工作。這是目前最值得研究之製程。

- 模組製程，主要是將切割完成的面板與驅動 IC、電路板、背光板等外部零組件組立起來，之後再做最後的檢查；這方面的技術難度不高，良率接近100%。

2.2 TFT-LCD 製造管理相關文獻

過去 TFT-LCD 製造管理的相關文獻，主要可分為兩大類，第一類為基板配對決策方法之研究，而第二類則著重在排序機派工決策之研究。

在基板配對決策的研究中，主要有三篇主要的文獻，分別是楊佳翰[3]、楊毅臻[2]及張東華[4]，在這部分的研究已有最佳化配對良率的演算法，因此不再進行探討。而在排序機派工決策方面，則有楊佳翰[3]、楊毅臻[2]及王君豪[5]三篇文獻，分別在以下進行簡介。

為縮短卡匣上下排序機的次數，楊佳翰[3]提出以基因演算法(Genetic algorithm, GA)及模擬退火法(Simulated Annealing, SA)發展良率配對的方法。該研究假設排序機有 1 個輸出埠、k 個輸入埠，若有 N 組卡匣的基板需要配對，該研究先將卡匣分群，一群包含有 k 個 TFT 卡匣和 k 個 CF 卡匣，然後以線性規劃法(Linear Programming, LP)求解一群內基板配對的最佳解。該研究可有效縮短卡匣上下排序機的次數，但是所規劃的基板配對良率未必是最佳解。

為確保基板配對良率，進而縮短卡匣上下排序機的次數，楊毅臻[2]應用匈牙利指派法求解良率配對，然後發展啟發式方法(Heuristic Methods)來決定目標卡匣產出順序(簡稱產出順序)及會員卡匣上下排序機的順序(簡稱投入順序)，期以最小化排序機的總作業時間。該研究雖然可確保最佳配對良率，然而，在求解良率配對的線性規劃問題時，該研究假設該線性規劃問題只有一組最佳解，並據以求解排序機派工問題。

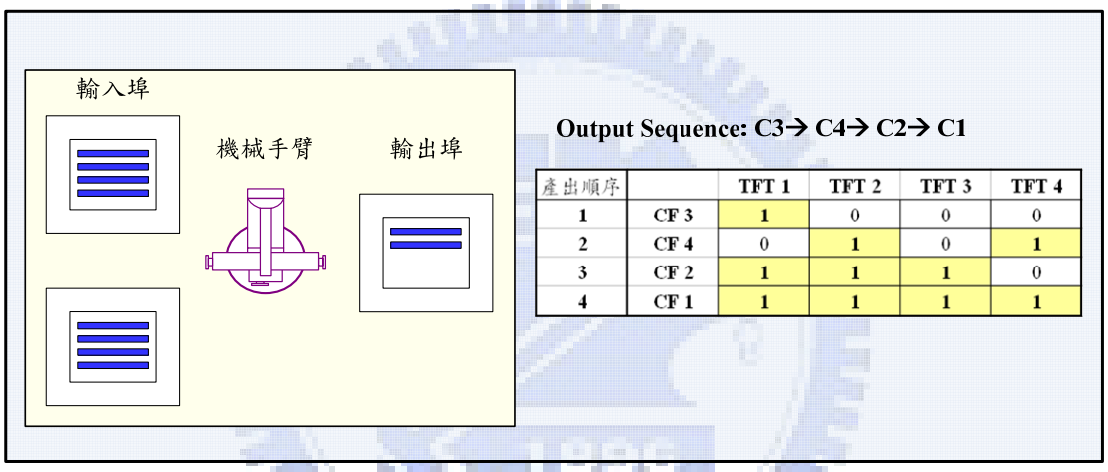
為配合 TFT-LCD 連續型生產的特性，王君豪[5]發展基因啟發式演算法來求解排序機派工問題，其研究方法利用啟發式解法進行求解投入順序問題(input sequencing problem)，下游組立壓合機台(assembly machine)利用率為績效衡量指

標，並以此作為基因演算法的適應函數，來進行規劃目標卡匣的產出順序(output sequencing problem)，此基因啟發式解法同時整合排序機作業及下游組立機台作業，以整體性的觀點增進組立製程的生產效率。

針對上述文獻回顧，吾人可以發現王君豪[5]雖以整體性的觀點來進行排序機派工決策，但在投入順序問題上仍有改善之空間，因此本研究的重點在發展一動態規劃法(Dynamic Programming, DP)以改善王君豪[5]的研究中，使用啟發式解法求解投入順序決策問題不足的部分。

第三章　研究構想

在本研究對會員卡匣上下排序機的順序(input sequence, 簡稱投入順序)進行決策前，必須先確定三項資訊，第一項是目標卡匣的產出順序(output sequence, 簡稱產出順序)，第二項是該產出順序下的卡匣配對矩陣，而第三項則是排序機上的輸入埠個數，之後才得以發展方法對投入順序進行決策；以圖 3.1 為例，產出順序為 C3→ C4→ C2→ C1，在此產出順序下的卡匣配對矩陣 M_{4*4}，且排序機輸入埠個數為 2 (p=2)，此時會員卡匣投入順序可為(T1,■)→ (T2, T4)→ (T1, T2)→…→(T3, T4), ■代表排序機輸入埠上無卡匣。



產出順序		TFT 1	TFT 2	TFT 3	TFT 4
1	CF 3	1	0	0	0
2	CF 4	0	1	0	1
3	CF 2	1	1	1	0
4	CF 1	1	1	1	1

圖 3.1 研究前提

本研究之構想是將投入順序決策描述成動劃規劃(Dynamic Programming, DP)模型，再以系統化的方法來決定最佳的投入順序，為了便於了解動態規劃問題的一般性結構，以下約略介紹動態規劃的典型問題－馬車問題(Traveling Salesman Problem, TSP)。

TSP 為動態規劃的典型問題，問題背景為一旅行者為了從美國的東岸到西岸，旅途中必須選擇經過的城市，使得總路途最短(或總成本最低)，該問題可一般化成網路結構，分成數個階段(stage)，相當於旅途中經過的各個地區(area)，各階段中有許多狀態(state)，代表旅途中可能會經過的城市，旅行者所進行的決策是，在階段 i 到階段 i+1 的遷徙過程中，選擇下一個到達的城市，直到終點為止。

圖 3.2 舉例說明 TSP 網路，旅行者從 A 城市出發，終點為 J 城市，旅途會經過四個地區，代表四個階段，每個地區包含了數個城市，代表各階段中數個可供選擇的狀態，旅行者所進行的決策是，在地區 i 遷徙到地區 i+1 的過程中，選擇下一個要前往的城市，直到到達城市 J 為止。

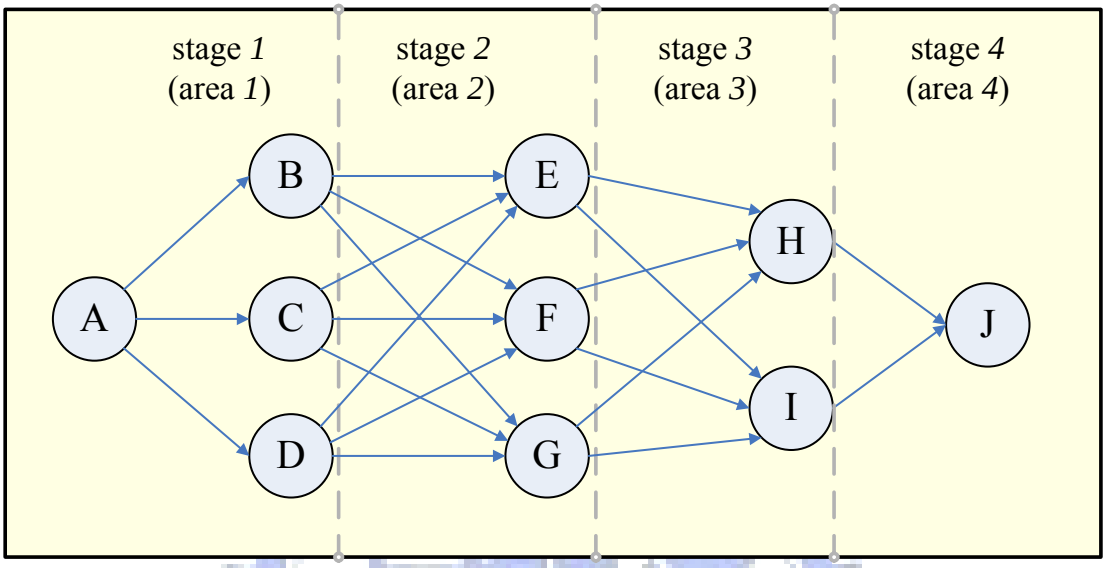


圖 3.2 馬車問題例圖

本研究在下一個章節中，介紹如何將投入順序決策問題描述成為動態規劃模型，並建構各階段、狀態以形成一完整的動態規劃網路，並求解最佳的投入順序組合。

第四章　研究方法

本研究主要的決策問題是規劃投入順序，因此本章節重點在將投入順序決策建構成動態規劃模型(Dynamic Programming Model)，其內容包含有二，分別是將投入順序決策問題建構成完整的網路，再依動態規劃特有的系統化方法找尋最佳投入順序。

4.1 建構網路模型

欲將投入順序決策過程描述成動態規劃網路，其過程主要可分為兩個部分，第一部分是基本的網路模型的建立，此基本架構包含了階段(stage), 狀態(state), 批次(batch), 而基本架構的建立透過 Stage Modeling, State Modeling 和 Batch Modeling 三個模組來完成；而第二部分為加強網路結構，利用 State Deployment 模組將原本不完整的狀態，重行部署成為一個或多個完整的狀態，目的是為了使網路架構更加完整。以下吾人便依序對 Stage Modeling, State Modeling, Batch Modeling 和 State Deployment 四模組進行介紹。

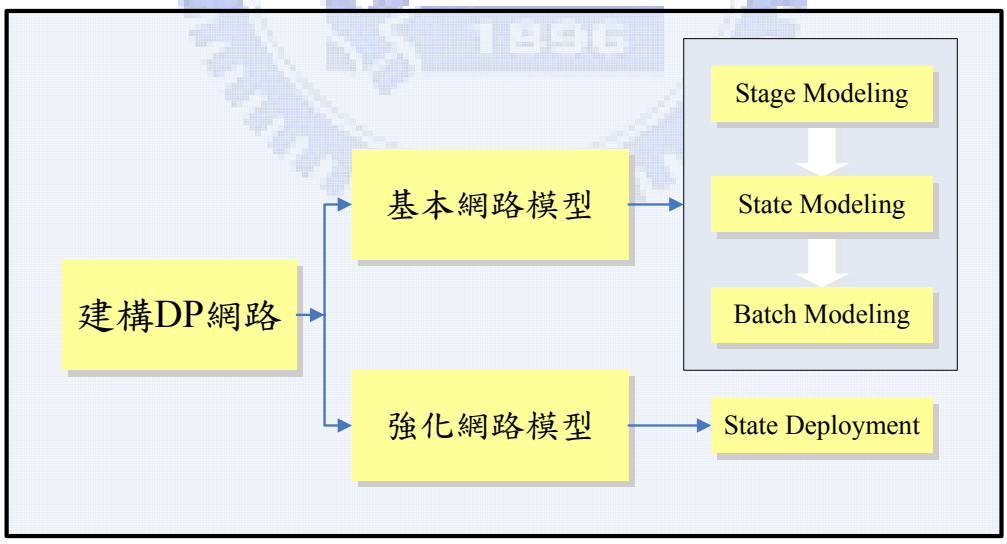


圖 4.1 建構 DP 網路流程圖

■ 符號定義

參數定義：

N: CF/TFT 卡匣的總數

n: CF/TFT 卡匣上的基板總數

$M_{N*N} = [m_{ij}]$, 卡匣配對矩陣, $1 \leqq i \leqq N$, $1 \leqq j \leqq N$

p: 排序機上輸入埠的個數

K_i: 產出順序中排序第 i 位的目標卡匣

$Member(K_i)$: 目標卡匣 K_i 的會員卡匣集合

Q_i: $Member(K_i)$所包含的元素個數

變數定義：

$Stage\_i$: 階段 i, $1 \leqq i \leqq N$

$seq_{i,j}$: 在階段 i 中的狀態(投入順序) j, $1 \leqq j \leqq s(i)$

$s(i)$: 階段 i 中的狀態個數

$B_{(i,j),k}$: 在 $seq_{i,j}$ 中的第 k 個批次, $1 \leqq k \leqq b(i)$

$b(i)$: 在階段 i 中的每個狀態所包含的批次個數

4.1.1 Stage Modeling

Stage Modeling 是建立基本網路架構的第一步驟，其目的是建立出網路中所有的階段 $Stage\_i$, $1 \leqq i \leqq N$，在本研究方法中，階段 i(stage i)是指能產出目標卡匣 K_i 的所有投入順序(state)所形成的集合，因此階段 i 可表示成 $Stage\_i = \{seq_{i,j} | 1 \leqq j \leqq s(i)\}$，以圖 4.2 為例，CF/TFT 卡匣總數為 4，產出順序為{C3, C4, C2, C1}，因此該網路包含四個階段，假設在 p=2 的情況下，對目標卡匣 K_3(K_3 =C2)進行 Stage Modeling，則可得 $Stage\_i = \{seq_{3,1}, seq_{3,2}, seq_{3,3}\}$，其中 $seq_{3,1}$=(T1, T2)→ (T3, ■), $seq_{3,2}$=(T1, T3)→ (T2, ■), $seq_{3,3}$=(T2, T3)→ (T1, ■)，■代表排序機輸入埠上無任何卡匣，因此可得知在第 3 階段為 3 個投入順序(state)的集合，意指要產出目標卡

匣 K_3，共有三種不同的投入順序(state)可供選擇。

產出順序		TFT 1	TFT 2	TFT 3	TFT 4
1	CF 3	1	0	0	0
2	CF 4	0	1	0	1
3	CF 2	1	1	1	0
4	CF 1	1	1	1	1

$p = 2$

圖 4.2 配對矩陣

Stage Modeling 主要的目的是為動態規劃網路建構出所有的階段，同時必須計算出各個階段中包含的狀態個數 $s(i)$, $1 \leqq i \leqq N$；Stage Modeling 以 Q_i 作為計算 $s(i)$的輸入資料，而 Q_i 和排序機上輸入埠個數 p 的關係可表示成一等式，$Q_i = p * q_i + r_i$, 根據等式中 q_i 的數值可分成兩種計算 $s(i)$的方法：

方法一：若 $q_i > 0$, 則

$$s(i) = C_p^{Q_i} \cdot C_p^{Q_i - p} \cdots C_p^{p + r_i} \cdot 1 = \prod_{i=0}^{q_i} C_p^{Q_i - i \cdot p} \cdot 1$$

方法二：若 $q_i = 0$, 則

$$s(i) = 1$$

以圖 4.2 為例，分別對目標卡匣 K_1, K_2, K_3, K_4 進行 Stage Modeling，所得到結果為：(1) K_1=C3, 因為 Q_1= 1≦ 2, 所以應利用方法二求狀態個數, 求得 $s(1)$= 1；(2) K_2=C4, 因為 Q_2= 2≧2, 所以應利用方法一求狀態個數，求得 $s(2)=C_2^2 \cdot 1 = 1$ ；(3) K_3=C2, 因為 Q_3= 3≧2, 所以應利用方法一求狀態個數，求得 $s(3)=C_2^3 \cdot 1 = 3$ ；(4) K_4=C1, 因為 Q_4= 4≧2, 所以應利用方法一求狀態個數，求得 $s(4)=C_2^4 \cdot C_2^2 \cdot 1 = 6$ 。根據以上所得到的結果，可得到如圖 4.3 的基本網路架構雛形，其中階段 0 及階段 5 為虛擬階段(virtual stage)，分別做為起點與終點，其餘四個為真實階段，階段 1 中狀態個數為 1，階段 2 中狀態個數為 1，階段 3 中狀

態個數為 3，階段 4 中狀態個數為 6。

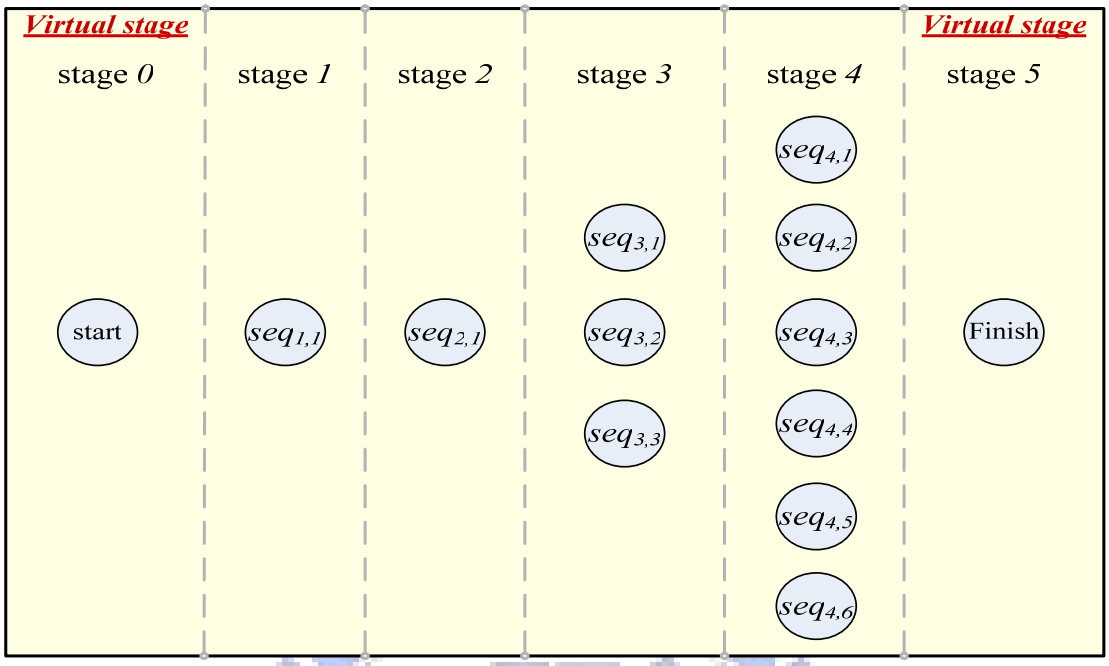


圖 4.3 網路架構雛形

4.1.2 State Modeling

State Modeling 為建構基本網路的第二步驟，其目的是建立出網路中的所有狀態(state) $seq_{i,j}$, $1 \leqq i \leqq N$, $1 \leqq j \leqq s(i)$，在本研究中，在階段 i 中的狀態(state)是指能產生目標卡匣 K_i 的投入順序，是由 $b(i)$個批次所構成，且單一批次包含最多 p 個會員卡匣，因此在階段 i 中的狀態可表示成 $seq_{i,j} = B_{(i,j),1} \rightarrow B_{(i,j),2} \rightarrow \ldots \rightarrow B_{(i,j),b(i)}$，是由 $b(i)$個批次所組成的批次序列(batch sequence)；以圖 4.2 中目標卡匣 K_3 舉例說明，在輸入埠個數 p=2 的情況下，階段 3 包含了 3 個狀態，分別為 $seq_{3,1}$=(T1, T2)→ (T3, ■), $seq_{3,2}$=(T1, T3)→ (T2, ■), $seq_{3,3}$=(T2, T3)→ (T1, ■)，■代表排序機輸入埠上無任何卡匣，由上述可得知在階段 3 中，3 個狀態均是由 2 個批次所組成的批次序列。

State Modeling 主要的目的是為動態規劃網路建構出各階段中所有的狀態，其重點在計算出階段 i 中的各個狀態中包含的批次個數 $b(i)$, $1 \leqq i \leqq N$；State Modeling 以 Q_i 作為計算 $b(i)$的輸入資料，而 Q_i 和排序機上輸入埠個數 p 的關係

可表示成一等式，$Q_i = p * q_i + r_i$, 根據等式中 q_i 和 r_i 的數值可分成三種計算 $b(i)$ 的方法：

方法一：若 $q_i > 0$ & $r_i = 0$, 則 $b(i) = q_i$

方法二：若 $q_i > 0$ & $r_i > 0$, 則 $b(i) = q_i+1$

方法三：若 $q_i = 0$, 則 $b(i) = 1$

以圖 4.2 為例，分別對目標卡匣 K_1, K_2, K_3, K_4 進行 State Modeling，所得到結果為：

(1) K_1=C3, 因為 $q_1 = 0$, 所以應利用方法三求批次個數, 求得 $b(1)= 1$；

(2) K_2=C4, 因為 q_2=1> 0 & r_2=0, 應利用方法一求批次個數，求得 $b(2)= q_2 = 1$；

(3) K_3=C2, 因為 q_3= 1> 0 & r_3=1, 應利用方法二求批次個數，求得 $b(3)= q_3 + 1$=2；

(4) K_4=C1, 因為 q_4= 2> 0 & $r_4 = 0$,應利用方法一求批次個數，求得 $b(4)= q_4 = 2$。

在完成 State Modeling 後，可得到基本網路架構，包含了階段、狀態，以及每個狀態中所包含的批次個數，根據建構完成的基本網路，可歸納出狀態的特性：在階段 i 中，每個狀態所包含的批次個數必定同為 $b(i)$，因此在階段 i 中的狀態可表示成為 $seq_{i,j} = B_{(i,j),1} \rightarrow B_{(i,j),2} \rightarrow \ldots \rightarrow B_{(i,j),b(i)}$, $b(i)$為批次個數。

4.1.3 Batch Modeling

Batch Modeling 為建構基本網路的最後一個步驟，目的是建立出網路中的所有狀態內所包含的批次 $B_{(i,j),k}$, $1 \leqq i \leqq N$, $1 \leqq j \leqq s(i)$, $1 \leqq k \leqq b(i)$；Batch Modeling 主要透過重複不放回隨機選取的程序(Random Select without Replacement Procedure)來產生一個狀態內的所有批次序列，該程序以 $Member(K_i)$作為抽樣母體，母體大小為 $Q_i = p * q_i + r_i$，進行 $b(i)$次的重複不放回隨機選取，在每次選取時，若母體內的會員卡匣個數仍大於 p，則隨機選取 p 個會員卡匣成為一個批次，此批次為一完整的批次(filled batch)，若母體個數已小於 p，則選取剩餘的 r_i 個會員當作狀態中的最後一個批次，則此批次為一不完整的批次(unfilled batch)，透過該程序所得到的全部組合情形便形成階段 i 中的 $s(i)$個狀態。

在完成 Batch Modeling 後，便可得到基本的網路架構，包含了階段、狀態以

及狀態內部所包含的批次，以圖 4.2 為例，CF/TFT 卡匣總數為 4，產出順序為{C3, C4, C2, C1}，經過 Stage Modeling, State Modeling, 及 Batch Modeling 後，可得到基本網路架構如圖 4.5 所示，其中包含四個階段，階段 1 有一個狀態，階段 2 有一個狀態，階段 3 有三個狀態，階段 4 中有六個狀態，而經過 Batch Modeling 後，各狀態內的批次序列如圖 4.3 所示。

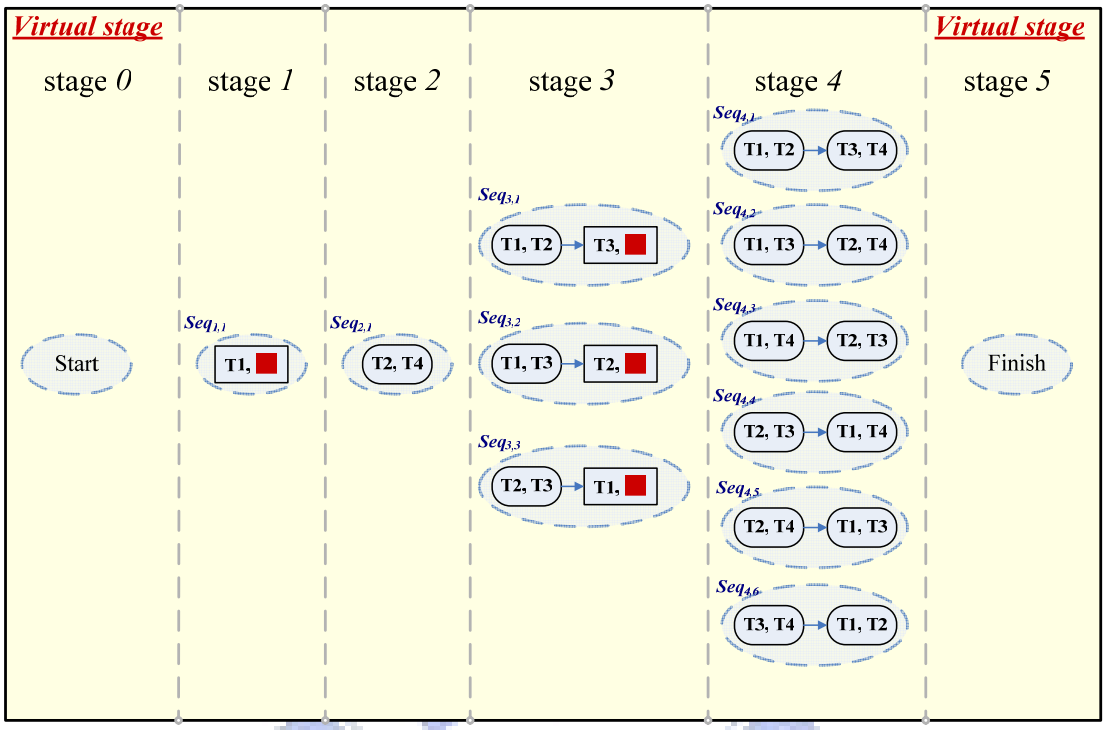


圖 4.4 基本網路架構圖

根據基本網路架構，吾人可歸納得到一結論，在同一階段中，狀態必屬於相同的組態，而在不同的階段間，狀態可分為以下三種不同的組態，以 $Q_i = p * q_i + r_i$ 此等式做為判別的依據：

組態一：若 $q_i > 0$ & $r_i = 0$, 則該狀態由 q_i 個完整的批次所組成；

組態二：若 $q_i > 0$ & $r_i > 0$, 則該狀態由 q_i 個完整的批次及 1 個不完整的批次所組成；

組態三：若 $q_i = 0$, 則 $b(i) = 1$，則該狀態由 1 個不完整的批次所組成；

以圖 4.2 之配對矩陣為輸入資料，在 p=2 情況下，分別探討目標卡匣 K_1, K_2, K_3, K_4

進行 Batch Modeling 後，各階段內的狀態組態情形：

(1) 階段 1, $Q_1 = 1 = 2*0 + 1$, $q_1 = 0$, 因此階段 1 內之狀態屬於組態三，由一個不完整的批次所組成；

(2) 階段 2, $Q_2 = 2 = 2*1 + 0$, $q_2 = 1$, $r_2 = 0$, 因此階段 2 內之狀態屬於組態一，由一個完整的批次所組成；

(3) 階段 3, $Q_3 = 3 = 2*1 + 1$, $q_3 = 1$, $r_3 = 1$, 因此階段 3 內之狀態屬於組態二，由一個完整的批次及一個不完整的批次所組成；

(4) 階段 4, $Q_4 = 4 = 2*2 + 0$, $q_4 = 2$, $r_4 = 0$, 因此階段 4 內之狀態屬於組態一，由二個完整的批次所組成；

上述所得到結果，以圖 4.4 整理如下所示：

產出順序		TFT 1	TFT 2	TFT 3	TFT 4
1	CF 3	1	0	0	0
2	CF 4	0	1	0	1
3	CF 2	1	1	1	0
4	CF 1	1	1	1	1

$p = 2$

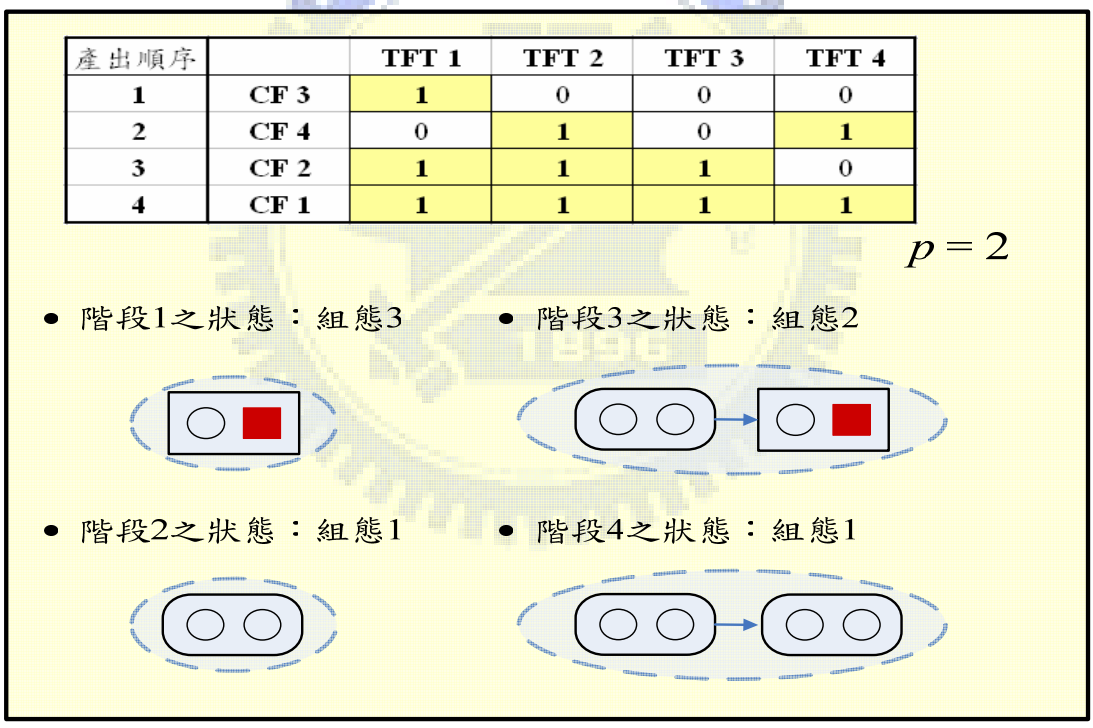


圖 4.5 狀態之組態例圖

4.1.4 State Deployment

經過 Stage Modeling, State Modeling 及 Batch Modeling 三個步驟，可得到基本的網路模型，同時可歸納出狀態具有三種不同的組態，如圖 4.5 所示，分別為：組態一：狀態均由完整的批次所組成，

組態二：狀態由至少兩個批次組成，且最後一個批次為不完整批次，

組態三：狀態只由一不完整的批次所組成；

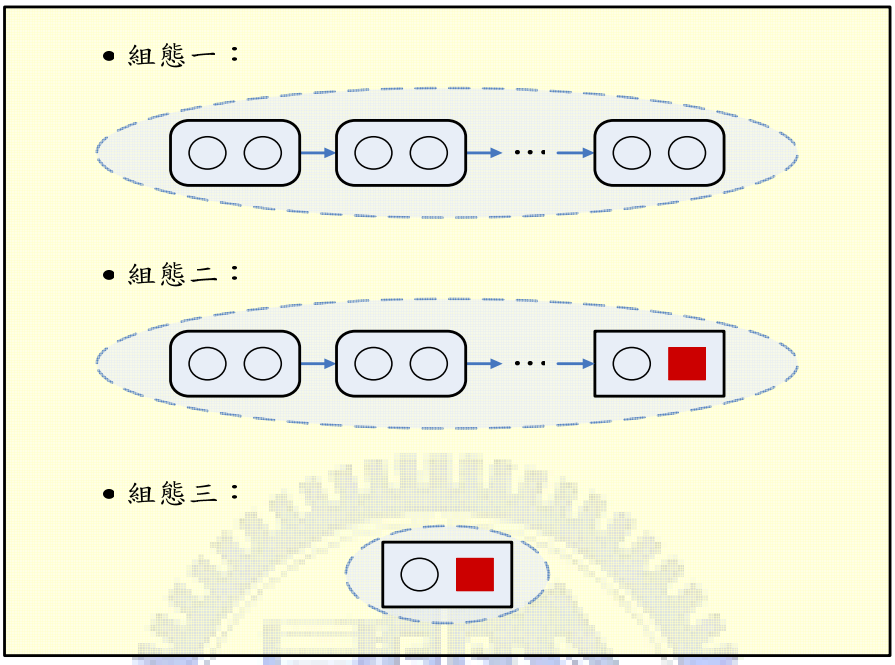


圖 4.6 三種狀態組態示意圖

在組態二及組態三中，狀態內的批次序列均包含一個不完整的批次，在本研究方法中，帶有不完整批次的狀態(state with unfilled batch)意謂著一個無法提供完整資訊的決策組合，因此本研究方法透過 State Deployment 模組來重組組態二及組態三的狀態，使一個帶有不完整批次的狀態能夠重組為一個或是多個均帶有完整批次的狀態，如圖 4.6 所示，原本屬於組態二或組態三的狀態能盡可能重組為組態一的狀態，如此一來能使網路架構更加完整。以下分別介紹 State Deployment 模組如何對屬於組態三及組態二的狀態進行重組。

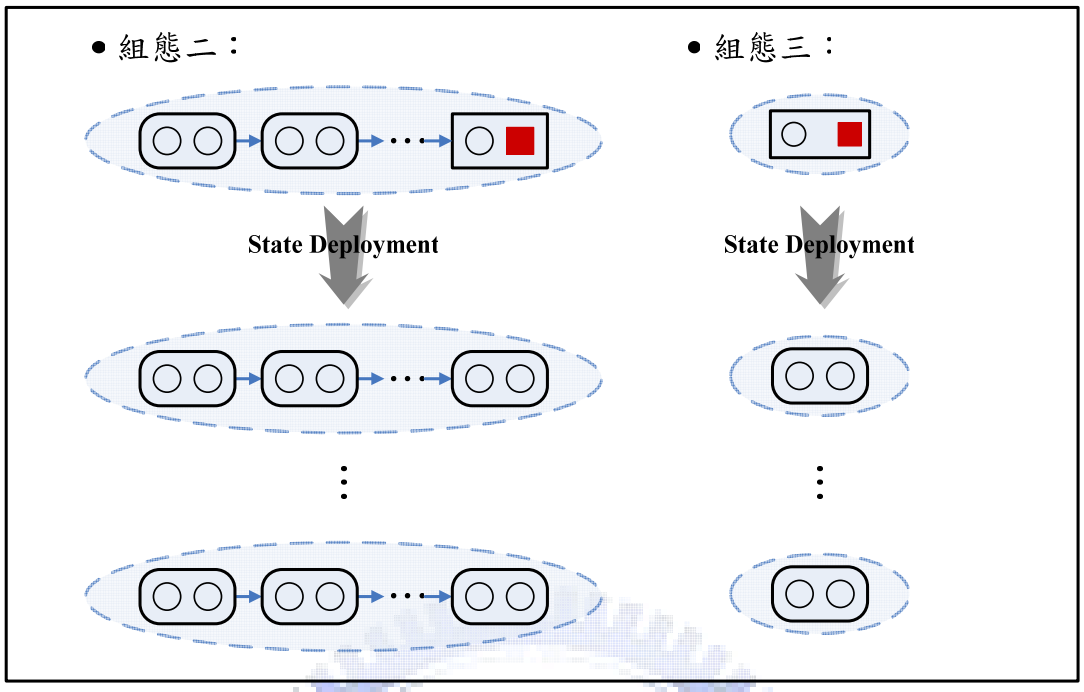


圖 4.7 狀態重組示意圖

組態三的狀態是由一個不完整的批次所構成，State Deployment 模組的目的是，將原本屬於組態三的狀態重組為一個或多個帶有一個完整批次的狀態；為方便說明，吾人將原本狀態中不完整的批次定義成 B_u, State Deployment 模組對 B_u 進行重組的方式會因為 B_u 所處的階段而有所不同，分成以下兩種情況討論：

情形一：若 B_u 在階段 i 中，$i = 1$, 則不進行重組，維持原本狀態組態。

情形二：若 B_u 在階段 i 中，$i > 1$, 則進行重組。

如上所述，可得知只有 B_u 在階段 i, $i > 1$ 的情形之下才需要進行重組，若目前要對在階段 i 的 B_u 進行重組，則 State Deployment 模組的輸入資料為階段 i-1 中，第一個狀態到第 $s(i\text{-}1)$個狀態中的最後一個批次 $B_{(i-1,\,j),\,b(i-1)}$, $1 \leqq j \leqq s(i\text{-}1)$, 吾人定義為 B_f，State Deployment 模組的演算構想是從 B_f 中挑選卡匣，以填滿 B_u 中的空缺，以形成新的批次 B_r, 進行重組後，狀態數目會變成原來的 $s_d(i)$倍。演算法內容說明如下：

Procedure *State_Deployment_for Case3*(B_u, B_f, $s_d(i)$, B_r)

$B_u = \{ u_i | 1 \leqq i \leqq r_i \}$ /*B_u 是由 r_i 個卡匣形成的集合*/

$B_f = \{ f_j | 1 \leqq j \leqq q \}$ /*B_f 是由 q 個卡匣形成的集合, $q \leqq p$*/

$B_d = B_f - B_u$, $N(B_d) = d$ /*B_d 為 B_f 和 B_u 的差集合*/

if $d \leqq (p - r_i)$,

Select the d cassettes of B_d to fill the vacant position in B_u to form B_r;

Number of states deployed $s_d(i) = 1$;

Else if $d > (p - r_i)$,

Random select $(p - r_i)$ cassettes from B_d to fill the vacant position in B_u to form B_r;

Number of states deployed $s_d(i) = C^d_{p-r_i}$;

End if

藉由 State Deployment 的演算，若原本屬於組態三的狀態位於階段 i, i =1, 則不必對該狀態進行重組，如下圖 4.7 所示：

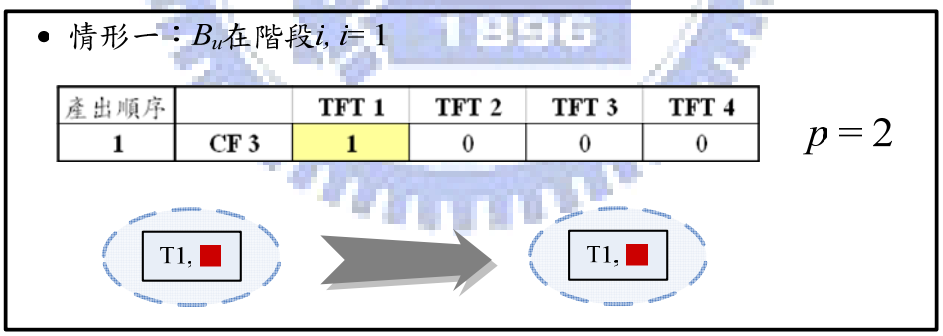


圖 4.8 組態三之情形一狀態重組例圖

若原本為組態三的狀態位於階段 i, i> 1, 則經過上述的程序，重組成 $s_d(i)$個新狀態，如下圖 4.8 所示：

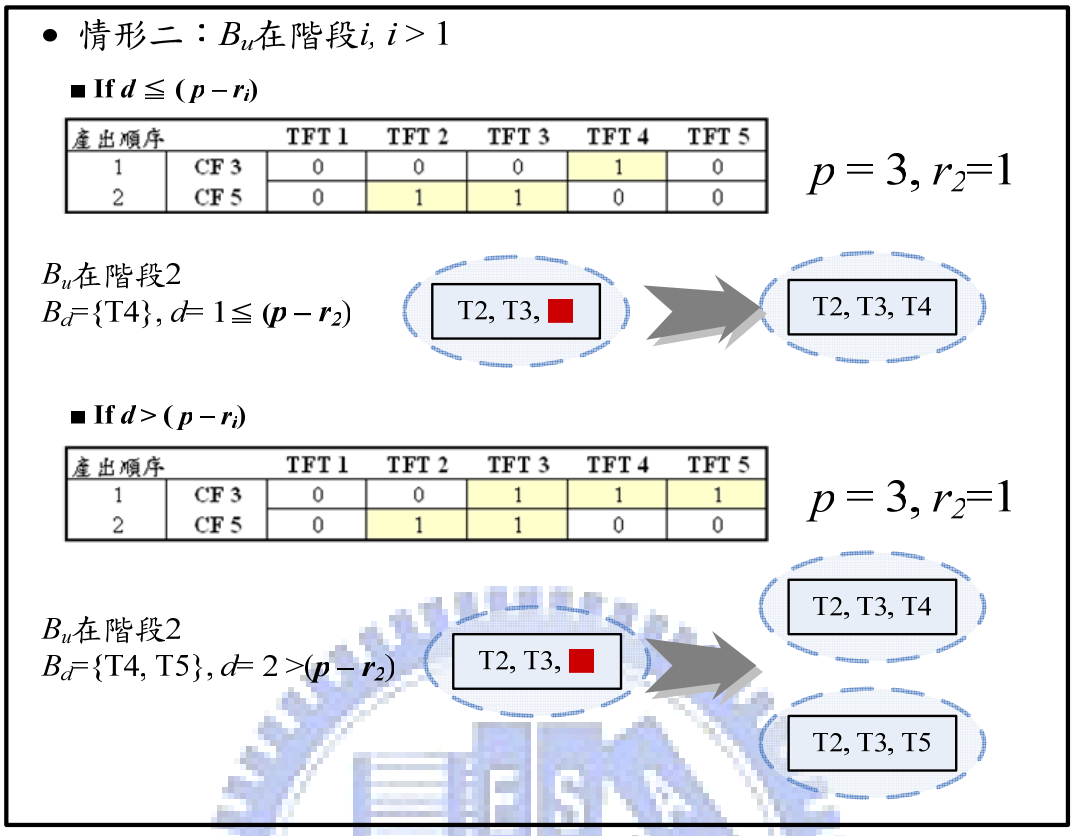


圖 4.9 組態三之情形二狀態重組例圖

組態二的狀態是由兩個以上的批次組成，其中最後一個批次為不完整批次，為方便以下說明，吾人將最後一個不完整的批次定義為 B_u, B_u 包含了 r_i 個卡匣及 $(p - r_i)$個空缺；State Deployment 模組的目的是，將 B_u 重組成為完整的批次 B_r，結果會導致單一組態二的狀態重組成為多個組態一的狀態。

若要對組態二的狀態進行重組，State Deployment 模組的輸入資料為同狀態中 B_u 前一個完整批次 B_g，B_g 包含 p 個卡匣，State Deployment 模組的演算構想是從 B_g 中隨機挑選$(p - r_i)$個卡匣，以填滿 B_u 中的空缺，以形成新的批次 B_r, 進行重組後，狀態數目會變成原來的 $s_d(i)$倍。演算法內容說明如下：

Procedure *State_Deployment_for Case2*(B_u, B_p, $s_d(i)$, B_r)

B_u= { u_i| 1≦ i≦ r_i}　　/*B_u 是由 r_i 個卡匣形成的集合*/

B_g= { g_j| 1≦ j≦ p}　　/*B_g 是由 p 個卡匣形成的集合*/

Random select $(p-r_i)$ cassettes from B_d to fill the vacant position in B_u to form B_r;

Number of states deployed $s_d(i)=C^{p}_{p-r_i}$;

透過 State Deployment 模組的演算，原本屬於組態二的狀態，經過以上的程序進行重組，會重組成 $s_d(i)$個新狀態，且新狀態均由完整的批次所組成，如圖 4.9 所示，階段 3 中原有 3 個組態二的狀態，每個狀態均由一個完整批次及一個不完整批次所組成，經上述程序進行重組後，單一狀態可重組成為 $C^{p}_{p-r_3}=C^{2}_{2-1}=2$ 個狀態，所以原本的 3 個狀態經重組後，變成 6 個新的狀態，且每個狀態均由兩個完整的批次所組成。

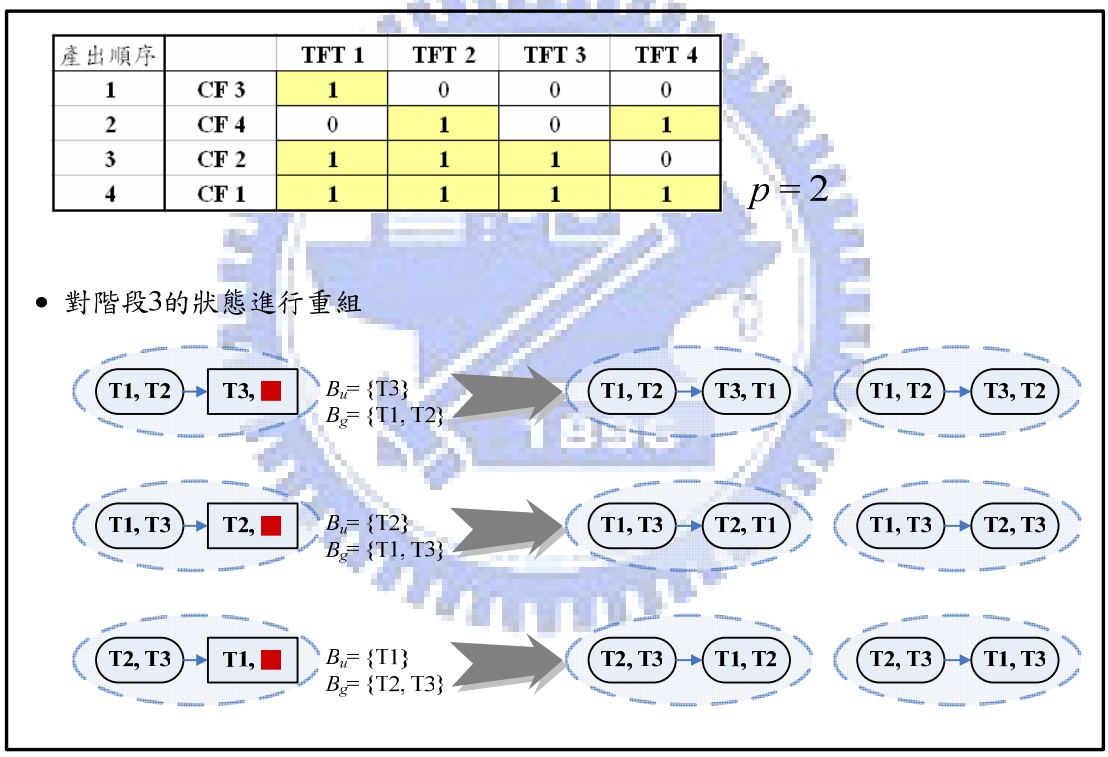


圖 4.10 組態二狀態重組例圖

在經過 Stage Modeling, State Modeling 和 Batch Modeling 三模組後，吾人可得到基本網路架構，如圖 4.3 所示，之後再透 State Deployment 模組進行演算後，目的是在使原本在網路中不完整的批次重組成為一個或多個完整的批次，意即原本屬於組態二和組態三的狀態均盡可能重組為組態一的狀態，且狀態個數會由原

本的 1 個轉變成為 $s_d(i)$個，如下圖 4.10 所示，原本在階段 1 中的狀態並不進行重組，故維持原狀態組態，但原本在階段 3 中的 3 個組態二的狀態，透過 State Deployment 模組的重組後，轉變成為 6 個組態一的狀態，形成一個完整的網路架構。本研究依序經過 Stage Modeling, State Modeling, Batch Modeling 和 State Deployment 的過程，可得到完整的網路架構，以便後續動態規劃法求解。

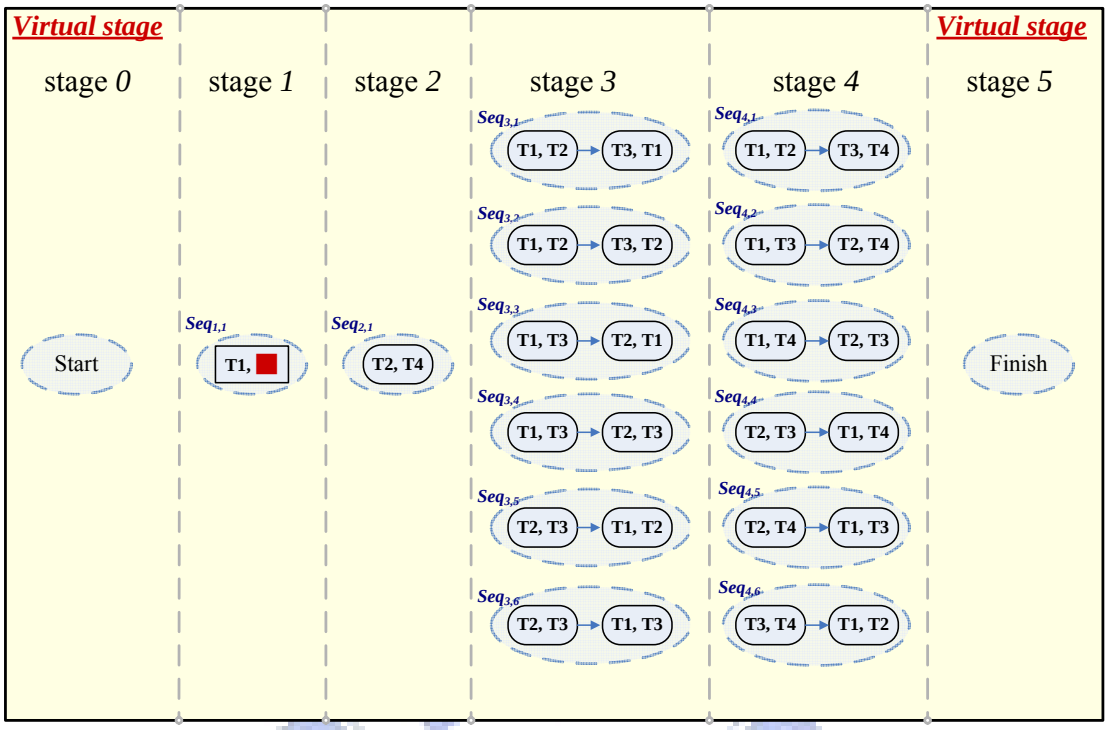


圖 4.11 完整的基本網路架構

4.2 動態規劃演算法

■ 符號定義

時間參數定義：

T_u: 將會員卡匣上載至輸入埠所需要的時間

T_d: 將會員卡匣從輸入埠卸載所需要的時間

T_p: 排序機抽取一塊基板所需要的時間

T_{Limit}: 下游組立壓合機台的作業週期時間

時間變數定義：

$S\_Cost(seq_{i-1,k}, seq_{i,j})$: 排序機以投入順序$(seq_{i-1,k}, seq_{i,j})$組合完成目標卡匣 K_i 所花費的時間

$TS\_Cost_{i,j}$: 排序機使用投入順序 $seq_{i,j}$, 完成目標卡匣 K_i 的時點

$W\_Cost_{i,j}$: 投入順序 $seq_{i,j}$ 在排序機上作業的時間成本

$I\_Cost(seq_{i-1,k}, seq_{i,j})$: 兩投入順序組合$(seq_{i-1,k}, seq_{i,j})$交替間所發生的時間成本

$DStream\_Cost_{i,j}$: 排序機利用投入順序 $seq_{i,j}$, 下游機台完成目標卡匣 K_i 的時點

$Delay(seq_{i-1,k}, seq_{i,j})$: 下游組立壓合機台等待目標卡匣 K_i 所發生的閒置時間

$TDelay_{i,j}$: 下游組立壓合機台至完成目標卡匣 K_i 所累積的閒置時間

將投入順序決策模組化成完整的網路架構後，本研究便利用動態規劃演算法進行求解，主要的重點在計算分枝成本(branch cost)，再求解備具最小總成本的投入順序決策組合，該投入順序決策組合便為本研究問題之最佳解。

在本研究問題，分枝成本所指的是選用階段 i-1 的投入順序(狀態)$seq_{i-1,k}$, 再選用階段 i 的投入順序(狀態)$seq_{i,j}$, 在這樣的兩個投入順序組合下所完成目標卡匣 K_i 產出，至下游壓合機台作業前所發生的閒置時間(Delay Time)，相較於典型的動態規劃問題－馬車問題，分枝成本代表的是選擇從地區 i-1 的城市 K 遷徙至地區 i 的城市 J，城市 K 和城市 J 之間的路徑長度。

若要計算本研究問題的分枝成本，則必須得知排序機產出單一目標卡匣所使用的作業時間 $S\_Cost(seq_{i-1,k}, seq_{i,j})$(簡稱作業成本)，該作業時間包含了兩個時間成本，分別為單一投入順序在排序機上作業的時間成本 $W\_Cost_{i,j}$ (Within Cost, 簡稱內部成本)、兩投入順序交替間所發生的時間成本(Interface Cost, 簡稱介面成本)，內部成本為狀態內的各批次上下排序機所發生的時間成本，而介面成本則為兩投入順序間批次轉換所發生的時間成本，如下圖 4.11 所示，若選用階段 i-1 中的投入順序 $seq_{i-1,k}$, 再選用階段 i 的投入順序 $seq_{i,j}$, 目標卡匣 K_i 的作業成本為內部成本加上介面成本，內部成本為由 $B_{(i,j),1}$ 開始，至 $B_{(i,j),b(i)}$結束，其中 $b(i)$個

批次上下排序機所花費的時間成本，而介面成本則為 $B_{(i-1,\,k),\,b(i-1)}$及 $B_{(i,\,j),\,1}$ 兩批次上下排序機所發生的時間成本。以下便對內部成本計算模組及介面成本計算模組分別進行描述。

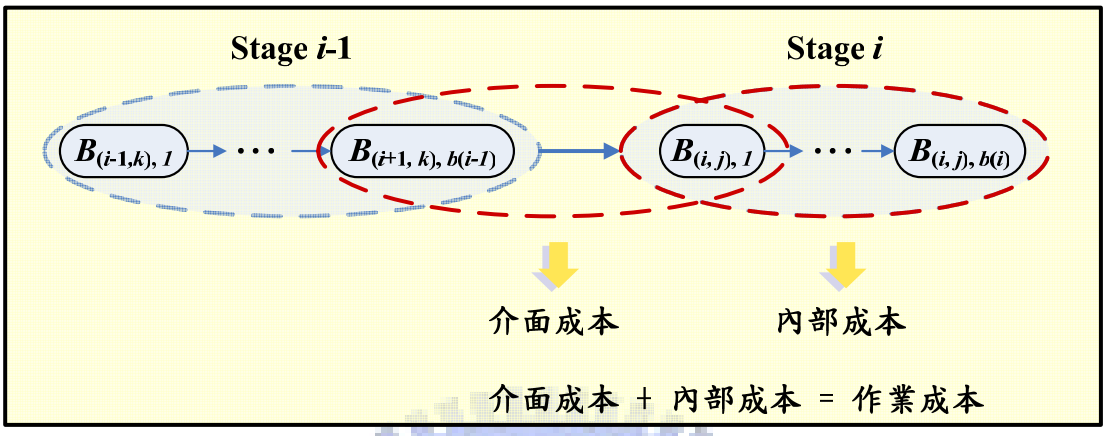


圖 4.12 作業成本示意圖

4.2.1 內部成本計算模組

承上，內部成本為狀態內的各批次上下排序機所發生的時間成本，因此內部成本計算模組以狀態內的批次 $seq_{i,j} = B_{(i,j),1} \rightarrow B_{(i,j),2} \rightarrow \ldots \rightarrow B_{(i,j),b(i)}$為輸入資料進行運算，而根據狀態內批次的個數 $b(i)$的不同可分為兩種情形討論：

情形一：若 $b(i)= 1$, 則內部成本只有抽取 n 個基板的時間 $n^* T_p$；

情形二：若 $b(i)> 1$, 則利用內部時間計算程序(Procedure *Compute_Within_Cost*)進行運算；

Procedure *Compute_Within_Cost*($seq_{i,j}$, $W\_Cost_{i,j}$)

Set $seq_{i,j} = B_1 \rightarrow B_2 \rightarrow \ldots \rightarrow B_{b(i)}$

$N\,(B_j \cap B_{j+1}) = z_j$

/*z_j: B_j和 B_{j+1} 兩批次轉換過程中，不需要從排序機上卸載的卡匣個數*/

$$W\_Cost_{i,j} = n^* T_p + \sum_{j=1}^{b(i)-1} (T_u + T_d) \cdot (p - z_j)$$

4.2.2 介面成本計算模組

根據定義，介面成本為兩投入順序間批次轉換所發生的時間成本，因此介面成本計算模組以 $B_{(i-1,\,k),\,b(i-1)}$, $1 \leqq k \leqq s(i\text{-}1)$, 和 $B_{(i,\,j),\,1}$, $1 \leqq j \leqq s(i)$為輸入資料進行運算，為方便說明吾人將 $B_{(i-1,\,k),\,b(i-1)}$ 定義為 B_s, 而 $B_{(i,\,j),\,1}$ 定義為 B_e, 在介面成本計算模組中，以因為 B_s 和 B_e 兩批次組態的不同，而有四種不同的情形進行討論：

情形一：B_s 為完整批次& B_e 為完整批次，則使用介面時間計算程序一(Procedure *Compute_Interface_Cost_1*)

情形二：B_s 為完整批次& B_e 為不完整批次，此為不合理情形；

情形三：B_s 為不完整批次& B_e 為完整批次，則使用介面時間計算程序二(Procedure *Compute_Interface_Cost_2*)

情形四：B_s 為不完整批次& B_e 為不完整批次，則使用介面時間計算程序二(Procedure *Compute_Interface_Cost_2*)

根據上述情形討論，情形一適用介面時間計算程序一進行運算，而情形三及四適用介面時間計算程序二進行運算，此外情形二在本研究問題中為不合理情形，原因是在網路建構的過程中，不會有這樣的批次組態發生，故不加以討論。以下便針對介面時間計算程序一及程序二進行介紹：

Procedure *Compute_Interface_Cost_1*(B_s, B_e, *I_Cost*($seq_{i-1,\,k}$, $seq_{i,j}$))

$N\,(B_s \cap B_e) = h_0$

/*h_0: B_s 和 B_e 兩批次轉換過程中，不需要從排序機上卸載的卡匣個數*/

I_Cost($seq_{i-1,\,k}$, $seq_{i,j}$)= $(T_u + T_d) * (p - h_0)$

Procedure *Compute_Interface_Cost_2*(B_s, B_e, *I_Cost*($seq_{i-1,\,k}$, $seq_{i,j}$))

$N\,(B_s \cap B_e)$ + N (vacant position in B_e)= h_0

/*h_0: B_s 和 B_e 兩批次轉換過程中，不需要從排序機上卸載的卡匣個數*/

N (vacant position in B_e) – N (vacant position in B_s) = h_1

/*h_1: B_s 和 B_e 兩批次轉換過程中，僅需要上載至排序機的卡匣個數*/

$I\_Cost(seq_{i-1,k}, seq_{i,j}) = (T_u + T_d) * (p - h_0 - h_1) + T_u * h_1$

4.1.3 計算閒置時間模組

經由上述內部成本計算模組和介面成本計算模組的計算後，可得到作業成本 $S\_Cost(seq_{i-1,k}, seq_{i,j})$，為了計算下游機台等待目標卡匣 K_i 所發生的閒置時間 $Delay(seq_{i-1,k}, seq_{i,j})$，吾人必須先利用一等式來描述排序機產出目標卡匣 K_i 的時點 $TS\_Cost_{i,j} = TS\_Cost_{i-1,k} + S\_Cost(seq_{i-1,k}, seq_{i,j})$, 另外再利用另一等式來描述下游組立壓合機台完成目標卡匣 K_i 的時點 $DStream\_Cost_{i,j} = DStream\_Cost_{i-1,k} + T_{Limit} + Delay(seq_{i-1,k}, seq_{i,j})$, 如此可得到下游機台等待目標卡匣 K_i 所發生的閒置時間 $Delay(seq_{i-1,k}, seq_{i,j}) = \{TS\_Cost_{i,j} - DStream\_Cost_{i-1,k}, 0\}^+$，經由以上的運算，可得到閒置時間 $Delay(seq_{i-1,k}, seq_{i,j})$，閒置時間為本動態規劃模型中的分枝成本。

4.1.4 動態規劃求解

本動態規劃求解方法和典型馬車問題求解最短總路徑(shortest path)過程類似，在本研究之動態規劃模型中，分枝成本為下游機台等待目標卡匣 K_i 閒置時間 $Delay(seq_{i-1,k}, seq_{i,j})$，欲求解從排序機開始運作至完成 N 個目標卡匣，具最小累積閒置時間的投入順序組合，相較於馬車問題，其分枝成本為城市之間的路徑長度，欲求解從旅途開始至結束，具最小總路徑長度的城市組合。

本研究在在每階段中，均透過以下決策函式進行求解：

$$TDelay^*_{i,j} = \min\{TDelay^*_{i-1,k} + Delay(seq_{i-1,k}, seq_{i,j})\}, 1 \le j \le s(i)$$

從起始階段 1 至終止階段 N，經由上述的決策函式進行運算，最終可得到 $TDelay_{i,j}^*$, $i = N$，此時即可得到下游組立機台利用率為 $Utilization = 1 - \frac{TDelay^*_{N,j}}{DStream\_Cost_{N,j}}$ 。

第五章　實例驗證

本章以 90 個工廠實際的案例來說明本研究所發展的方法及成果。此驗證所使用的電腦硬體是 AMD Athlon (TM) SP 2200+ 1.5GHz 1GB DDRAM，並以 Microsoft Visual C++撰寫演算法的程式。

5.1 資料輸入與參數設定

本實例驗證採用張東華[4]基板配對決策所得到的結果，TFT 及 CF 卡匣總數以 N 表示，每個卡匣的基板個數是 n，每個基板的面板個數以 *panel* 表示。CF 基板的平均良率是 90%，TFT 基板的平均良率是 85%，兩者都是二項式分配 (Binomial Distribution)。

排序機包含一個機器手臂、p 個輸入埠、一個輸出埠；本案例討論五個情境分別為 $p = 1, 2, 3, 4, 5$。排序機下游組立作業每完成一組卡匣的時間為 T 分鐘，TFT 卡匣上載或卸下排序機的時間定義相對於下游組立作業的時間，在此考慮每卸下或裝上一個卡匣須耗時 $0.05T$。本模擬比較的基準為目前業界使用的隨機完成順序及王君豪[5]提出的基因啟發式解法。

本案例在驗證執行程序上，是先採用王君豪[5]所發展的基因啟發式解法求解 15 次，可得到 15 個較佳的產出順序，以這 15 個目標卡匣組產出順序為輸入資料，再以本研究方法求解投入順序決策問題，15 個解中績效最高的者則為本演算法之最佳解。

5.2 結果分析

下表 5.1 是單位裝卸卡匣時間為 $0.05T$ 的模擬結果。此參數的設計乃根據實際生產的情形而訂定，不同規模及代數(Generation)的 LCD 廠，往往有不同裝卸卡匣時間對組立時間的比，但比值大多為 0.05。

表 5.1　單位裝卸卡匣時間為 0.05T 的模擬結果

生產情境	輸入埠數 p	隨機解	王氏	本研究	改進績效
N=10 *n*=20 *panel*=6	1	89.9%	96.2%	100.0%	3.8%
	2	91.2%	95.9%	100.0%	4.1%
	3	90.0%	95.5%	100.0%	4.5%
	4	89.7%	95.1%	100.0%	4.9%
	5	88.7%	94.5%	100.0%	5.5%
N=10 *n*=20 *panel*=12	1	90.9%	94.5%	100.0%	5.5%
	2	91.2%	95.0%	100.0%	5.0%
	3	89.7%	94.6%	100.0%	5.4%
	4	89.5%	94.1%	100.0%	5.9%
	5	90.1%	93.6%	100.0%	6.4%
N=10 *n*=20 *panel*=30	1	88.5%	92.8%	100.0%	7.2%
	2	89.1%	92.6%	100.0%	7.4%
	3	88.6%	92.7%	100.0%	7.3%
	4	86.3%	92.7%	100.0%	7.3%
	5	85.6%	91.7%	100.0%	8.3%
N=15 *n*=20 *panel*=6	1	93.4%	98.3%	100.0%	1.7%
	2	92.6%	98.2%	100.0%	1.8%
	3	93.6%	98.1%	100.0%	1.9%
	4	94.0%	98.0%	100.0%	2.0%
	5	92.6%	97.9%	100.0%	2.1%
N=15 *n*=20 *panel*=12	1	92.6%	95.1%	100.0%	4.9%
	2	92.5%	95.3%	100.0%	4.7%
	3	92.7%	95.5%	100.0%	4.5%
	4	93.1%	95.8%	100.0%	4.2%
	5	92.9%	95.5%	100.0%	4.5%
N=15 *n*=20 *panel*=30	1	93.1%	94.4%	100.0%	5.6%
	2	92.5%	94.2%	100.0%	5.8%
	3	91.6%	94.2%	100.0%	5.8%
	4	90.9%	94.3%	100.0%	5.7%
	5	91.4%	94.4%	100.0%	5.6%
N=20 *n*=20 *panel*=6	1	94.9%	98.5%	100.0%	1.5%
	2	95.1%	98.5%	100.0%	1.5%
	3	95.7%	98.4%	100.0%	1.6%

	4	95.5%	98.3%	100.0%	1.7%
	5	96.2%	98.3%	100.0%	1.7%
N=20 *n*=20 *panel*=12	1	94.8%	97.9%	100.0%	2.1%
	2	94.6%	98.2%	100.0%	1.8%
	3	95.1%	98.1%	100.0%	1.9%
	4	94.7%	98.0%	100.0%	2.0%
	5	94.7%	97.9%	100.0%	2.1%
N=20 *n*=20 *panel*=30	1	92.8%	95.2%	95.2%	0.0%
	2	93.1%	96.2%	98.2%	2.0%
	3	93.1%	96.9%	99.1%	2.2%
	4	93.3%	97.7%	99.5%	1.8%
	5	93.5%	97.6%	100.0%	2.4%

由驗證的結果可得知，本研究方法在現今 TFT-LCD 生產情境中具有十分顯著的效益，壁排序機裝卸卡匣的時間和下游組立機台作業時間比例為 0.05 的情形下，本研究方法所得到的績效比王氏利用基因啟發式解法有更佳的表現，改進的績效如表 5.1 所示，最大有 8.3%的改進幅度。

進行實例驗証的過程中，可得知程式的執行運算時間和資料規模的大小有很大的關係，本研究以動態規劃法進行解題，若資料規模越大者，其建構出的網路模型便越大，意即需要求解的投入順序組合數便愈多，根據本研究的案例資料，在各種不同的情境下，以本研究方法建構出的網路中路徑數目如下表 5.2 所示，舉例最大的網路說明，路徑總數為 $1.47*10^{138}$ 條，意即網路包含有 $1.47*10^{138}$ 個不同的投入順序組合，表 5.2 中亦列出在不同情境下，程式運算所需時間，以秒為計算單位。

表 5.2　動態規劃網路之路徑數目

生產情境	輸入埠數 *p*	路徑數目	運算時間(秒)	大小順序
N=10 *n*=20 *panel*=6	1	$1.44*10^{17}$	11.8	8
	2	$1.32*10^{13}$	10.2	6
	3	$2.17*10^{11}$	9.4	4

	4	$9.07*10^{9}$	8.6	3
	5	$9.45*10^{5}$	7	1
N=10 *n*=20 *panel*=12	1	$3.64*10^{20}$	13	12
	2	$4.00*10^{17}$	11.8	9
	3	$4.11*10^{14}$	10.6	7
	4	$1.13*10^{12}$	9.8	5
	5	$3.30*10^{8}$	8.2	2
N=10 *n*=20 *panel*=30	1	$2.81*10^{37}$	19.8	25
	2	$5.23*10^{28}$	16.2	19
	3	$4.25*10^{25}$	15	16
	4	$1.19*10^{18}$	12.2	10
	5	$7.29*10^{19}$	12.6	11
N=15 *n*=20 *panel*=6	1	$4.87*10^{41}$	21.4	30
	2	$2.27*10^{32}$	17.8	23
	3	$3.79*10^{26}$	15.4	17
	4	$9.88*10^{23}$	14.2	15
	5	$4.94*10^{20}$	13	13
N=15 *n*=20 *panel*=12	1	$3.49*10^{53}$	26.2	36
	2	$5.08*10^{42}$	21.8	31
	3	$4.72*10^{35}$	19	24
	4	$4.21*10^{29}$	16.6	20
	5	$8.54*10^{27}$	15.8	18
N=15 *n*=20 *panel*=30	1	$1.97*10^{74}$	33.6	40
	2	$4.38*10^{58}$	28.2	37
	3	$1.59*10^{50}$	25	34
	4	$1.10*10^{40}$	21	29
	5	$8.98*10^{38}$	20.2	27
N=20 *n*=20 *panel*=6	1	$3.90*10^{49}$	24.6	33
	2	$1.70*10^{39}$	20.6	28
	3	$1.95*10^{30}$	17	21
	4	$6.59*10^{30}$	17.25	22
	5	$4.47*10^{22}$	13.8	14
N=20 *n*=20 *panel*=12	1	$5.43*10^{74}$	34.6	41
	2	$9.04*10^{58}$	28.2	38
	3	$1.33*10^{52}$	25.8	35
	4	$1.65*10^{38}$	20.2	26
	5	$2.47*10^{43}$	22.2	32

N=20 *n*=20 *panel*=30	1	$1.47*10^{138}$	60.2	45
	2	$9.52*10^{110}$	49	44
	3	$1.23*10^{98}$	44.2	43
	4	$9.33*10^{84}$	38.6	42
	5	$2.44*10^{65}$	31	39

根據上表 5.2 所得資料，以路徑大小等級 1 至 45 為 X 軸，程式運算時間為 Y 軸繪出圖 5.1 的分布圖形，可明顯看出路徑數目的增加，會對程式的運算造成很大的負荷。

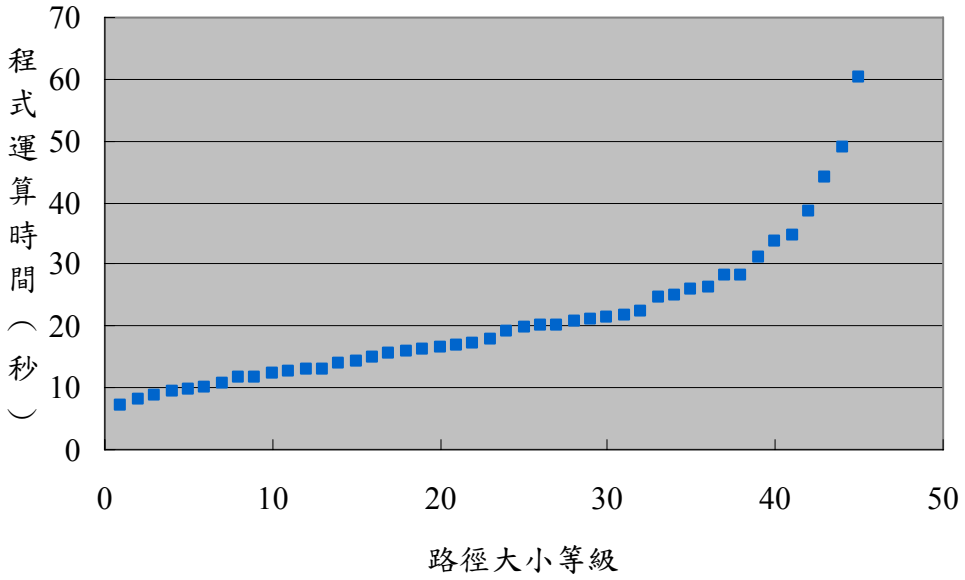


圖 5.1 分布圖

第六章　結論與未來研究方向

6.1 結論

本研究著焦於 TFT-LCD 組立製程中排序機派工決策問題，在給定目標卡匣產出順序的情形下，吾人以動態規劃模型來描述投入順序決策問題，並使用動態規劃系統化的解法進行尋優，績效衡量是為了最大化下游組立壓合機台的利用率。在經過案例驗證後，實驗結果顯示本研究方法就目前業界的生產情境來說，本研究方法顯著地比過去研究所提出的基因啟發式演法及以目前業界使用的方法有更高的效率。

6.2 未來研究方向

在本研究方法中，將組成成員卡匣上下排序機的方式是以輸入埠的個數 p 為批次單位，但在生產現場，組成成員可以一次一個卡匣的方式上下排序機；因此可以考慮以模擬來進行解題，會更加符合現場的情形。

參考文獻

[1] 李秀玉 (1999) 應用賽局理論分析我國薄膜電晶體液晶顯示器產業之競爭策略，交通大學科技管理研究所碩士論文

[2] 楊毅臻 (2003) 以基板配對為基礎的 LCD 廠排序系統規劃之研究，華梵大學工業管理研究所碩士論文

[3] 楊佳翰 (2004) 應用改良型巨集式啟發式方法於TFT-LCD良率控制最佳化之研究，交通大學工業工程與管理研究所碩士論文

[4] 張東華 (2005) TFT-LCD 組立製程配對作業之研究，交通大學工業工程與管理研究所碩士論文

[5] 王君豪 (2005) TFT-LCD 排序機派工之研究，交通大學工業工程與管理研究所碩士論文

[6] Jeong, B., Kim, S. W., and Lee, Y. J., An assembly scheduler for TFT LCD manufacturing, *Computers & Industrial Engineering*, 2004, 41(1), 37-58.

[7] Kuhn, H. W., The Hungarian method for the assignment game, *Naval Research Logistics Quarterly*, 1955, 2, 83-97.

[8] Reevest C. R., A genetic algorithm for flowshop sequencing, *Computers Operations Research*, 1995, 22(1), 5-13.

[9] Shin, H. J., and Leon, V. J., Scheduling with product family set-up time: an application in TFT LCD manufacturing, *International Journal of Production Research*, 2004, 42(20), 4235-4248.

[10] Toshihisa, T., TFT/LCD: liquid-crystal displays addressed by thin-film transistors, 1996 (Gordon and Breach Publishers: Australia).

[11] Toba H., A tight flow control for job-shop fabrication lines with finite buffer, *IEEE Transactions on Automation Science and Engineering*, 2005, 2(1), 78-83.