國立交通大學

電機學院與資訊學院 資訊學程

碩士論文

基因演算法用於增加產能的研究-以 TFT-LCD 廠 Array 製程 RGV 搬送為例

Using GA to Increase Throughput for RGV Based TFT-LCD Array Fab

研 究 生：紀碧如

指導教授：曾憲雄 教授

中 華 民 國 九 十 五 年 六 月

基因演算法用於增加產能的研究-以 TFT-LCD 廠 Array 製程 RGV 搬送為例

Using GA to Increase Throughput for RGV Based TFT-LCD Array Fab

研 究 生：紀碧如　　Student：Pi-Ju Chi

指導教授：曾憲雄教授　　Advisor：Dr. Shian-Shyong Tseng

國 立 交 通 大 學

電機學院與資訊學院專班 資訊學程

碩 士 論 文

A Thesis
Submitted to Degree Program of Electrical Engineering and Computer Science
College of Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Computer Science
June 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年六月

基因演算法用於增加產能的研究
-以 TFT-LCD 廠 Array 製程 RGV 搬送爲例

研究生：紀碧如　　指導教授：曾憲雄 博士

國立交通大學電機學院與資訊學院專班

摘要

TFT-LCD 廠主要分爲三大製程：陣列(Array)、組立(Cell)、模組(Module)，其中陣列製程由於機台昂貴，最重視的是機台使用率及產能。本研究是針對陣列製程的工廠，此工廠會因爲不同機台製程特性的不同，將工廠劃分成爲數個區域，再針對區域配置適合的機台設備，並且每個區域內是由 RGV(Rail Guided Vehicle)負責搬送。此工廠的特性是每個區域同一時間會有許多的搬送命令，而 RGV 一次執行一個搬送命令來完成區域內的搬送工作。因此，藉由調整區域中 RGV 搬送的順序，就可以影響區域整體的生產時間，進而提昇區域的產能。

而目前相關研究，一般而言皆只是提出一、兩個影響搬送效能的因素，找出改善小部分狀況的方法，如此未能考慮其他大部分的狀況，也沒有探討其他因素的影響。另外，這些研究皆是針對在 AGV(Automated Guided Vehicle)搬送下，因此結果是針對 AGV 的特性，無法應用到其他特性的工廠中。

因此，本研究希望針對陣列製程工廠的單一特定區域，導入一適用於 RGV 特性的搬送策略來選擇搬送命令，進而提昇區域的產能。首先建立一個盡量符合現實工廠狀況的模擬器，並提出相關的假設，如每次離開此特定區域後再回到區域的時間。製作此模擬器主要是希望利用模擬器計算此特定區域的總生產時間，利用生產時間來評估區域產能的大小，時間愈短表區域產能越高。接下來提出搬送策略爲命令屬性的線性組合。由於屬性之間的組合複雜且互相牽制，故利用基因演算法找出較適合的權重，得到一個針

對此特定區域較適合的搬送策略，如此一來就可以藉由此搬送策略提昇區域產能。經由實驗的結果，不同的搬送策略對於同一個區域生產時間有顯著的影響，針對特定的區域可找到較適合的搬送策略，可以有效減少區域生產時間提昇區域產能。

關鍵字：搬送系統、TFT-LCD、基因演算法、陣列製程、產能、RGV、搬送策略

Using GA to Increase Throughput for RGV Based TFT-LCD Array Fab

Student: Pi-Ju Chi **Advisor: Dr. Shian-Shyong Tseng**

Degree Program of Electrical Engineering and Computer Science

National Chiao Tung University

Abstract

As we know, TFT-LCD factory contains three major process：Array, Cell, and Module. Because the manufacturing equipments in Array are more expensive than others, we are very concerned with the throughput and the utilization rate of the equipments. TFT-LCD Array factory can be divided into several areas according to different process features of equipments, and then we will configure equipments and RGV (Rail Guided Vehicle) for transporting Lot between equipments for each area. The issue of factory is how can we choose one command from several transport commands in the area to improve transport efficiency since RGV only handles one command at a time. Therefore, the ordering of the commands can affect total manufacturing time and increase throughput in an area.

In related researches, which are based on AGV transport and can only be applied for AGV behavior, only one or two factors influencing transport efficiency have been proposed.

Hence, we hope to find an appropriate transport strategy to reorder transport commands for RGV in an Area of TFT-LCD Array Fab to increase throughput. First, we set up a simulator that is similar to real factory with some assumptions such as the time between Lot leaving the area and coming back to the area should be constant. Since simulator can calculate the manufacturing time of all Lots in the area, we thus evaluate throughput using the total

manufacturing time where the less total manufacturing time indicates the throughput is high. Then we propose a transport strategy which is the linear combination of commands' attributes, and then use GA to find more suitable weight and get one transport strategy in a specific area. Applying the obtained transport strategy to the specific area will increase throughput in this area. The experimental results show that difference transport strategy will affect manufacturing time obviously, and that specific transport strategy for specific area will reduce manufacturing time effectively and increase throughput in this area.

Keywords: AMHS, RGV, TFT-LCD, Array, Throughput, Transport Strategy

誌謝

此篇論文能在兩年內順利的完成，首先得感謝指導教授 曾憲雄教授，不厭其煩的指正及建議，讓我的論文逐漸成型進而完成，也讓我從研究論文的同時，得到更多的新知。此外也感謝口試委員們百忙之中，給予我許多的論文的建議，讓我受益良多。

同時也感謝實驗室的學長們威州、慶堯、及衍旭學長們，在我論文研究中，給了許多的建議，讓我對於我的研究有更深的體會，也從中學到不少相關的知識。也感謝實驗室其它雖未直接指導我的學長姐們，謝謝你們的鼓勵與幫忙。感謝一路走來的實驗室的同學們，在我喪失信心時，不斷的給我鼓勵及安慰，大家也在互相勉力下一起完成論文順利畢業。

此外也要感謝中華映管的主管及同事們，在論文的研究上給了很大的幫忙，讓我更加了解工廠內部的運作及問題發生的原因，才能順利的完成此篇論文。也謝謝班上的同學，不斷的給我鼓勵及相互的討論的機會。

最後也要謝謝我的家人及朋友，在這兩年中不斷的支持和鼓勵，讓我能在兩年內順利的完成此篇論文。除此之外，也謝謝那些曾經給我過建議和鼓勵的好朋友們，能順利完成此篇論文，都是靠老師及學長們不斷的指導及大家的幫忙與鼓勵，謝謝大家。

目錄

圖目錄

表目錄

符號說明

s ：投入後至第一次進入選定區域的時間

P_i ：Lot 第 i 次離開選定區域

T_i ：Lot 第 i 次離開區域後，到第 i+1 次回到區域

n ：Lot 投入數

v ：每個 Lot 需進入選定區域的次數

Rt ：Lot 進到選定區域到離開區域的總花費時間

Rt_j^i ：第 i 個 Lot 在第 j 次進入區域花費時間

RPt ：區域中，製程機台製造所花費的時間

RMt ：區域中，在機台間流動所花費的時間

RWt ：區域中，由於機台數量不足而造成的等待機台的時間

$f_{w1,w2,w3,w4,w5}$ ：搬送策略涵數

Att_{From} ：搬送命令中，來源機台的狀態

Att_{To} ：搬送命令中，目地機台的狀態

Att_{RGV} ：RGV 與搬送命令來源機台的距離

Att_{LOT} ：搬送命令的等待時間

Att_{STK} ：搬送命令等待 Stocker 的時間

$\sum_{i=1}^{n}\sum_{j=1}^{v} Rt_j^i$ ：區域總計生產時間

一、緒論

生產製造廠中尤其重視工廠的產能，產能的高低直接影響成本的回收，產能越大代表著成本越低，因此希望找出提昇產能的方法。本研究針對TFT-LCD Fab[1]中Array製程進行增加產能研究，並提出提昇產能的方法。欲提昇產能，首先必須瞭解TFT-LCD生產流程及Array製程的特性，然後才能在既有的流程找到提昇產能的方法。故在 1.1 節中首先介紹TFT-LCD Array製造流程，然後在 1.2 節中根據TFT-LCD 的生產特性，提出研究的動機及目標，接著 1.3 節中提出達到目標的方法及成果。由於TFT-LCD Fab運作複雜，範圍廣泛，因此在 1.4 節中提出研究的範圍及限制，最後在 1.5 節中列出本論文的架構。

1.1 TFT-LCD Array 製造流程概念

薄膜液晶顯示器(Thin Film Transistor Liquid Crystal Display；TFT-LCD) 主要是利用玻璃當基板，在玻璃上加工而成，主要是利用薄膜電晶體 (TFT)當做開關，控制液晶顯示器 (LCD)上每一個畫素 (Pixel)的明暗，以達到顯示色彩的效果。

薄膜液晶顯示器製造主要分爲三大製程：陣列製程(Array)、面板組立(Cell)、模組組裝(Module)。在陣列製程中主要是生成 TFT 的電路，藉由成膜(Depoe)、曝光(Photo)、蝕刻(Etch)、剝膜(Strip)等步驟製造。其機台特性爲單一機台處理單一事項，例如曝光機只能處理曝光需求，一般都以非串聯的的方式擺置，故在更換生產機台時需大量借助搬送系統來完成。而面板組立製程主要是將陣列製成生產的 TFT 面板與彩色濾光片(Color Filter)結合灌入液晶，並切割爲生產面板大小，其機台的配置一般以串聯方式存在，類似於生產線的模式，故較少使用搬送設備。而模組主要針對前製程的結果再加工，加入驅動 IC 組裝後出貨。其中陣列製程可說是 LCD 生產的根本，其對工廠整體的產能影響最深，且由於機台也較昂貴，故工廠特別重視陣列製程的產能。再者，在陣列製程中，除生產機台外也大量使用搬送系統，故除機台生產時間外，搬送的時間對產能也有所影

[1] Fab: fabrication的縮寫,表製造生產廠。

響。

針對陣列製程而言，TFT-LCD 在生產時，是由事先定義好的製造流程(Route)進行，其製程的順序是固定不可重新排列的，並且可能多次進入同一種類的機台，例如需先成膜再曝光，則不可改爲曝光後再成膜，並且在製程中爲達到較好的製造及搬送效益，所以製造及搬送的計算以 Lot 爲單位，而每個 Lot 則包含 20 片的玻璃基板，非以單片玻離計算。由於各種生產機台的製程特性不同，工廠一般會劃分成爲數個區域，如下圖 1。區域中配置適合的機台設備，如：多種不同種的生產機台(Equipment)數台、倉儲(Stocker)、搬送裝置，其中搬送裝置包含搬送車 RGV(Rail Guided Vehicle)及其運行所需的軌道，如圖 2。

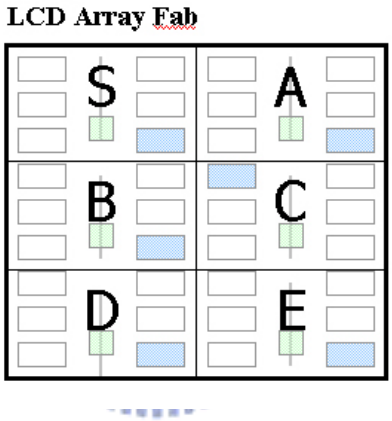

圖 1：TFT-LCD Array Fab 區域示意圖

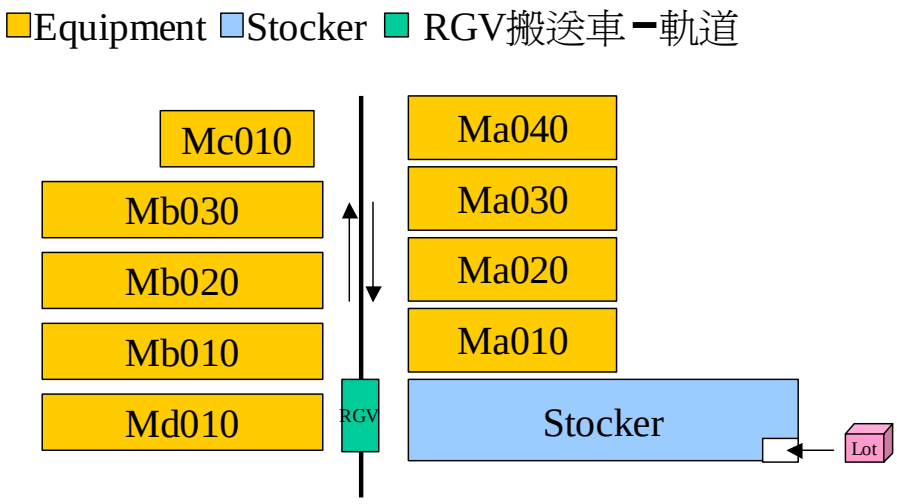

圖 2：A 區域內部詳細組成配置圖

由於 LCD 的製造是經由循序且固定的製造流程生產，這些生產機台分布於工廠的個區域中，可能多次經過同一區域，且每次進入區域後，其所需經過區域內部機台種類及順序不相同，例如:假設 LCD 製造步驟所需經過的區域如下:

Start→S(1)→A(1) →B(1) →C(1) →D(1) →A(2) →B(2) →E(1) →A(3) →B(3) →C(2) →…A(4)…A(5)…→End 如針對 A 區域討論，由上述的製造流程可知，從投入後總共需進入 A 區域 5 次，才可完成 LCD 在此製程的製造，進而分析每次進入此區域，所需經過的機台種類，其經過的機台種類如下表 1：

表 1：Lot 每次進入區域的詳細製造流程

進入次數	經過的機台種類與次序
A(1)、A(3)	Ma→Mb
A(2) 、A(5)	Ma→Mb→Mc→Md
A(4)	Ma→Md→Md

當 Lot 在前一個區域生產完成後，則搬送系統會將 Lot 搬到下一個區域的 Stocker 暫存。然後在區域中，Lot 會根據製造流程中下一製程機台是否等待處理(Idle)決定目的機台，如果下一製程機台等待處理，則直接送到下一製程，此稱之爲直送，如機台皆在執行中，則將 Lot 搬回 Stocker。當製程機台狀態爲等待處理時，先確認其它製程機台上的 Lot，是否可以直送，如無法產生直送，則再從 Stocker 尋找等待此製造機台的 Lot 並搬出，如 Stocker 有多個 Lot 等待相同的機台，則以等待最久的優先處理。由 Lot 根據目前的狀態發出搬送需求，然後在搬送系統中將產生一筆搬送命令，搬送命令內容主要指定 Lot 及其目前所在的機台位置(From)及欲到達的目的機台(To)，然後交由 RGV 執行，RGV 先移動到 Lot 所在的位置，然後將 Lot 自機台搬到 RGV 車上，然後再移往目的地，將 Lot 自 RGV 搬到機台上，則結束搬送命令，每執行一筆搬送命令的時間約爲 20s~40s 之間，與 RGV 的位置及機台的遠近有關。

由於區域中可能同時有多個 Lot 發出搬送命令，故在每個時間點會有許多的搬送命令，則這些搬送命令組成命令列表。由於 RGV 只有一台，故每次只能針對一筆命令做

搬送，所以 RGV 每次在執行搬送指令前，需針對當時的命令列表挑選適當的命令，此稱之爲搬送策略。目前 TFT-LCD 工廠使用的搬送策略爲 FIFO(First In First Out)，其根據命令產生的順序來決定，先產生的命令會先被挑中執行。圖三爲 FIFO 搬送的比較的例子，如當時有兩筆命令先後產生①Mb2→Mc1 及②Ma3→STK，則綠色箭頭表 RGV 行經的路線，當產生順序爲①②時，RGV 的移動軌跡爲 STK→Mb2→Mc1→Ma3→STK 如圖三左圖，反之，當產生順序爲②①時，RGV 的移動軌跡爲 STK→Ma3→STK→Mb2→Mc1 如圖三右圖，命令產生的順序不同，則搬送的順序也不同。

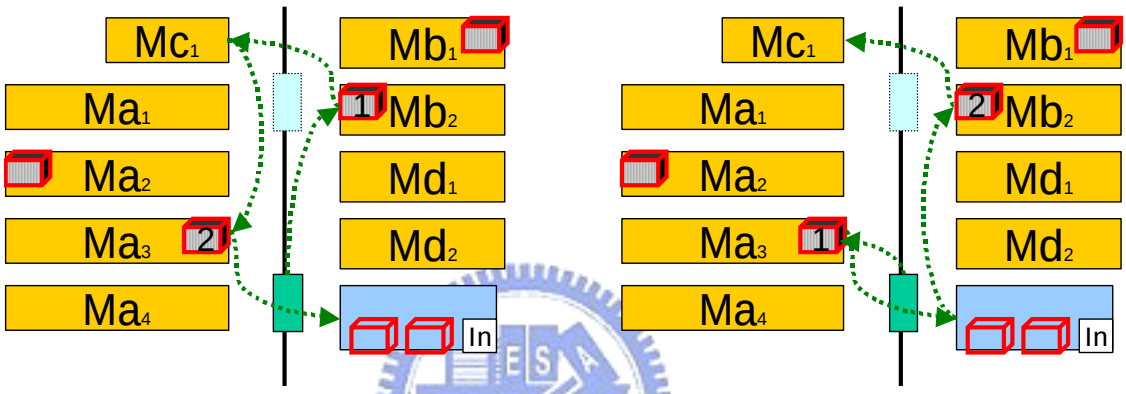

圖 3：FIFO 搬送方法的比較

當命令產生到命令結束此段時間稱之爲命令執行時間(RMt)，其中包含兩個部份，當命令產生後到命令被搬送策略選到，稱爲命令等待時間(CWt)，當命令被選到後到達目的機台爲命令搬移時間(CMt)，其中命令搬移時間包含了，RGV 的搬送時間及 Stocker 的等待時間。

當在區域中的製程處理完成後，Lot 將會被搬回到 Stocker，再移往下個區域的 Stocker，直到完成製造流程的所有步驟，則結束在陣列製程的製造。

1.2 研究動機與目的

由於陣列製程可說是 TFT-LCD 的根本，故其特別重視產能，希望能有越高的產能越好。產能主要受限於機台設備的數量及製造生產的時間，但無法以不斷增加機台設備的方式提昇產能，其原因在於兩者，一爲廠房的大小有限，無法增加過多的設備，二則

這些製造機台是非常昂貴的，買機台將造成製造成本及維護成本增加，故一般針對產能提昇，皆以減少生產時間的方式達成。

由於針對整個陣列製程的工廠的產能，進行產能提昇的研究較複雜，所需定義的參數及變數也較多較困難，且不容易發現調整前後的差異，所以希望依其工廠本身有的特性，將其畫分爲數個區域，再依個別的區域一一來討論其區域搬送和區域的產能。在區域中主要有三種不同的設備：製程機台、搬送設備、及倉儲，利用自動化的系統來生產，在電腦的運算速度很快的狀況下，因此可歸納出產能應該與機台、搬送車、倉儲使用率有關，且其三者彼此會互相影響且關係複雜。

針對區域的產能，很難透過計量的方式來計算，故希望利用生產時間來看區域的生產狀況，當 Lot 的生產時間越短表產能越高，故針對 Lot 進入區域後，在區域的生產時間一般可分爲三種：Lot 在製造機台的製造時間(Process Time)、Lot 等待處理的時間、及搬送命令執行時間。在正常狀況下同種機台每次製造時間，應該是相同，而 Lot 進入的次數固定，所以無法針對機台製造的時間調整。而 Lot 等待處理的時間及搬送命令執行時間，則與搬送策略有關。故想要利用搬送策略調整 RGV 搬送的順序，就可以影響區域整體的生產時間，進而提昇區域產能。但由於每個命令的執行時間很短(約 20 秒~40 秒)，很難在這麼短的時間找到最佳的解，故希望找到一個比較好的搬送策略。

在提昇工廠的區域產能方面，由於工廠的生產是連續生產的狀況，故我們希望提昇整體區域產能而非只有某一時間點有不錯的結果，因爲工廠的生產中每一個時間點都會影響下一個時間點的結果，所以不希望找到的方法只讓某各時間點很好，但卻造成後面 Lot 的生產時間增加，所以希望找到的搬送策略是對整體的產能提昇。

根據以上的特點，希望能夠考慮各種因素關係，針對區域找到一個搬送策略，且在大部分的狀況下有不錯的結果。

1.3 研究方法與成果

研究的目的是希望找到一個針對區域的搬送策略，也就是在許多的搬送策略中，挑選出一個比較適合區域的搬送策略。由於目前有許多已知的搬送策略，被用來挑選搬送

命令，例如常見的 FIFO(First In First Out)：先進先出、FEFS(First Encounter First Service)：最近的先搬。除此之外，仍有許其他的搬送策略未被提出或整理，其中某些搬送策略可能會在區域的運行得到比較高的產能，故希望找到比較適合區域特性的搬送策略。因此考慮搬送策略的數量多，且與產能相關的屬性彼此之間相互影響，很難用簡單的方法找到。故藉由導入基因演算法，利用基因演算法的特性找出區域適合的搬送策略。

因此在研究的方法中，提出以下的步驟:

1. 藉由文獻的紀錄，及實際與領域相關的專家訪談，找出足以描述當時命令狀況的屬性值。
2. 定義搬送策略及區域產能。
3. 將實際的問題對應到區域的模型中，並利用此模型建立對應的模擬器，模擬 LCD 廠實際的運作。
4. 導入基因演算法，找出適合的搬送策略。
5. 藉由模擬實驗找出結果，並進行分析比較。

而研究成果，如下列：

1. 提出一個 Heuristic 定義搬送策略爲搬送命令屬性的線性組合。
2. 提出區域產能的指標與整體 Lot 在區域製造時間的總和成反比。
3. 將實際的問題對應到合理的模型中，然後利用模型建立適當模擬器。
4. 利用基因演算法，藉由模擬器找出，區域適合的搬送策略，進而提昇產能。
5. 分析搬送策略與實際工作生產的關係。

1.4 研究範圍與限制

基於上述的研究背景與動機，本研究主要針對 TFT-LCD Array 製程，並且利用 RGV 搬送的工廠而言，以國內某 LCD 廠爲例，探討不同的搬送策略對區域產能的影響。針對不同特性的區域，能找到較適合的搬送策略，導入到區域中，能有效的減少區域整體的生產時間。故首先定義研究的範圍如下：

1. 本研究主要針對 TFT-LCD Array 工廠，且以 RGV 車搬送而言，不考慮其它搬

送方式或其它特性的工廠。

2. 本研究針對區域搬送對產能的影響，不考慮其它非關搬送的問題或異常的狀況。

TFT-LCD Array 工廠的生產製造流程複雜，包含的範圍相當廣泛，諸如：生產排程、機台派工、投料策略、Layout 規劃等。而本研究主要是在 TFT-LCD 廠陣列製程的區域中，找到一個搬送策略使區域有不錯的產能，達到增加產能的目標。過去針對增加搬送效能的研究，主要提出方法並且利用已開發的模擬器實驗，並訂定各種評估指標，與舊有的方法比較其差異性[14]。在這樣的方式下，其的範圍與限制，一般提出下列項目：

1. 施行的領域，例如: LCD 廠或是半導體廠。
2. 調整的項目，例如:搬送或是排程….。
3. 針對調整項目的包含項目，例如：以搬運而言，只討論內部行爲或外部行爲。

針對研究的範圍只提出大致的概念未能詳述，並未說明未討論項目的處理方式，且由於使用現有的模擬器，因此對於模擬器的限制也未能詳加說明，或是只概略說明使用的元件。

因此在本研究中，爲了減少研究時可能遭遇的困難，提出研究限制，並避免上述所提含糊或未定義的問題，針對各種狀況詳加列述，研究限制如下：

1. 本研究只針對一般正常 LCD 的製造流程不考慮抽檢的部份或其它不同製程的流程，故針對所有的 Lot 而言，只有唯一一種製造流程，無其它異常製造流程或重工(Rework)的現象皆不考慮。
2. 本研究針對 TFT-LCD Array 工廠中的區域各別討論，只考慮區域中的影響，不考慮區域間的移動，及非選定區域內的影響。
3. 針對區域內各種不同的製程機台，如這些機台在其它區域也有時，不考慮其它區域部份分工的狀態，由調整投入數及更改流程設定的方式，來使區域更貼近實際狀況，忽略其它區域的影響，例如：如在實際工廠中，某一類似區域有三個，挑選其一，則原工廠的投入數爲每日 120Lot，由於針對單一區域，故投入數減爲每日 120/3=40 Lot，即可貼近工廠實際狀態。
4. 本研究所探討在穩定的工廠而言，製程機台每次處理時間固定，並且不考慮人

爲因素、機台當機、機台保養、或其它異常狀態，假設其在最好的狀態下，故不同的光罩及機台設定各種製程參數皆可使用，並且不考慮更換光罩、參數、物料所需花費的時間，並且每個機台只有一個 Port 可共上下 Lot。

5. 本研究針對固定的搬送行爲討論，不考慮其它經調整過後的搬送行爲。
6. 針對 RGV 而言不考慮因機械造成非定速的情形，例如：RGV 起步後有等加速度，加速到一定值後等速移動，本研究不考慮起步及停車的等加速度，及 RGV 空車及運載中的速度應相同，固 RGV 以定速移動，並且每次只能搬送一個 Lot。
7. 針對 Stocker 而言，Stocker 將 Lot 自 Prot 移進倉儲中，會根據倉儲放置的位置遠近，時間上應有不同，本研究不考慮內部移動的差異，將自 Port 搬到倉儲的位置爲一定值，並且定義 Stocker 的行爲爲同 Prot 進出，每次只能處理一個 Lot，且 Stocker 內可存放量無限制。
8. 本研究針對搬送策略討論，不考慮其它非關搬送的調整或策略的導入，例如：派工管理、物料管理、生產規劃、作業方法，以現有的機台布置及機台全自動生產爲原則。

1.5 論文架構

本論文共分爲六章，第一章主要介紹目前 TFT-LCD 的生產現況，及針對研究目標 Array 製程的作業流程做詳細的介紹，說明研究的動機範圍及目的。第二章文獻回顧分爲兩個部份，首先介紹自動化搬送系統的相關研究，第二部份爲研究所需的基本知識。第三章主要根據目前的製造現況，描述問題定義，並且提出相關的假設，建立適當的模擬器，及定義區域產能計算方式，利用區域產能的大小來衡量搬送策略的有效性。第四章的部份，主要根據第三章所提出的問題定義，建立找到適合搬送策略的方法。第五章利用前述的內容建置系統，執行模擬實驗，並分析實驗的結果。第六章則爲歸納本研究所探討議題的結論，並提出對未來此研究的相關建議及發展方向。

二、文獻回顧

欲瞭解自動化搬送系統對產能的影響，首先必須瞭解搬送系統的作用與運行對生產的影響，而目前對於搬送系統的研究，主要探討半導體廠的搬送效能，在各種不同的搬送策略下搬送的狀況，及當時產能的影響。而 TFT-LCD Array 製程很類似於半導體廠的搬送方式，故在 2.1 節介紹半導體廠自動化搬送系統及其目前所針對的問題及解法。而除搬送系統的研究以外，本章節還回顧基因演算法相關內容，透過對基因演算法的瞭解，才能利用基因演算法的特性，找到適當的解，以利提昇產能，因此 2.2 節主要介紹基因演算法相關知識。

2.1 自動化物料搬送系統研究

自動化物料搬送系統的起因，主要由於在半導體工廠中，隨著晶圓的大小及重量越來越大，使得晶圓的搬運無法藉由人力來完成，並且在晶圓在製造過程中，對於環境中的塵度非常重視，當塵度過高將造成不良率的升高，造成莫大的損失，故希望讓工廠內部的塵度越低越好。而根據研究發現人體本身會發塵，如使用人力搬送則容造成汙染，故一般在此類型的工廠中都轉換爲自動搬送系統，減少因人而產生良率下降的問題[18]。

對於半導體的搬送系統而言，主要用來運送晶圓到製程機台上，其搬送的方式有許各種不同的類型，目前我們只討論與第一章提到的 TFT-LCD Array 工廠區域中較近似的搬送方式，也就是晶圓在同一個加工中心(within bay)移動，即晶圓只在機台與機台或機台與倉儲之間移動。

在討論搬送的效能問題之前，先簡單介紹相關文獻中，對於搬送設備的規劃與功能，一般而言，在同一個加工中心中移動，需借助搬送車的功能，而又有許多不同種類的搬送車，先前主要有兩種:OHT(Overhead Hoist Transport) 及 AGV(Automated Guided Vehicle)，OHT 是屬於空中軌道的搬運，需先架設好軌道且一般會規劃跨區域的搬運，而 AGV 爲地面上之搬運，不需架設軌道，與 RGV 相較之下在規劃路線上較有彈性，OHT 及 AGV 相同點爲，都爲單方相封閉式的運行，一般在路線上有多台的搬送車[18]。

故其兩者會產生的問題近似，以目前的研究而言大多以 AGV 來探討。

針對這種單方向多台車運行的搬送方式，在自動化的搬送系統中，將對搬送車管理造成很多的問題，因爲有多台車且單方向運行，故會有許多車量派送、死結、塞車、及路途遠近的問題，文獻中也有許多針對這些問題的解決方式[1][5][11][12][15][16][17][18][19]，將在 2.1.1 節中詳細介紹。在自動化搬送系統下，除了搬送車會影響搬送的時間，進而影響生產時間外，另外就是搬送的次數也會影響生產時間，故文獻中提出減少無謂的搬送，來達到提高搬送效能的目的，這方法將在 2.1.2 節介紹。

2.1.1 搬送車管理及派車問題

搬送車的管理與派車問題上，主要針對塞車、死結、派車及路途遠近的問題，提出相關的解決方案，分別提出許多方法來解決問題。

對於搬送車在封閉環境中單方向運行，如有多輛車時可能會發生壅塞及死結的現象，針對壅塞的問題提出 PV(Push Vehicle)法則，將阻擋搬送任務的閒置車輛(未接受搬送任務)，往前推進，保持軌道暢通，減少塞車的問題[19]。對於死結的問題，提出 STS(Send to Stocker)法則，當無法載運到目的地或目的地無法卸載時，將晶圓暫存到 Stocker 中，減少死結的產生及搬送等待時間，相對可以提高搬送車的績效，並利用實際模擬結果，發現此方發對產出量及搬運時間有不錯的表現[19]。

另外針對雙迴圈搬送系統，提出針對系統環境的變化，來考慮使用晶舟選車或車選晶舟[17][11]。此兩種方法的不同點，優先考慮搬送車的績效或是優先考慮晶舟的週期性，舉例而言，車選晶舟的方法如 FEFS (First Encounter First Service)，搬運車選擇最近的優先處理，LWT(Large-waiting-time transport)有較長的等待時間先處理如下圖，而 SD-NV(Shortest Distance-Nearest Vehicle)是一個晶舟選車的方法，先選擇搬運最短的路徑，在選最近的車子，處理晶舟的搬運需求。模擬的結果，發現在搬運量不同時，不同的搬送方法有不同的表現。

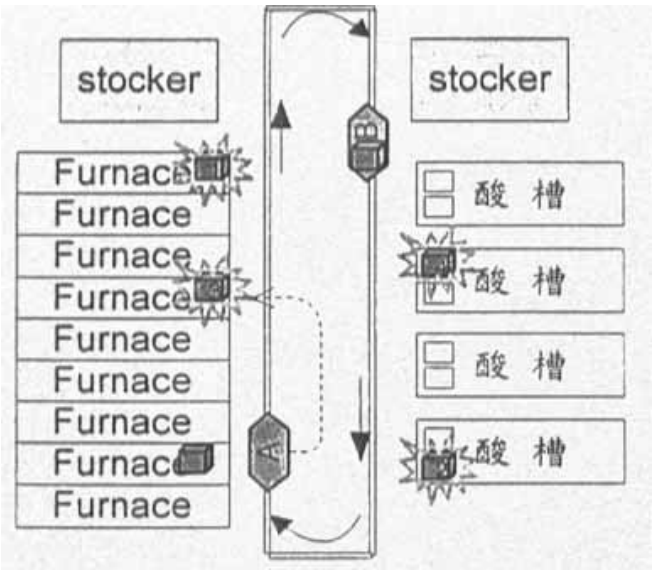

圖 4：近的先搬策略(First Encounter First Service transport)

來源：[18] 張原銘, 晶圓廠之自動化物料搬運系統模擬研究—以黃光區與爐管區爲例,國立清華大學工業工程與工程管理學系碩士論文,2003

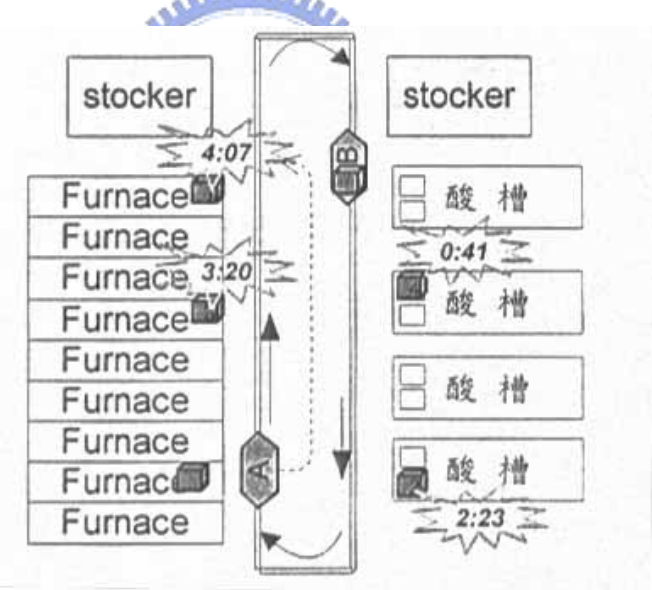

圖 5：等待時間長的搬送策略(Large-waiting-time transport)

來源：[18] 張原銘, 晶圓廠之自動化物料搬運系統模擬研究—以黃光區與爐管區爲例,國立清華大學工業工程與工程管理學系碩士論文,2003

在派車問題方面，指的是當有搬送需求產生時，如何從一群車中，選取一台最適合的搬送車去執行搬送。一般考慮的問題提出一到兩個因素來考慮如何派車。例如利用模擬的方式來看不同的派車法的績效表現[1]。發現以距離爲基礎的派車比利用不同屬性

(區域內或全區域)爲基礎的派車法則好。前面所提爲來源導向的派車法則，另有提出需求導向的派車法則[5]，考量的因素以工作站爲依據，由系統面來考慮派車原則，而非以車或晶舟爲派車原則。兩者的不同主要在選擇物件(車或晶舟)還是選則製程的執行順序不同[15]。並且其研究的結果發現，高搬運量的狀況下，兩者的產出率表現是差不多的。

前面主要利用單一個因素來控制搬送的順序，但無法讓派車更有有效的執行，故提出利用等候的先後及車輛移動時間，來增加派車的績效[12]，這樣的方法比只考慮單一因素的結果來的好。

由上面文獻中關於自動化搬送系統的研究，可歸納下面幾個結論：

1. 增加搬送績效，由於可減少生產時間，故可增加生產量。
2. 針對不同的搬送狀況，適合不同的派車方式，而派車方式可說是針對每個搬送命令，來挑選搬送車。
3. 一般的派車法則都取單一因素的影響，例如 FEFS 爲近的先搬，FIFO 先進先出。
4. 研究中主要利用系統本身的變化導入一個單一的派車法則，例如：開始使用 FIFO，當量逐漸增大的時後改採 FEFS 的策略。
5. 考慮多個因素會比只考慮一個因素的效果好，例如單考慮車輛移動的距離，和同時考慮等候的先後，後者更能提昇效率。

但由於先前提到的方法主要都是針對封閉式單向行走方式的搬送車，由於此特性封閉的軌道中都有多輛搬送車同時運行來增加其搬送的效能，例如：OHT 及 AGV，故主要針對這樣配置的缺點進行提昇，例如因爲多輛車故易有塞車、撞車、死結的問題，由於單向運行，錯過後要再繞一大圈，故著重派車的問題，有效的派車方式將大量的增進效益。

至於 RGV 車搬送的環境，主要是 RGV 在已鋪設軌道上運行，通常一段軌道上只會有一台 RGV 車，並且車可原地折返，目前沒有人針對 RGV 的特性提出解決方法，探討及原因，主要有二點，第一點爲 RGV 的本質上屬於須先於地面鋪設軌道，不易更動行經的路徑，相較於半導體多變化的製程，其可調動的幅度較小，故半導體廠較不考慮使用，雖然本身限制難以更動，但對於不需經常更動的工廠，RGV 的速度比 AGV 快了許多，而 LCD 的製程變化不大，故 RGV 一般使用在 LCD 廠，第二點爲 RGV 的本質上

已避免掉很多的問題，因爲可原地折返，且只有一台車，故不會有塞車、死結的問題，且原地折返針對每個搬送命令而言，不會因錯過標的物而多繞一圈浪費時間。由 RGV 的特性可知，上面針對 AGV 特性的結論並不適用於 RGV 上。

2.1.2 Push & Pull 法則

先前提過，減少生產時間主要爲增加搬送效能及減少無謂搬送，2.1.1 節已提過文獻中對於增加搬送效能研究的理論與方法，而在此節主要針對減少無謂的搬送方法做介紹，Push & Pull 法則主要目的爲減少無謂的搬送，此方法爲張原銘於 2003 年提出[16]，其是以機台的角度當做出發點，加入生產行爲的考量，因此機台有了製造加工時間、產能與容量的限制，判斷晶舟是否需要移動，以下就詳細介紹此法則的做法。

此法則其分四種狀況來討論，(a)當晶舟於瓶頸機台結束製程時，需呼叫搬運車要求搬運，而此時要決定搬運的目的地，晶舟先”向前看”判斷是否有其他晶舟準備進入次機台加工，若有，則其他晶舟會對此機台上的晶舟執行”Push”，使其搬離而確定是釋放產能，接著”向後看”判斷下一步驟的機台是否閒置，若閒置，則下一步驟機台執行”Pull”將晶舟搬到下一步驟機台。(b)當機台需釋放產能，且下一步驟機台忙碌時，無法執行”Pull”，則將晶舟搬回 Stocker 暫存，以確保能釋放機台的產能。(c)當無 Push 時，表無等待的晶舟需要進此機台，而下一步驟機台閒置時則產生 Pull，則直接搬到下一機台。(d) 當無 Push 時，而下一機台忙碌時，則無法產生 Pull，此時將晶舟留在原來的機台上，等待其他的晶舟或機台的觸發，這樣的方式對機台的使用率並無大影響，但可減少搬送車的負擔及進 Stocker 的次數[16]。

Push & Pull 法則主要是利用機台的 Port 當作暫存，當有需要的時候才將晶舟搬出，與傳統的搬運方式比較，減少進入 Stocker，並增加機台之間的直送率，能有效明顯增加產能。增加機台之間的直送率，可使 RGV 及 Stocker 減少搬運次數，相對可節省整個運送時間及不佔用 RGV 及 Stocker 的資源，故一般來說要有好的搬送效率，希望可以增加直送的次數，因每次並不一定下一步驟機台有空，故須原地等待，但原地等待會造成機

台無法使用，使得機台使用率下降，這也是不樂於見到的情形，所以 Push & Pull 法則可說是兼顧機台使用率及增加直送的方法。

文獻中利用模擬器來探討有無導入此方法的差異，並訂定各種指標來探討其關係，發現在高投產量且搬運車輛少的狀況下[16][20]，有較顯著效果，原因爲當投產量高時，相對搬送及暫存量會提高，此時若無增加搬送車，則易拖累其產能，但使用 Push & Pull 後，可用少量的搬用車，去解決搬送的問題。而利用這樣的方式，可能會造成的問題如下，機台的使用率下降，雖然此 Push & Pull 法則，也注重機台的使用率，但某些情形仍會拖累機台的使用率及生產時間，例如：晶舟可能佔住 Port，等到有晶舟要進入時才”Push”，此時就需要搬送車先將 Hold 在機台上的晶舟搬離，才能在將其他的晶舟搬入，故欲進入的晶舟就需等待之前的晶舟搬離的時間，此時多了等待的時間，相對機台的使用率下降，且其他的晶舟等待時間變長。

2.2 基因演算法

基因演算法（Genetic Algorithms:GA）是一種啓發組合式（heuristic combinatorial）的最佳化搜尋方法，是最佳化工程計算常用的一種演算工具。於 1970 年由密西根大學 John Holland 教授及他的學生首先提出之演算法則，主要依據生物學家查理・達爾文（Charles Oarwin）在 1859 年出版的 ”物種原始” 一書所提出的適者生存理論（Darwinian survival-of-the-fitness philosophy）所發展出來一套演算法則，其基本精神在於模仿生物界物競天擇、適者生存的自然演化過程，以求得問題最終穩定且最佳的結果。故基因演算法源自於自然界中”物競天擇”及”適者生存”的特性，模擬生物間的競爭，倖存者得以繁衍下一代的觀念，藉著對於演化現象的觀察，John H. Holland 認爲可以透過把問題轉爲基因型(Genotype)，利用競爭-生存以及基因交換-突變，尋求出問題的正確解答[10]。因此，”生存”、”競爭”、”交配”、”突變”就是整個基因演算法，也可以說是演化的中心思想。

透過競爭-生存，擁有好基因的品種會有較高的機會生存到下一代，而與生存較無

關的基因則會隨著時間逐漸被淘汰，。在理想的狀態下，競爭-生存會迫使不具優勢的品種逐漸消失[3][6]。

然而單單擁有一種好基因是不夠的；透過基因的交配，不同的個體可以把它們的好基因組合起來，變成更具優勢的下一代；而若是組合出來的後代不理想，也只會加速被淘汰。但交換基因也不能改變現有的基因狀態，還要再配合突變，才能夠產生革命性的改變，進而對族群進步有突破性的發展[8][14]。

基因演算法與傳統搜尋法最大不同之處在於它是採多點搜尋的方式在參數空間中同時尋求問題最佳解，在尋找過程中僅需藉由所定義的適合度函數值（Fitness）作爲演化過程中性能指標依據，再經由演化機制-複製、交配、突變，適切地調整搜尋的方向及區域，使能逐漸搜尋到最佳解，目前基因演算法已經廣泛地被應用於不同領域，如：物流分配、工程科學應用、人工智慧 ……等等。

基因演算法中，最開始的資料結構隨機產生一些可行的解，在根據這些解所得到的效用，當作調整這些資料結構，使其不斷的提高效用。Holland 證明出利用基因演算法可得到最佳解或得到趨近的最佳解[9][10]，在傳統找尋最佳解的方法對於一個有很多可能的問題上很難有進展[7]，所以一些關於最佳化的做法，利用一些限制來縮小範圍做出小範圍的搜尋方法，然而這樣的方法很難應用到實際的問題上面，由於搜尋的範圍太廣而無法有效使用。

2.2.1 基因演算法理論

基因演算法是模仿生物進化遺傳的過程，因此基因演算法：，應用此基本觀念即可完成演算法之運算。傳統之演算法往往由幾個起始點，依照一定的數學模式產生下一次疊代值，如此反覆計算求得最佳解；而基因演算法以隨機方式產生許多的點，同時搜尋最佳解，因爲在每一次疊代過程皆是取相對最佳的點，因此只能找到最接近之最佳解。不過因爲其他演算可能也只找到局部最佳解 (Local Optimum) 而不能保證是真正的最佳解 (Global Optimum)。基因演算法是相當不錯的最佳化運算工具。

基因演算法主要是在許多對於問題的可能的解中搜尋最好的答案[13]，一般而言在

基因演算法中關於這些解的表是方式都利用一連串的 0 和 1 來表示，而在問題中用這樣表示的解，稱之爲基因，對於每個基因都會對應到一個目標函數，此目標函數稱之爲適應函數(Fitness Function)。在原來的問題上，一個好的基因會對應到一個很高或很低的適應函數值(Fitness value)，很高或很低由問題的本身需求出最大或最小值決定，對於每一個基因的強度會顯示在適應函數值上，適應函數值代表基因被帶到下一子代的可能性。一些基因及其適應函數值的集合稱之爲族群(Population)，族群將利用基因演算法的規則產生出每一子代，產生的規則如下：

```
Genetic Algorithm()
{
    Initalize population;
    While(not terminal condition)
    {
        Selection parents from population;          /*Selection*/
        Construct offspring by combining parents;   /*Crossover*/
        Mutation offspring;                         /*Mutation*/
        If suited(offspring) then
            Replace worst fit(population) with better offspring;
    }
}
```

首先產生隨機產生一個族群，利用初始的族群開始演化，演化之前需設定停止點，在何種狀況下取得最佳解，然後經過選取(Selection)、複製（Reproduction）、交配（Crossover）、突變（Mutation）得到下一子代，直到符合停止的條件。

基因演算法中每代產生固定數量的族群，意指只有比較據有優勢的基因會被帶到下個子代，直到得到最佳或趨近最佳的解才停止搜尋[9]，因此在機因演算法中三個主要的運算子：取(Selection)、交配（Crossover）、突變（Mutation）將在下一節討論。

2.2.3 選擇函式(selection function)

由於每一代的族群大小是固定的，是故只有基因有好的適應函數值，才會被留到下一子代，這樣的方法比隨機取樣的方式更能得到較快達成目的[7]。

選擇函式(selection function)；一般來說，選擇函式都會依照適應函式的分數來作爲判斷依據，但選擇有很多方式，最簡單的就是前面一半就晉級，另外也有依照得分多寡加權計算機率的，甚至於保送到下一代的方式。選擇函式做的就是"天擇"的動作。

2.2.4 交配

交配主要有兩各動作：首先，從上面選擇函式中得到的集合中，隨機任意取得兩個基因，第二個動作，將兩個基因利用固定的方法生成兩個新的子代，通常利用單點切割的方法，先產生單點切割的位置 c，將兩個基因在 c 位元以後的質交換產生下一子代，例如：假設基因的長度都爲 5 個位元，目前挑選到量個基因分別爲 x=11111 和 y=10101，如此時單點切割的位置爲 3，則產生新的子代爲 11101 及 10111，其示意圖如下圖 6。除此之外還有其它各種的方法，雙點切割或是平均切割.....等[1]。

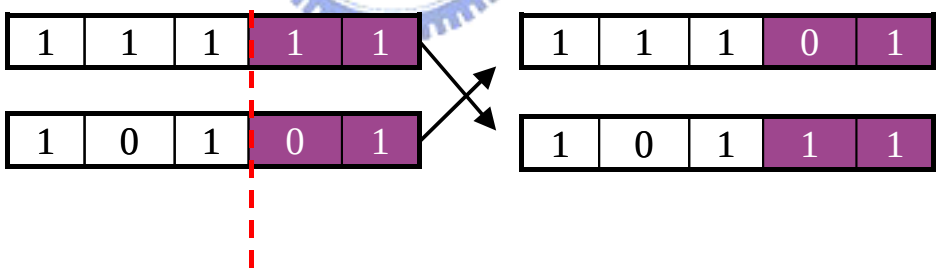

圖 6：基因演算法單點切割交配

2.2.5 突變

做完前面介紹的選擇函示及交配之後，已產生下一子代，如不做突變，則容易造成產生只針對部份問題集合的最佳解，而非全區域的最佳解。因爲子代都是由前一代所產生的，故如在隨機初始化的時候，未產生某一種還不錯的組合，則將來一定不會產生，造成並未找到真正的最佳解。故在已產生的子代需加入突變的方法，來產生非初始集合

內的值。利用突變率來控制基因是否產生突變，例如設定突變率爲 0.05，則針對每個基因內部的位元，產生一個隨機的變數，當其小於突變率時發生突變，將此位元由 0 變 1 或 1 變 0，而突變率的大小將會影響尋找最佳解的速度。

三、問題定義

由前面所提到的目前相關研究，一般而言皆只是提出一、二個影響搬送效能的因素，找出改善小部份狀況的方法，如此未能考慮大部份的狀況，也沒探討其他因素的影響。如相關文獻所提出：進入 Stocker 的次數將影響搬送的旅程時間，提出利用生產機台當做暫存區，有效減少進去 Stocker 的次數及旅程時間，但此方式未考慮機台的使用率可能會下降的問題。另外，這些研究主要針對在 AGV(Automated Guided Vehicle)搬送下，因此結果是針對 AGV 的特性，解決多台車單一方向繞行的問題，例如：塞車問題、搬送遠近的問題、撞車的問題。當使用 RGV 搬送時，由於 RGV 的特性，區域中只有一台車雙向移動，不同於 AGV，則不產生上述問題。

因此希望針對陣列製程中單一區域，導入一適用於區域特性的搬送策略來選擇搬送命令，進而提昇區域產能。首先需建立一個盡量符合現實工廠狀況的模擬器，並提出相關的假設，主要是每次離開此特定區域後再回到區域的時間。製作此模擬器主要目的，是爲了利用模擬器計算此特定區域的總生產時間，再利用生產時間來評估區域產能的大小，時間愈短表區域產能越高。接著提出搬送策略爲命令屬性的線性組合，則線性組合的所有可能爲搬送策略的集合。由於屬性之間的組合複雜且互相牽制，故利用基因演算法找出較適合的權重，得到一個針對此特定區域較適合的搬送策略，如此一來就可以藉由此搬送策略提昇區域產能。

3.1 模擬器(Simulator)

TFT-LCD Array 工廠的實際的製造複雜，故先將實際的問題定義到針對區域的合理模型中，再利用模擬器來模擬工廠實際的運作，藉此得到每個 Lot 在此區域的生產時間。在此針對特定區域的模型中，提出 Lot 投入後，經過一段時間開始第一次進入區域製造，完成本次區域的製程後，藉由 Stocker 離開區域，每次離開此特定區域後，應經過一段時間又回到此區域，除最後一次離開。假設本實驗選定 A 區域爲實驗對象，則區域模型如下圖七所示。投入到第一次進入區域的間隔時間，及每次離開後再回到區域間隔的時

間可都能不同，但不同的 Lot 在相同的製程下，其同一次離開區域後再回到區域間的時間應相同，而最後一次離開特定區域則結束生產，其中如何取得設定值將在第五章模擬實驗中介紹。

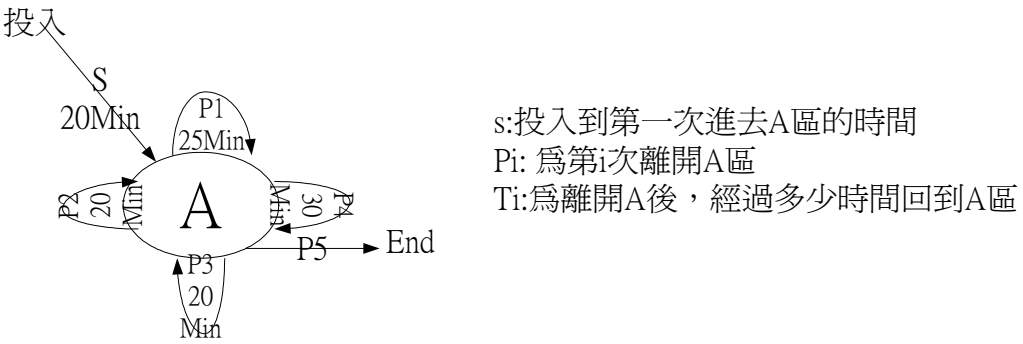

圖 7：模擬器模型示意圖

模擬器首先需給定每日的投入量，因爲受限於生產機台的數量，無法同時開始生產，故須先給定投產機台的數量及其處理時間。然後給定 LCD 的製造流程，指定本次模擬的區域，則可知進入指定區域的次數及內部機台的製造順序，然後給定每次離開後經過多久的時間回到此區。針對所挑選的區域，給定此區域相關的參數：機台的種類、各種類機台的數量、各種機台的處理時間、機台之間的距離、RGV 的速度、RGV 搬動 Lot 的時間、Stocker 內部移動時間，則模擬器中的製造流程如下圖 8。

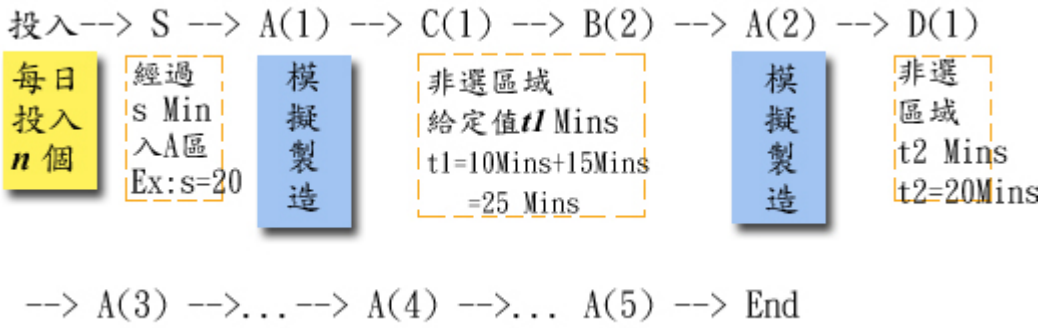

圖 8：模擬器中製造流程圖設定示意圖

給定以上的參數後，模擬器會根據投產機台的數量及其處理時間讓 Lot 進入生產線上，例如投產機台兩台，處理一個 Lot 需要 10 分鐘，則每 10 分鐘產出兩各 Lot 到生產線上，當 Lot 到非指定的區域，則其所耗費的時間爲先前給定的時間參數，當 Lot 從

Stocker 進入指定區域，則 Lot 根據目前的狀態發出搬送命令，再利用搬送策略挑選適當的搬送命令，交由 RGV 執行，直到區域內的製程完成，回到 Stocker 內，移動到下區域，直到 Lot 完成生產。如以區域"A"爲例，工廠中投入的機台數爲 2 台，每經過 10 分鐘有一個 Lot 產出，則前六個 Lot 的總體生產時間如表 2，而每個時間點在區域中的 Lot 列表如表 3,括號中註明此爲第幾次進去 A 區生產。

表 2：Lot 生產時間列表

LOT	投入00:0(	Not A(20)	A1(20)	Not A(25)	A2(10)
L1	00:10	00:30	00:50	01:15	01:27
L2	00:10	00:30	00:52	01:17	01:29
L3	00:20	00:40	01:01	01:26	01:43
L4	00:20	00:40	01:06	01:31	01:46
L5	00:30	00:50	01:17	01:42	02:02
L6	00:30	00:50	01:20	01:45	02:10

表 3：區域中製造的 Lot 列表

時間	A 區Process
01:00~01:1(	L3(A1),L4(A1),L5(A1),L6(A1)
01:10~01:2	L1(A2),L2(A2),L5(A1),L6(A1)
01:20~01:3	L1(A2),L2(A2),L3(A2),L4(A2),L6(A1)
01:30~01:4	L3(A2),L4(A2),L5(A2)
01:40~01:5	L4(A2),L5(A2),L6(A2)
01:50~02:0	L5(A2),L6(A2)

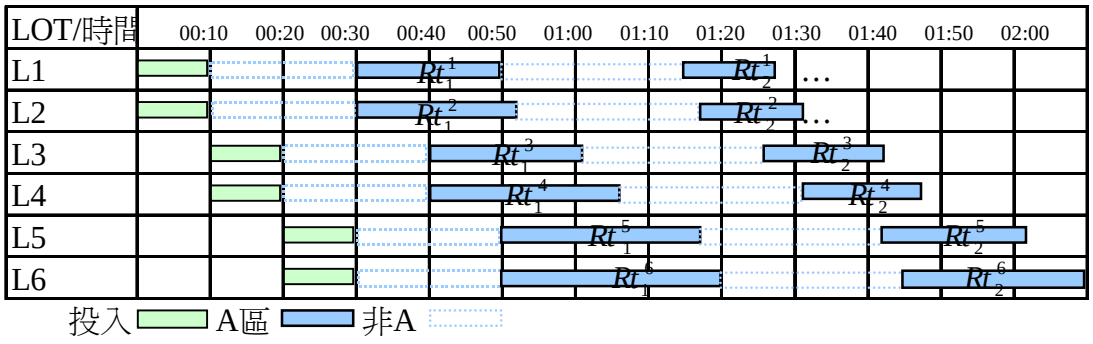

圖 9：Lot 生產時間圖

則根據上述，圖 9 爲前六個 Lot 的生產時間示意圖，在非 A 區域中的生產，每個 Lot 所需的時間相同，而在 A 區中模擬生產，則由於資源有限分配的狀況下，每個 Lot 生產時間會互相影響而有所不同。

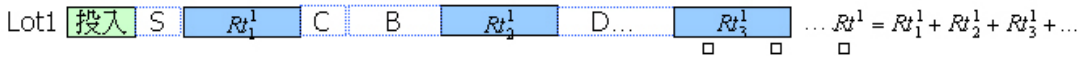

圖 10：單一 Lot 生產時間的圖

則針對單一 Lot 來看，則其生產時間如圖 10 所示，則定義每次進入 A 區域的生產時間爲 Rt_j^i ， i 代表爲 Lot 的編號，而 j 代表爲 Lot 進入區域的次數，則編號 1 的 Lot 在區域 A 的總生產時間爲 $Rt^1 = Rt_1^1 + Rt_2^1 + ...$ ，則所有 Lot 在區域的總生產時間爲 $Rt = Rt^1 + Rt^2 + ...$ 。

在模擬的過程中，將會記錄每個 Lot 在此區域的停留時間，主要可分爲三種：機台的處理時間(RPt)、搬送時間(命令執行時間:RMt)、等待時間(RWt)，故 $Rt_j^i = RPt_j + RMt_j^i + RWt_j^i$ ，如圖 11 所示，Lot 自 Stocker 進入區域後，自 Stocker 搬出然後藉由 RGV 搬到機台的時間爲搬送時間，到機台後產生機台的處理時間，而當機台都忙碌時，在 Stocker 停留爲等待時間。

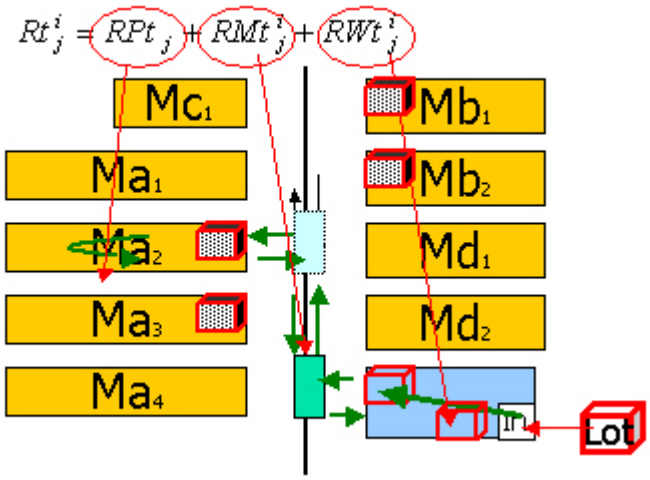

圖 11：區域內部生產流程及生產時間對應圖

3.2 搬送策略

搬送策略主要使用在每個時間點挑選命令列表中搬送命令，而挑選命令的目的在於希望能使區域有較好的區域產能，在區域中影響區域產能的因素，經由先前的研究及專

家的建議可知：生產機台的使用率、RGV 的使用率、Stocker 的使用率。所謂機台的使用率，指的是生產機台處於生產的狀態的時間，如果機台一直處於生產狀態，針對此機台有比較高的產能，故可知希望機台使用率越高越好；而 RGV 的使用率，指的是 RGV 處於搬送的時間，而對於生產時間而言，希望 RGV 搬送 Lot 的次數越少，搬的的距離越近，則生產時間越短，故希望 RGV 的使用率越低越好；Stocker 的使用率，指的是 Stocker 將 Lot 從 Port 搬到 Stocker 內部的定點，此時爲使用 Stocker 的狀態，針對生產時間而言，每進一次 Stocker，則需花費 Stocker 搬運時間，故希望越少進入 Stocker 越好，故希望 Stocker 使用率越低越好。根據上述影響產能的因素，找到能夠描述每筆命令狀況的參數如下:

WIP : 命令中來源機台的待處理數(個)
C : 命令中來源機台的數量(個)
T : 命令中來源機台的處理時間(s)
NWIP : 命令中目的機台的待處理數(個)
NC : 命令中目的機台的數量(個)
NT : 命令中目的機台的處理時間(s)
RR : RGV最候停的位置到來源機台的距離(m)
CW : 命令產令產生後到目前候時間(s)
SW : 使用Stocker須等候的時間(s)

例如：表 4 爲某時間的命另列表，其中每筆搬送命令，都帶有上列的參數。

表 4：區域中某時間點的命令列表

	Lot	From	To	WIP	C	T	NWIP	NC	CT	RR	CW	SW
1	L1	Ma1	STK	5	4	1800	0	0	0	9	80	0
2	L2	STK	Ma2	0	1	0	5	4	1800	0	70	40
3	L4	Mb1	STK	4	2	300	0	1	0	12	65	0

搬送策略主要挑選命令，故本研究提出搬送策略是命令狀況的屬性的線性組合。針對每一個時間點的，給定命令列表，針對命令列表中的每個命令計算出一個 f 值，由值高的決定，那筆命令先做。假設其存在一組值，使得每次都可以挑選到比較適合的命令，則將得到較好的區域產能。且由目前較常使用的搬送策略，FIFO(First in First Out) 先進先出、FEFS(First Encounter First Service)近的先搬，都是搬送策略函數的特例，例

如：FIFO: 1*CW，FEFS:1*RR，然後決定挑選的命令。

搬送策略函數：定義 *f* 爲一個搬送策略的函數:

$$f_{w_1,w_2,w_3,w_4,w_5} = w_1 Att_{From} - w_2 Att_{To} - w_3 Att_{RGV} + w_4 Att_{LOT} - w_5 Att_{\mathrm{STK}} \text{.....(1)}$$

Att_{From}:搬送命令來源機台狀態
Att_{To}:搬送命令目的機台狀態
Att_{RGV}:RGV與搬送命令來原機台距離
Att_{LOT}: 搬送命令等待時間
Att_{STK}: 搬送命令預計須等待STOCKER的時間

$$0 \le w_1, w_2, w_3, w_4, w_5 \le 1$$
$$w_1 + w_2 + w_3 + w_4 + w_5 = 1$$

其中 Att_{From} 表示來源機台的狀態，其應該表現出目前 Lot 所在機台的忙碌狀態，如果此機台越忙，則越需要將其搬離，故其應該優先被挑選到，與 f 成正比； Att_{To} 爲目的端機台的狀態，其越忙碌則希望 Lot 下個 Lot 能夠直送減少搬送時間，故希望 Lot 能在原機台等待，越晚搬越好，故與 f 成反比； Att_{RGV} 爲 RGV 與搬送來原機台地距離，因爲距離遠近將影響時間的 RGV 行走的時間長短，故希望先執行近的命令，與 f 成反比； Att_{Lot} 爲搬送命令產生到目前的時間，爲了防止一直選不到此命令的狀態，造成 Lot 等待過長的狀態，並且長時間的等待也會影響良率，故希望其不要等太久，等越長的越容易被選到，與 f 成正比； Att_{STK} 表示需等待 Stocker 的時間，等待的時間越久，表示造成搬送花費更多的時間，故希望等待 Stocker 時間越短越好，與 f 成反比。

由於無法知道各屬性之間的權重關係，因此利用一組 w_1, w_2, w_3, w_4, w_5 將屬性值加總起來，且爲了讓權重值之間彼此互相牽制，及防止重複的現象，故設定權重值的總和爲 1。

$$Att_{From} = Normalize_{from}\left(\frac{WIP \times T}{C}\right), \quad 0 \le Att_{From} \le 1 \text{.....(2-1)}$$
$$Att_{To} = Normalize_{To}\left(\frac{NWIP \times NT}{NC}\right), \quad 0 \le Att_{To} \le 1 \text{.....(2-2)}$$
$$Att_{RGV} = Normalize_{RGV}(RR), \quad 0 \le Att_{RGV} \le 1 \text{.....(2-3)}$$
$$Att_{LOT} = Normalize_{LOT}(CW), \quad 0 \le Att_{LOT} \le 1 \text{.....(2-4)}$$
$$Att_{\mathrm{STK}} = Normalize_{STK}(SW), \quad 0 \le Att_{STK} \le 1 \text{.....(2-5)}$$

Att_{From}及Att_{To} 主要顯示出目前來源及目的機台的狀態，狀態主要希望顯示出目前機

台忙碌的程度，由忙碌的程度來決定要不要先處理此筆命令，而機台忙碌的程度表示方式，可轉爲目前待處理時間，故以機台待處理數乘以機台處理時間表示待處理時間，但由於同種類的機台有數台，爲了得到正確的待處理時間，需再除以機台數量，則以此數值代表機台的忙碌狀態；而 Att_{RGV} 表示爲 RGV 與來源機台的距離，並不將需要的 Lot 搬送距離列入，因爲命令的執行先後並不影響 RGV 搬送 Lot 的距離，只與 RGV 目前所在位置與 Lot 所在位置有關，故只考慮 RGV 與 Lot 的距離爲 Att_{RGV} ； Att_{Lot} 則是目前命令已等待的時間；Att_{STK} 則是 Stocker 所需要等待的時間，由於 Stocker 的資源只有一個，故希望考慮 RGV 需等待 Stocker，以防止 Stocker 造成 RGV 閒置過久影響搬送的效能，定義完這些屬性之後，將這些屬性正規化(Normalize)到 0~1 之間，以避免屬性之間的落差過大。

由這些屬性可發現考慮了區域裡面所有的設備：製程機台、搬送車、Stocker，藉由這些屬性來代表每筆命令在這區域的狀況，但不瞭解屬性之間的對搬送策略的影響，故把這些屬性利用權重值線性組合，形成搬送策略函數。所以希望找到一組權重值，能使區域有比較好的區域產能。

3.3 區域產能

對於產能的研究，通常以單位時間的生產的數量來計算，例如：每日產出 100 個 Lot，但由於工廠處於不斷生產，且多次進出區域，每次進入區域後的製程可能不同造成生產時間也不同，故難以利用計算生產數量的方式來評估區域產能的大小。而一般而言單位時間的生產數量，可換而言之每個 Lot 平均所花費的時間，故可利用 Lot 生產時間的大小來當做衡量區域產能的指標，由單位時間生產量所換算的平均花費時間，很難看出 Lot 之間互相影響，例如：當兩個 Lot： L_i, L_j ，同時發出搬送命令時，如先搬送 L_i 則造成的 L_j 等待時間增加，反之亦然。

以生產時間作爲區域產能的指標，利用 Lot 在區域的製造時間來作爲評估區域產能，但一個 Lot 並不能代表區域整體的產能狀況，故針對區域產能，應由投入 Lot 進入

區域製造時間的總和來決定，時間越短代表區域產能越高。

Lot 的區域生產時間，由 Lot 自 Stocker 進入區域中後開始計算，Lot 會根據下一製程的機台是否忙碌來決定對應的動作，當機台空閒時，發出搬送命令，呼叫 RGV 將 Lot 搬到機台上，此時產生 Lot 的搬送時間(RMt)，當 Lot 進入機台後，即刻開始生產製造，產生製造時間(RPt)，而當下一製程機台忙碌時，Lot 在 Stocker 中等待，此時產生等待時間(RWt)，Lot 根據上述規則在區域中製造，直到完成區域中的製造流程後，將 Lot 搬回 Stocker 中，轉往下個區域，則在區域中的生產時間如下：

$$Rt = RPt + RMt + RWt \quad \text{.....(3)}$$

Rt：*Lot*進入區域到離開區域總花費時間

RPt：*Lot*在區域中總計的生產時間

RWt：*Lot*在區域中總計的等待時間

RMt：*Lot*在區域中總計的移動時間

在這樣的計算方式下，能加入 Lot 彼此之間的影響，直接反映在生產時間上。

由於區域中同時有許多 Lot 進行生產，Lot 之間彼此互相作用，故需要利用模擬器模擬工廠區域中的生產，使得到所有 Lot 在每各區域內每次進入所花費時間的累計值：

$$\sum_{i=1}^{n}\sum_{j=1}^{v}\left(Rt_j^i\right) \quad \text{.....(4)}$$

Rt：*Lot*進入區域到離開區域總花費時間

n：*Lot*投入的數量

v：每個*Lot*需進入區域的次數

由模擬器可得到$\sum_{i=1}^{n}\sum_{j=1}^{v}\left(Rt_j^i\right)$，在由$\sum_{i=1}^{n}\sum_{j=1}^{v}\left(Rt_j^i\right)$的大小評估產能的大小，值越大表示產能越小。

3.4 問題定義模型

前面三節我們已經描述的模擬器的運作方式、搬送策略及區域產能的定義後，將定義問題如下:

給定一個 LCD 的製程步驟 R 及一個區域 A，希望找到一個搬略 f_{w_1,w_2,w_3,w_4,w_5} 使得

每日投入 *n* 批 Lot 的狀況下，當完成製程步驟 R 時， 使得經過區域 A 的區域產能最大，也就是 $\sum_{i=1}^{n}\sum_{j=1}^{v} Rt_j^i$ 最小 。

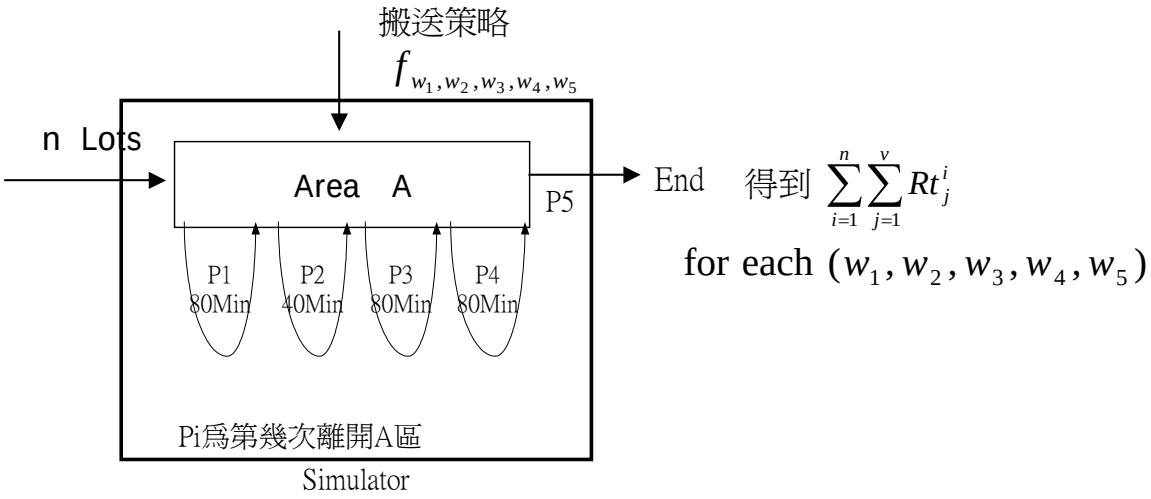

圖 12：問題模型示意圖

根據問題定義，設定模擬器的資料，在每日投入 n 個 Lot 的狀況下，在模擬器中導入不同的搬送策略，利用模擬器計算出 Lot 的整體生產時間 $\sum_{i=1}^{n}\sum_{j=1}^{v} Rt_j^i$ ，由此時間可知不同策略對產能的影響，如上圖。

四、方法論

定義爲 f_{w_1,w_2,w_3,w_4,w_5} 一個搬送策略的函數，在模擬器中每日投入 n 個 Lot，利用模擬器模擬工廠現況，並且不同導入搬送策略，找出導入不同搬送策略的整體區域生產時間。在這樣的定義下，如何找到好的 w_1, w_2, w_3, w_4, w_5 就變的很重要，故導入基因演算法，利用基因演算法針對區域找到一組 w_1, w_2, w_3, w_4, w_5，使得每次能即時找到應執行的搬送命令，使得到較短的區域製造的時間，能到到較好的產能，如下圖，並且將此般送策略導入到區域中，在每次 RGV 執行搬送時，利用此教適合的搬送策略在搬送命令列表中挑選命令，挑選的方式，將搬送命令的屬性帶入搬送策略函數，可計算出一個衡量值，由此衡量值大者決定挑選此搬送命令。

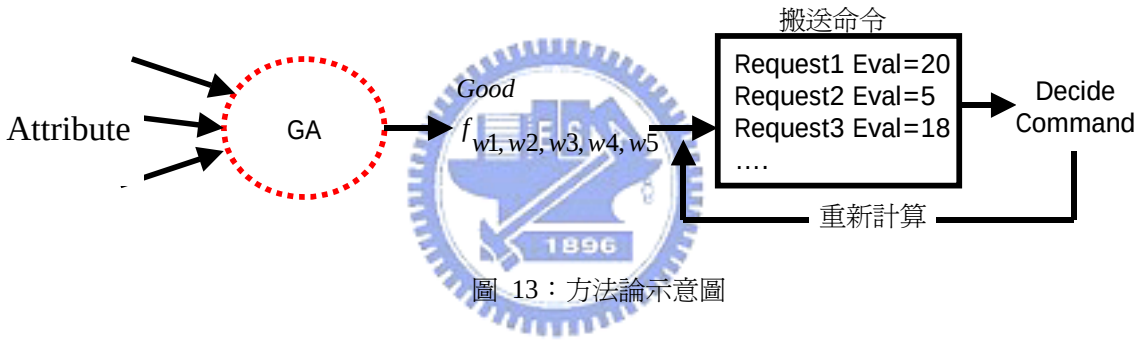

圖 13：方法論示意圖

欲導入基因演算法找到一組好的權重值，若使用傳統的基因演算法，在編碼上將侷限權重值的組合，因此發展物件化編碼基因演算法，來調整傳統基因編碼及交配上的問題。物件化編碼基因演算法與傳統基因演算法最大的不同在於編碼的方式，物件化編碼基因演算法主要始用數值型態的資料結構，更能表現出權重值的特色，其相關的細節如下面章節所述。

4.1 物件化編碼基因演算法

由於每次決定下個命令的時間很短，因爲在 RGV 完成前一筆搬送命令時，並需即時決定下一筆執行的命令，以避免因爲等待選取的命令，而增加生產時間。故事先找出針對區域較適合的策略，由於屬性之間的關係複雜且互相牽制，因此利用基因演算法找出。

另外，在搬送策略中，w_1, w_2, w_3, w_4, w_5 的值得大小代表每個屬性對產能的影響性，故找到區域中較適合的 w_1, w_2, w_3, w_4, w_5 有太多的可能，故希望利用物件化編碼基因演算法來找到，經過不斷的演化找到適合的解。

4.1.1 編碼 Encoding

利用基因演算法，主要是找適合區域的搬送策略，也就是找尋一組適合的 w_1, w_2, w_3, w_4, w_5 ，故將搬送策略視爲一基因，而 w_1, w_2, w_3, w_4, w_5 組成基因，找尋一組針對區域的權重值 。

傳統基因的編碼方面，通常每個基因只有 0 與 1 兩種可能，主要應用在某各屬性是否存在，但對於搬送策略而言，搬送策略是屬性的線性組合，利用一組 w_1, w_2, w_3, w_4, w_5 將其組合起來，故 w_1, w_2, w_3, w_4, w_5 代表爲每個屬性對 $f_{w_1, w_2, w_3, w_4, w_5}$ 的影響，也可說是屬性之間重要度比值，故如單純使用 0 與 1 的方式無法表達其內涵。故在此對於每個基因的應以數值方式呈現，而非傳統的 0 與 1。

搬送策略爲屬性的線性組合，故不同的搬送策略應有不同的權重值 w_1, w_2, w_3, w_4, w_5 ，且爲了讓所有每個屬性互相牽制，定義權重值的總和爲 1 $(w_1 + w_2 + w_3 + w_4 + w_5 = 1)$，則可知其範圍 $0 \le w_1, w_2, w_3, w_4, w_5 \le 1$ ，在這樣的定義下，權重值可互相牽制，並且可防止同一物件化編碼的基因以不同方式呈現，例如：1,1,1,1,1 及 2,2,2,2,2 可說是同一組基因。

利用物件化編碼基因演算法，主要是找適合區域的搬送策略，故將 w_1, w_2, w_3, w_4, w_5 當作基因，找尋最適合 w_1, w_2, w_3, w_4, w_5 ，則其編碼後範例如下圖所示，每個權重值爲小於 1 的小數，且權重值的總和爲 1。

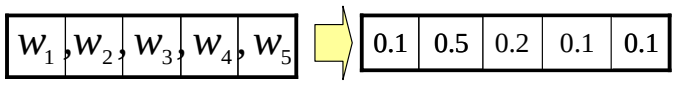

圖 14：物件化編碼基因演算法之編碼結果範例

4.1.2 評估標準 Fitness

不同的搬送策略透過模擬器的執行，可得到區域的整體生產時間，對於提昇產能而

言，就必須縮短生產時間，故可知目標函式希望取得最小的生產時間 $\min(\sum_{i=1}^{n}\sum_{j=1}^{v} Rt_j^i)$。

因此適應函式主要表現出產能的大小，而產能越大生產時間越小，因此針對不同的搬送策略，利用模擬器將得到利用此搬送策略的區域生產時間，則設定適應函式就利用模擬器得到的所有 Lot 整體的區域生產時間計算 $\sum_{i=1}^{n}\sum_{j=1}^{v} Rt_j^i$，但與一般不同的地方爲，希望此生產時間越小越好。

4.1.3 交配 Crossover

傳統的基因演算法在交配這方面，主要使用單點交配或雙點交配的方法，但由於每次在同一個位置切割，這樣的演化效率在某些狀況下並不是很好，因此先前有人提出利用隨機的方法，來決定基因中的每個值是否進行交換，因此在此物件化編碼基因演算法中，採取此方法。

首先在搬送策略的集合中，選取兩組搬送策略，並且隨機產生 5 個布林數值，由此數值來決定兩各基因是否需要做交換，當碰到 1 時則交換，碰到 0 時不變，如下圖所示。

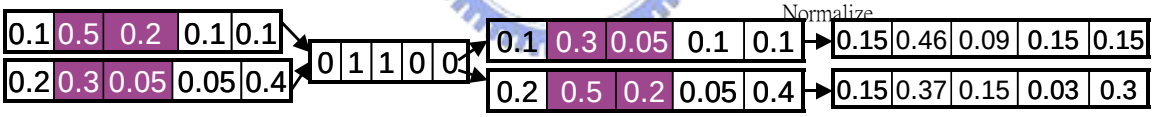

圖 15：物件化編碼基因演算法之兩基因交配產生子代流程圖

當交換完之後，由於不滿足 $w_1 + w_2 + w_3 + w_4 + w_5 = 1$，故需要再做正規化的動作，其方法將其平均到分配到各數值中，即爲 $w_i = w_i / w_1 + w_2 + w_3 + w_4 + w_5$ 其中 $w_i \in w_1, w_2, w_3, w_4$，計算結束前四個基因後，則 $w_5 = 1 - (w_1 + w_2 + w_3 + w_4)$，此方法可避免正規化時小數點的誤差以確保權重值的總和爲 1，結束交配後將搬送策略放回集合中。

4.1.4 突變 Mutation

而在突變方面，主要是爲了防止在初始的時後無法包含所有的可能，以致於造成只得到部份(Local)的最佳化，而非全區域問題的最佳化，故需要使用突變的方法來跳脫目前的組合，傳統的方式利用突變率來控制那些權重需要突變，而由於傳統基因演算法的

編碼使用布林值，只有 0 與 1 的可能，故突變時只需要將 0 與 1 交換即可，但對於目前問題的編碼方式是使用，基因內容採用數值，故針對突變做部份的修改，在發生突變時，應隨機產生一個數值，置換原來的數值，但由於這樣的方式，將使的基因不滿足 $w_1 + w_2 + w_3 + w_4 + w_5 = 1$，故需要在將基因正規化，其留程如下圖所示。

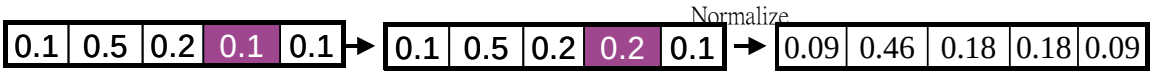

圖 16：物件化編碼基因演算法之基因突變流程圖

五、模擬實驗

模擬實驗的目的，主要利用模擬器及基因演算法找出最適合區域的搬送策略，故首先需將工廠的現況的各種參數設定到模擬器中，此將在 5.1 節討論設定的項目及實驗所使用的值，及基因演算法的數值。而在 5.2 節中，利用模擬器導入不同的搬送策略進行模擬實驗，將得到的實驗結果，進行分析。

5.1 模擬器參數

在模擬實驗開始之前，首先需建置模擬器中各種製造所需的各項參數，主要分四大項目：製造流程、特定區域設定、非特定區域的製造時間、及投入方式等設定。這些參數主要以國內某 LCD 爲例，實際訪談現場的狀況，選定實驗的區域，然後取得實際的內容將其設定到模擬器中，以下將針對各項設定的內容逐一說明。

在建立 LCD 的製造流程之前，首先需瞭解目前工廠區分的區域，如第一章圖一所示，如目前將工廠劃分爲 6 各區域：S、A、B、C、D、E。由製造所經過的機台種類，在這些區域的劃分，即可定義在製造的過程中所需要經過的區域，如下：

Start→S→B(1)→A(1)→C(1)→B(2)→A(2)→D(1)→B(3)→A(3)→E(1)→B(4)→

A(4)→C(2)→B(5)→A(5)→D(2)→End

其中 S 代表的投入，其他則爲製造所經過的區域，前者爲區域代碼，括號內爲製程中第幾次經過此區域。自上述的區域集合，選定 A 區域爲本次實驗的對象，則由製造流程可知每次進去 A 區域的機台順序，如下表。

表 5：A 區的製造流程

進入次數	經過的機台種類與次序
A(1)、A(3)	Ma→Mb
A(2) 、A(5)	Ma→Mb→Mc→Md
A(4)	Ma→Mb→Md

選定 A 區域爲實驗的對象，則 A 區的機台配置如下圖，而區域內相關設定如下表：

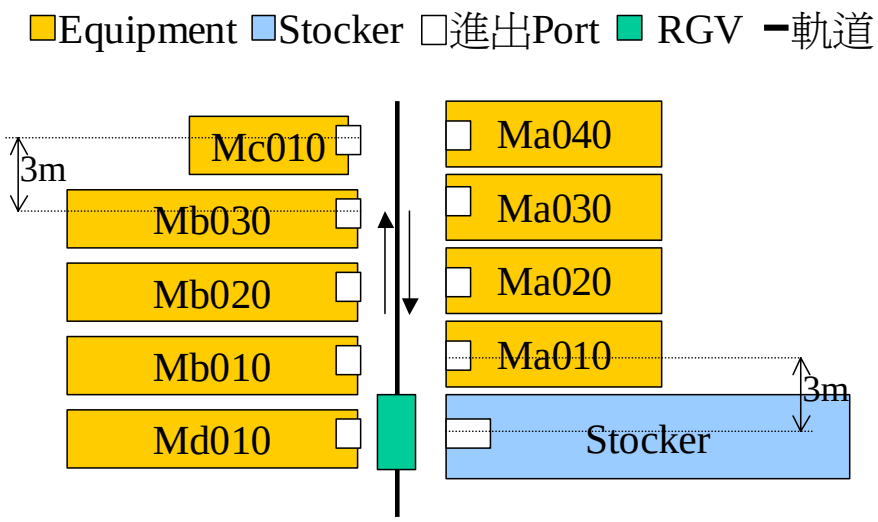

圖 17：區域內機台配置圖

表 6：區域內設備參數表

項目	A 區域的內容
機台的種類	Ma、Mb、Mc、Md
機台的數量	Ma:4{Ma010、Ma020、Ma030、Ma040} Mb:3{Mb010、Mb020、Mb030} Mc:1{Mc010} Md:1{Md010}
機台的處理時間(RPt)	Ma:540s、Mb:400s、Mc:300s、Md:80s
機台之間的距離	3m
RGV 的速度	1 m/s
RGV 搬動 Lot 的時間	8 s/次
Stocker 內部移動時間	40 s/次

在製造流程及區域的設定後，接著要設定非 A 區製造的時間。製造的過程中，共計進入 A 區 5 次，投入後經過一段時間進入 A 區，之後每次離開 A 區後，經果一段時間又回到 A 區，除第 5 次離開後，結束製程。其中每次所應間隔的時間，則利用目前工廠現有的歷史資料，找出重新回到 A 區中間的間隔時間，計算其平均值，則取得的製造流程如下所示：

S→(8200 秒) →A(1)→ (13500 秒)→A(2)→(10400 秒)→A(3)→(13800 秒) →A(4) →(11200 秒)→A(5)

建立以上的基本資訊後，接下來要看目前的工廠的投入狀況，投入主要與三個項目有關：投入機台的數量、投入機台的處理時間、每日投入數。以目前的工廠而言，投入的機台有兩台，生產時間爲每個 Lot 需 20 分鐘(1200 秒)，很容易取得，但針對每日的投入數而言，取得上有很大的困難。原因在於每日的投入數的數量，一般根據工廠每月或每週的生產計劃、客戶的訂單、備料狀態、或是現場狀態……等來決定，且這些項目與投入數的關聯，需要有經驗的專家才能決定，考量的因素眾多，因此無法直接由現有的資料給定一個定值。故針對每日的投入數，我們希望藉由原來的搬送策略 FIFO 取得，我們將在 5.1.1 節中討論並決定每日的投入數。

建立以上製程資料後，接著要建立及產生物件化編碼基因演算法的各項參數。首先設定基因演算法中族群的大小爲 100，隨機產生第一個，之後每代的交配率爲 0.9，而突變率爲 0.05。當每代的最小區域生產時間連續 50 代未改變時，則判斷基因演算法收斂，求得較佳解。

在開始實驗時，由於模擬器中未有 Lot 開始生產，區域的機台閒置，Lot 不需等待可直接送入機台，則發現機台開始的平均使用率偏低，Lot 的處理時間較短。而工廠實際狀況應是不斷的生產，與實際工廠的情況不合，並且如將初始的機台使用率及 Lot 處理時間，併入計算，則由於開始的 Lot 佔所有 Lot 的比例過高而影響，實驗的正確性，故對於實驗結果的計算，只採用第二天投入的 Lot 區域生產時間，做爲區域產能的指標，當可計算出區域產能時，也就是完成第二天投入 Lot 的製造時，結束此搬送策略的模擬實驗。

5.1.1FIFO 的結果

投入數的高低將影響機台的使用率及 Lot 生產時間，當投入數低時，則發現機台的使用率低，造成生產率低落，而投入數太高時，則機台數量過少，造成每個 Lot 等待機台的處理時間過長。在實際的工廠中，一般不希望有過長的等待時間，原因有二，一爲某些製成須在時間內完成，例如成膜後到剝膜，如時間過久，則無法施行，造成重工的問題，另外一個與良率有關，當長時間的等待，其被污染的機率變高，而造成良率的下

降，此問題已在之前被提出探討[1]。因此投入數太高或太低都不符合工廠的需求。

由於實際在工廠中，投入數的決定是參考各種不同的狀況而決定投入的數量，但在模擬器中，並沒有專家根據實際的需求給定投入數，因此我們在實驗中希望給定一個每日固定的投入數來模擬生產。而決定次每日投入數的方法，希望利用原有的搬送策略 FIFO 找到最適當的投入數，表示在機台的使用率及搬送效能方面都有不錯的結果，意即利用 FIFO 就有不錯的結果。將得到的投入數給定到模擬器中，利用基因演算法找到更好的搬送策略。以下爲在不同投入數的狀況下，利用 FIFO 所做的分析比較。

利用 FIFO 的搬送測略，則搬送策略函數爲：

$$f_{w_1,w_2,w_3,w_4,w_5} = 0 \times Att_{From} - 0 \times Att_{To} - 0 \times Att_{RGV} + 1 \times Att_{LOT} - 0 \times Att_{\mathrm{STK}} \quad \text{.....(5)}$$
$$w_1 = 0, w_2 = 0, w_3 = 0, w_4 = 1, w_5 = 0$$

然後每日投入不同的數量的 Lot，利用模擬器計算第二天的 Lot 區域生產時間的總和及機台使用率，希望找到投入數高、機台使用率高，且整體的生產時間少。

首先討論整體的區域生產時間(以下簡稱生產時間)如下圖 18，爲了避免模擬初始產生的邊際效應，故生產時間的取樣爲第二天投入 Lot 的整體生產時間的總和，而由於每日的投入數的不同，平均生產時間爲整體生產時間除已投入數。由圖中標號 1 可發現當投入數很少的時候，生產時間皆很短，當投入量增加到一定的程度後，生產時間會大量的提昇，因此但我們希望投入數越多但生產時間越短，則必須取圖形中，每次轉折前的那一個點，如標號二，分別指出三各轉折爲投入數 62、69、及 74，在此三各投入量時，有較高的投入數但生產時間與少一個 Lot 時不會增加過多，故可說再次投入數時 FIFO 有較佳的表現。

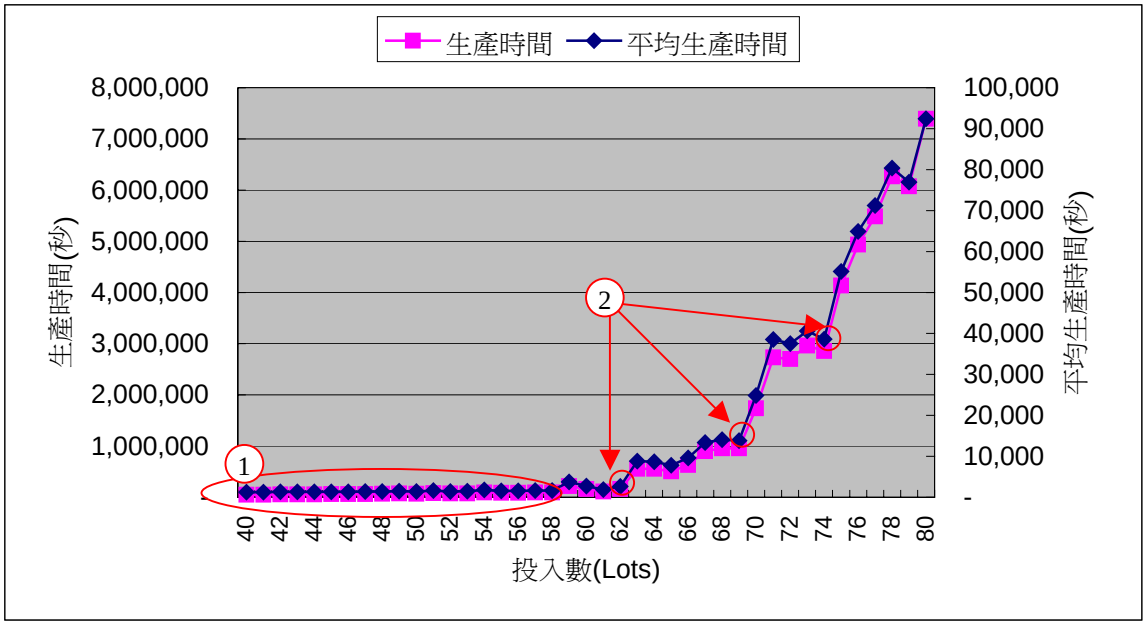

圖 18：在 FIFO 搬送策略下，投入數與生產時間分析圖

分析生產時間主要是由三種時間組成：等待搬送時間、等待機台時間、搬送時間。由上圖我們可以發現，當投入量過大時，生產時間將以倍數成長，進而往下分析，爲何在投入量大時，產生較大的生產時間？則如下圖所示，爲總時間與三個項目的時間的比較圖，可以發現當投入量過大於時，則等待機台的時間會隨著投入的數量大量增加，而造成生產時間過長。對於等待搬送時間也隨著投入數增加而增加，但與總生產時間與等待機台時間的相較，其只是些微的增加，究其原因在於當投入數增加則相對發出搬運命令的次數較頻繁，其同時發出的機率較高，相對容易產生等待搬送時間，而對於搬送時間與投入數，則相對於總生產時間的影響較不深，故由此可知，等待機台的時間將影響總生產時間，而等待機台的時間由於機台數量不足所引起，故在投入的規劃上，會考量區域所能容納的最高產能來進行投入的計算，而非無限制的投入。

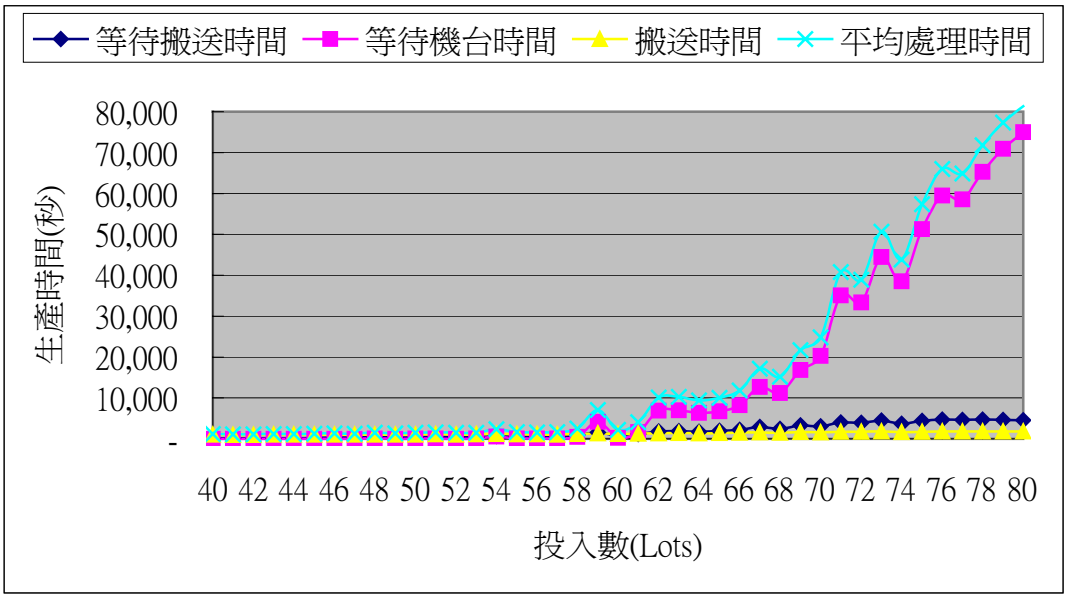

圖 19：在 FIFO 搬送策略下，投入數與各細項生產時間的關係圖

根據上述，我們由生產時間得到三各不同的投入數：62、69、74，因此我們希望重中討論何者爲較適合的投入，此時考慮機台使用率，如下圖所示，三條紅線落於此三個不同的投入數上。

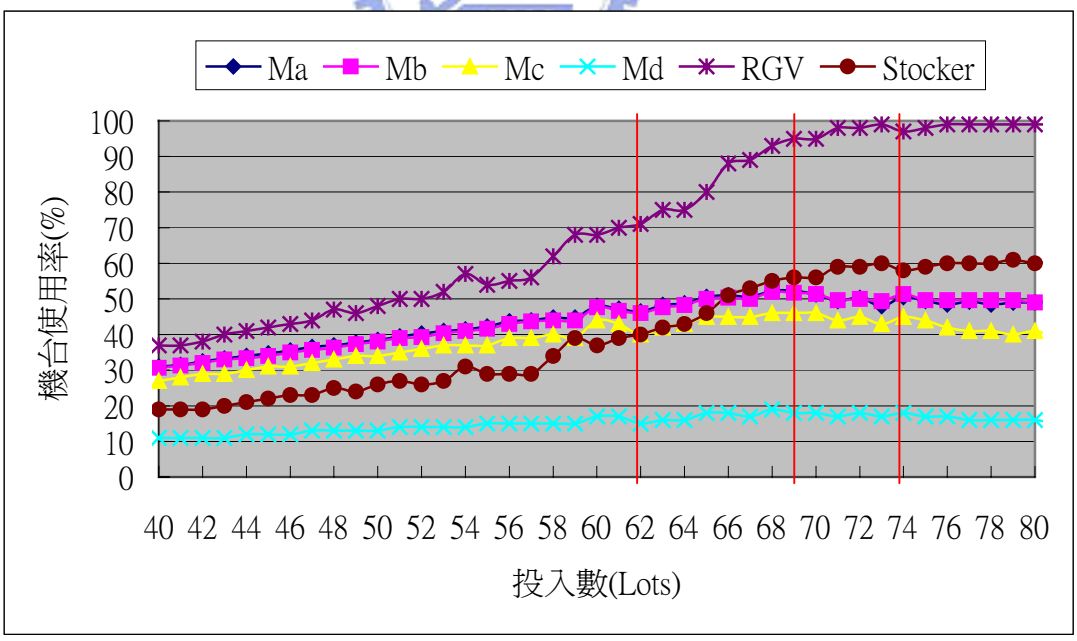

圖 20：在 FIFO 搬送策略下，投入數與機台使用率的關係圖

在區域中主要有四種不同的製程機台、RGV、及 STOCKER，分別討論其使用率。首先針對製程機台的部份，在區域中共有四種機台：Ma、Mb、Mc、Md，四個機台的

使用率中，可發現 Ma 及 Mb 無論在多少投入數時，都有較高的機台使用率，表示此兩者應該爲區域中的瓶頸機台，故以此兩者的使用率爲主要的考量，故由圖中可明顯發現，在投入數 62 時，機台的使用率並非最高，而在投入數 69 及 74 時的機台使用率較高，相對也表示此時區域中不因 Lot 數不足，而產生的機台閒置的現象。當投入數低時，機台使用率低，表機台的產能未能充分的利用，但在每日投入數 69 之後，機台的使用率未能持續提昇，反而有時會下降，表示以已達到區域機台最大的產能，再增加投入數已無法提昇機台使用率。

另外在區域中的 RGV 及 Stocker 的使用率，發現 RGV 的使用率，當投入數高時幾乎接近 100%，因此我們針對 RGV 來討論。RGV 的使用率，與搬送策略有關，好的策略能增加搬送效能，降低 RGV 的使用率，我們希望找到在 FIFO 有比較好結果的投入數。由於前面的探討，目前針對投入數 69 與 74，發現投入數 69 比投入數 74 中，有較高的機台使用率且有較低的 RGV 使用率。根據以上的考量，因此我們選擇每日投入 69Lot，來進行模擬實驗。

5.2 實驗結果分析

當決定以上的參數後，開始進行模擬實驗，利用物件化編碼基因演算法找出最適合區域的搬送策略。以下爲實驗的結果與分析，其中 5.2.1 章爲每日投入數 69Lots 的實驗結果分析，而 5.2.2 節中則是利用不同投入數的結果做比較。

5.2.1 固定投入數的結果分析

根據上一節設定的參數，利用物件化編碼基因演算法找出最適合區域的搬送策略。在實驗結果中，首先討論基因演算法的結果，如下圖所示，前一張圖爲基因演算法的代數與每代的搬送策略所產生的生產時間的平均值，後一張圖爲代數與每代的最小生產時間的推移圖。由圖中可發現，在平均生產時間中，發現其初始時，生產的平均時間很大，在前五代時急速下降，之後就於在 500,000 秒之間來回移動，逐漸收斂。而對代數與最小時間之間，也有相同的情行，在前五代時快速下降，之後到 15 代時就到最小值，測

試 50 代以後不再變動，表示其收斂。由結果可發現，在解此問題中利用物件化編碼基因演算法，能快速的找到適合的解。

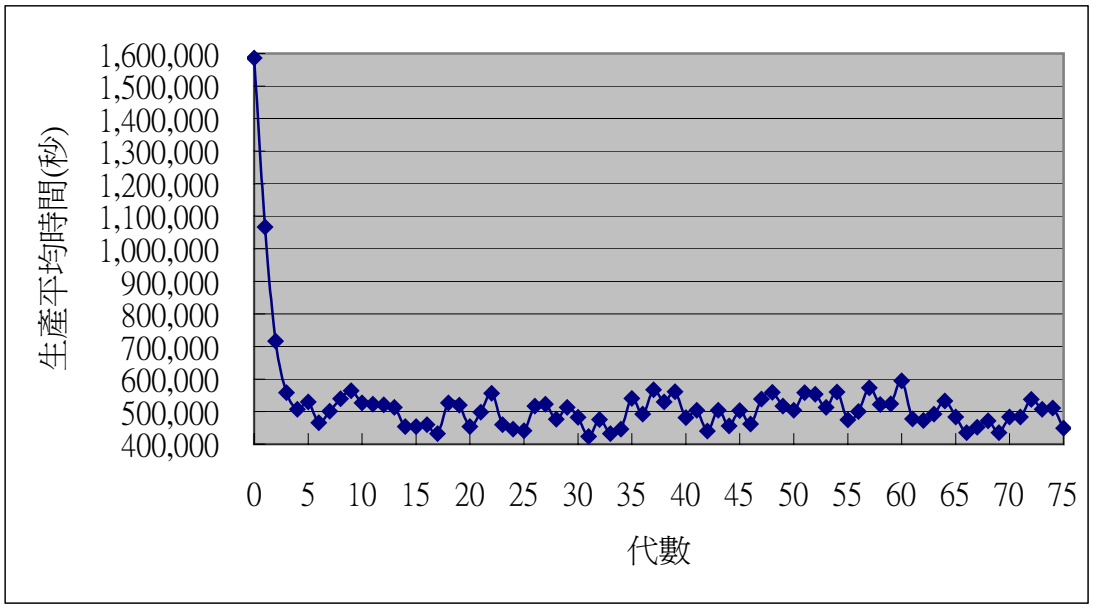

圖 21：利用物件化編碼基因演算法，代數與每代生產時間平均值關係圖

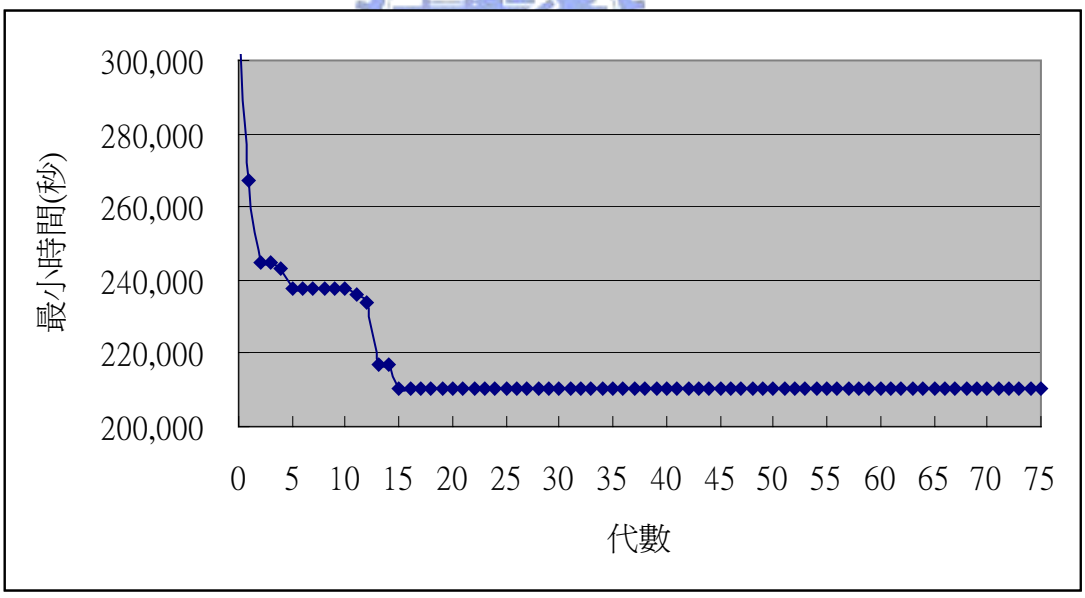

圖 22：利用物件化編碼基因演算法，代數與每代最小生產時間關係圖

接著針對找到的區域的搬送策略進行分析，由基因演算法找到的搬送策略函數如下：

$$f_{w_1,w_2,w_3,w_4,w_5} = 0.064456 \times Att_{From} - 0.415093 \times Att_{To} - 0.185991 \times Att_{RGV} + 0.039848 \times Att_{LOT} - 0.294612 \times Att_{\mathrm{STK}} \quad \text{.....(6)}$$

$w_1 = 0.064456,$

$w_2 = 0.415093,$

$w_3 = 0.185991,$

$w_4 = 0.039848,$

$w_5 = 0.294612$

由訓練的結果發現權重值的關連如上面所示，其中最爲重要的爲 w_2 ，依次爲 w_5, w_3, w_1, w_4 ，其中 w_2 佔整體比例的 41%及 w_5 佔 29%，此兩者的重要性包含合整體的 70%。以下分別對應之前搬送策略的函數說明，逐一探討每個權重值。

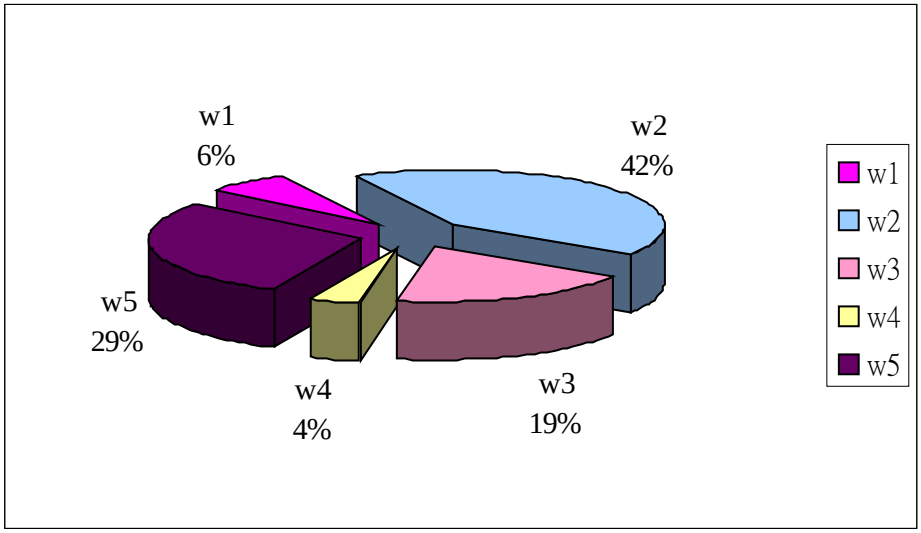

圖 23：較佳搬送策略，權重值分布圖

w_2 的權重值代表爲下一製程機台的狀態，當下一製程越忙碌則此命令的優先權越低。探討其原因，可發現在下一機台忙碌時，則讓命令先等待，則可增加搬送的直送率，減少搬送次數及 Stocker 的使用率。故我們進行直送率的分析，在區域中每個 Lot 最多可直送的次數爲 10 次(可由製造流程中計算出)，故投入 69 個 Lot 總計的直送次數爲 690 次，根據實驗中的統計，發現其中有 598 次形成直送，則直送的比例爲 86.7%。由此高的直送率，可解釋當 w_2 高的時候可增加直送率進而減少 Lot 生產時間。

而 w_5 代表 Stocker 的等待的時間，此權重值高表示不希望在 Stocker 中花費過多的等帶時間，以至於讓不需等待的命令由於 Stocker 等待的時間過久而造成搬送等待的時

間加長，故如搬送命令需要等待 Stocker，則較不易被選中優先處理。在這樣的狀況下，如搬送命令產生的狀況很頻繁的時候，有可能會造成餓死的問題(需佔用 Stocker 資源的命令，一直選不到)，因此需要用等待時間來平衡，以搬送策略函數中指的爲 w_4 的權重值，當等待時間過越久就越容易被優先選到，而由 w_4 及 w_5 的比例可發現約有 7 倍的差異，表當 Stocker 等待時間越大，則等待時間越久，其數據在正規化後仍有 7 倍的差距。

另外 w_3 表示 RGV 的使用度，此值高表此區域重視 RGV 的搬送的距離，探討其原因，可發現在 FIFO 策略投入 69 個 Lot 中，RGV 的使用率高達 95%，RGV 一直處於忙碌的狀況，因此在策略上希望減低 RGV 的忙碌程度，故重視每筆命令搬送的遠近，相對此權重值也較高。

除上述的搬送策略的分析以外，將機台的使用率列出，如下表，與 FIFO 相較之下，明顯發現瓶頸機台的使用率有明顯的上升，而 RGV 及 Stocker 的使用率有明顯的下降，由其以 RGV 的使用率較明顯，此與研究動機中收集的資訊：產能與機台使用率成正比、RGV 使用率成反比、Stocker 使用率成反比符合。

表 7：最佳搬送策略下，機台使用率列表

機台別	每日機台使用率(%)	每日處理數(Lots)	備註
Ma010	65	104	Ma 平均使用率 54.5%，平均處理數 87.75Lots
Ma020	60	97	
Ma030	53	86	
Ma040	40	64	
Mb010	40	87	Mb 平均使用率 54.3%，平均處理數 118Lots
Mb020	57	124	
Mb030	66	143	
Mc010	48	141	
Md010	19	215	
RGV	78	1536	
Stocker	43	949	

分析第二天投入 Lot 的生產時間，將取得的最佳策略與 FIFO 相較，如下表。

表 8：最佳策略與 FIFO 生產時間比較表

時間種類(秒)	最佳策略		FIFO	
	所有 Lot	一個 Lot	所有 Lot	一個 Lot
製程時間	382260	5540	382260	5540
搬送時間(M)	56982	824.5	111517	1616.2
搬送等待時間(T)	80993	1173.8	224455	3253
等待機台時間(W)	72437	1049.8	1161828	16838.1
M+T+W	210412	3049.4	1497800	21707.2

由上表發現導入此新的搬送策略比原有的 FIFO，在生產時間上有顯著的進步，時間減少爲原來的 14%(210412/1497800)，也就是減少了 86%的搬送時間，效果極爲明顯。如以時間的觀點來看，每各 Lot 在搬送效能方面可減少 18657.8 秒(21707.2-3049.4=18657.8)，以每個 Lot 得生產時間 8589.4 秒(製程時間+M+T+W)，則原來可做一個 Lot 的時間，導入新的策略後可處理 2.5Lot。

由於搬送時間、搬送等待時間、等機台時間會互相影響，當搬送時間變短，則同時在命令列表中，命令的等待時間也會變短，且其他等待機台的 Lot 也可減少等待時間，而好的搬送策略，可使這三各時間同時減少。根據以上的數字，可以發現此區域的特性，非常重視搬送的效能，且在最好的狀況下，搬送時間、搬送等待時間、等機台時間的總和佔整各 Lot 的處理時間的 35%，故由實驗可知，好的搬送策略可提昇產能。

5.2.2 不同投入數的結果比較

由上面投入 69 各 Lot 的狀況下，檢視其機台使用率的狀況，發現利用好的搬送策略後，則有效下降 RGV 的使用率，其表示可在區域中多投入 Lot，因此針對不同的投入數，利用基因演算法，模擬 20 代產生的結果，其最小的時間分不如下圖 24：

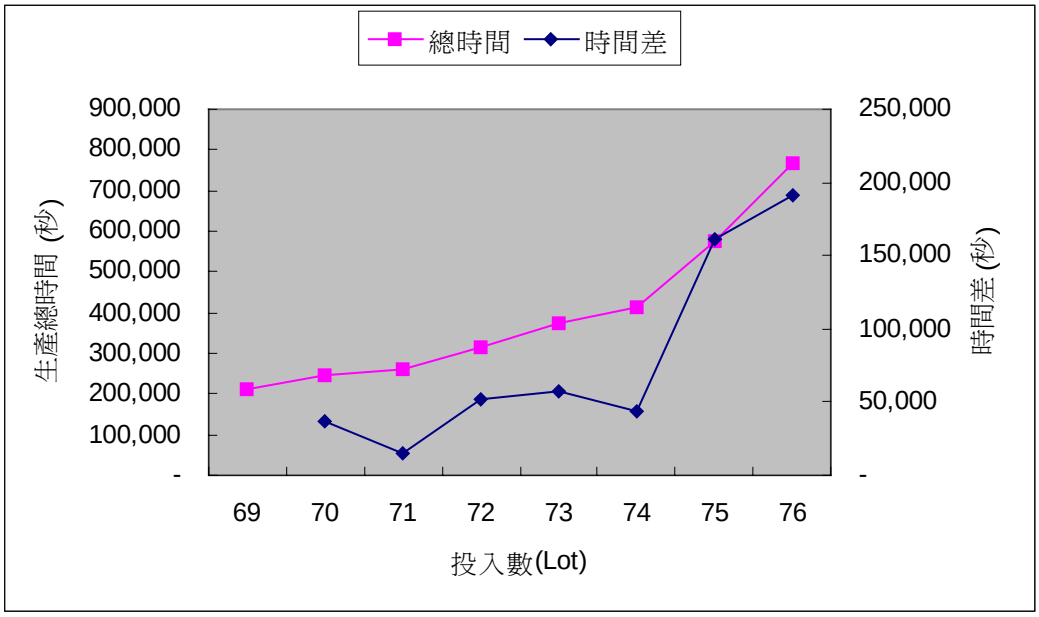

圖 24：不同投入數與最短生產時間關係圖

由圖中可發現每多投入一個 Lot，則總體的時間會些微的增加，其中投入數 69 到 74 之間較平緩，表增加的時間不大，而在投入 74 個 Lot 以後發現時間明顯的增加。而另外一折線表每多投一個 Lot，所需增加的時間分析，會發現在投入數 71 與 74 中有較不錯的結果，原因在於自 70 增加一個 Lot 所增加的時間較小，意即須付出的代價較小，73 到 74 也有相同的情形。

而在不同的投入數中，所得到的權重值的列表如下所示，可發現投入數在 69~76 之間，得到的權重值， w_2, w_3, w_5 都較高而 w_4 都較低，表示無論在不同投入數下，都重視下一製程機台的狀態、RGV 搬送的效能及 Stocker 的等待時間，而較不重視命令的等待時間。

表 9：不同投入數利用物件是編碼基因演算法演化 20 代，得到目前最佳的搬送策略權重值列表

投入數	w_1	w_2	w_3	w_4	w_5
69	0.06445612	0.41509296	0.18599089	0.03984787	0.29461216
70	0.06038007	0.31115731	0.30253812	0.00658138	0.31934312
71	0.04966362	0.32166434	0.27990352	0.05367062	0.2950979
72	0.17215936	0.38077056	0.21027838	0.01536714	0.22142456
73	0.20122572	0.26317331	0.25464451	0.02167964	0.25927682
74	0.09041248	0.34483012	0.27230724	0.01093738	0.28151278

75	0.05364603	0.52211761	0.19780754	0.0384623	0.18796652
76	0.03330901	0.23344564	0.34335479	0.00933156	0.380559

由此結論可發現區域某些現狀，在w_2, w_3, w_5分別代表機台直送、RGV 搬送、Stocker 的等待，這些的權重較高，表示在區域中較薄弱的部份，故需要優先考量，如果可以解決這些問題，相信更能提高產能。

六、結論與展望

6.1 結論

本研究主要是在 TFT-LCD Array Fab 中，利用 RGV 搬送的模式下，進行提昇產能的研究。由於 LCD 工廠內部複雜且本身具有區域特性，區域之間幾乎可以獨立運行，故假設其針對區域提昇產能後可間接提昇整個工廠的產能，所以提出針對 TFT-LCD Array Fab 中的區域來提升產能。針對區域，利用好的搬送策略來增加產能，故希望針對特定區域，事先找到一個較適合區域特性的搬送策略，再將此搬送策略導入到區域中，以達到增加產能的目的。

因此，針對區域提出相關的假設，提出針對區域的模型，並將其建置爲模擬器，量測生產時間。提出搬送策略爲搬送命令屬性的線性組合，及區域產能的指標由區域總生產時間的長短決定，時間越短表產能越高。由於搬送策略的可能性眾多，希望利用基因演算法取得較適合的搬送策略。並且由於傳統的基因演算法在編碼上的不足，提出利用物件化編碼基因演算法，找出針對區域較適合的搬送策略，接著進行模擬實驗。

而模擬實驗的結果中，先給定模擬器詳細的製造流程及區域的配置，再利用物件化編碼基因演算法，找出針對此區域比較適合的搬送策略，由實驗的結果發現利用物件化編碼基因演算法能快速的找到較適合的搬送策略，且不同的搬送策略對於區域生產的時間影響深遠。而且根據不同的區域特性，其搬送策略的影響大小也不同，主要與機台的生產時間有關，機台生產時間越長，則搬送的狀況較少，則搬送策略所能節省的時間也有限，反之，如機台生產的時間較短，則搬送較頻繁，相對於使用好的搬送策略，能明顯的增加區域產能。

利用基因演算法能事先針對不同的區域特性取得較適當的搬送策略。適當的搬送策略期能提昇搬運績效，進而提高機台使用率，這樣才能考量長久的搬送效能，並非利用一段時間最佳不考慮未來的影響。

另外由實驗的結果中發現，在選定的實驗區域中，比較適合的搬送策略，在下一製

程機台、Stocker 的等待時間、及 RGV 的距離方面，有較高的權重值，表示區域中重視這三項目的效能，如果能提高這三者的效能，相對也能提高區域的產能，由此較適合區域的搬送策略，也能發現出區域較須改善的地方。

6.2 展望

對於工廠生產而言，總希望利用現有的資源，能得到較高的產能，而在現有的狀況下，設施規劃已定的情形下，針對區域找到較適合的策略，能有效提昇區域產能，但仍有許多未能考慮，歸納下列幾點建議作爲未來研究的參考：

1. 搬送命令的屬性或許可在拆解，或有其他未知的屬性值未探討，可再加以研究。
2. 搬送策略或許有其他的組合方式。
3. 本研究每次只討論其中一個區域，非討論區域則使用歷史資料的平均值，對於此均值的大小與區域及搬送策略的關係，並未加以討論，或許有更適合的假設。
4. 在本研究中主要以現行的 LCD 廠爲例，其工廠內部的規劃已完成，但其佈置規劃及其投入的數量，也對於產能有所影響，可再加以研究。
5. 本研究只針對區域，利用生產時間討論其區域產能的好壞，對於區域及區域之間的交互做作用，並沒有加以討論其中的關連性。
6. 在此可加入工廠排程及現場派工的因素，更能針對實際的工廠。
7. 本研究只簡單討論投入數與生產時間的關聯，但投入數實際對工廠的影響甚遠，值得再加以研究。
8. 針對搬送系統而言，本研究只針對搬送命令的執行順序探討，實際上仍有其他針對搬送系統的調整方式。

參考文獻

[1] Cheng, T. C. E., “A Simulation Study of Automated Guided Vehicle Dispatching”, Robotics and Computer-Integrated Manufacturing, Vol.3, No.3, pp.335-338, 1987.

[2] Chi-Chung Lio, An Intelligent Manufacturing Defect Detection Method for Time Issue, National Chiao Tung University Computer and Information Science,2003

[3] E. GOLDBERG DAVID, GENETIC ALGORITHMS IN SEARCH, OPTIMIZATION, AND MACHINE LEARNING, READING, MASS. : ADDISON-WESLEY PUB. CO., 1989.

[4] E. Rolland, “Abstract Heuristic Search Method for Genetic Algorithm” Ph.D dissertation, The Ohio State University Columbus, 1991.

[5] Egbelu, P. J., “Pull Versus Push Strategy for Automated Guided Vehicle Load Movement in a Batch Manufacturing System”, Journal of Manufacturing System, Vol.6, No.3, pp.271-280, 1987.

[6] Gen, Mitsuo and Cheng, Runwei, Genetic algorithms and engineering design, New York : Wiley, 1997.

[7] H. Pirkul and E. Rolland, “A lagrangian based heurisctic for graph partitioning.” Working paper, University of California. Riverside, California, 1991.

[8] J. Craig Potts, Terri D. Giddens, and Surya B. Yadav, “The Development and Evaluation of an Improved Genetic Algorithm Based on Migration and Artificial Selection,” IEEE Transactions on System, Man. and Cybernetics, Vol.24, No.1, pp.73-86, January 1994.

[9] J. T. Rhapsodic, M.R. Palmer, G. E. Liepins, and M. Hilliard, “Some guidelines for genetic algorithms with penalty functions,” In Proceedings of the 3rd international Conference on Genetic Algorithms (Edited by J. D. Schaffer).

[10] L. Davis (ed), Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.

[11] Lin, J. T., Wang, F. K. and Yen, P. Y., “Simulation Analysis of Dispatching Rules for an Automated Interbay Material Handling System in Wafer Fab”, Internal journal of Production

Research, Vol.39, No.6, pp.1221-1238, 2001.

[12] Liu, C. M., and Duh, S. H., “Study of AGVS Design and Dispatching Rules by Analytical and Simulation Methods”, International Journal of Computer Intergrated Manufacturing, pp.290-299, 1992.

[13] Richardson, J. T. M. R. Palmer, G. L. Liepins, and M. Hilliard, (1989) “Some guidelines for genetic algorithm with penalty functions.” In Proceedings of the 3rd international Conference on Genetic Algorithms (Edited by J. D. Schaffer). George Mason University, 1989.

[14] Ting-Yu Chen and Chung-Jei Chen, “Improvements of Simple Genetic Algorithm in Structural Design,” International Journal for Numerical Methods in Engineering, Vol.40, 5(4&5), pp.1323-1334,1997.

[15] Yim, D. S. and Linn, R. J. “Push and Pull Rules for Dispatching Automated Guided Vehicles in a Flexible Manufacturing System”, Internal journal of Production Research, Vol.31, No.1, pp.43-57, 1993.

[16] 吳俊寬，“晶圓廠連接式自動化物料搬運系統搬運策略之模擬研究”，清華大學工業工程與工程管理所碩士論文，2001。

[17] 周上傑，“晶圓廠自動化物料搬運系統之派車模擬研究”，清華大學工業工程研究所碩士論文，1999。

[18] 張原銘, 晶圓廠之自動化物料搬運系統模擬研究—以黃光區與爐管區爲例,國立清華大學工業工程與工程管理學系碩士論文,2003

[19] 陳紹偉，“12 吋 IC 代工廠自動化物料搬運系統之系統模擬與派工法則的研究”，台灣大學機械工程研究所碩士論文，1999。

[20] 楊景如，“晶圓廠自動化物料搬運系統之搬運車運作策略模擬研究”， 清華大學工業工程與工程管理所碩士論文，2002。

附錄一 模擬 FIFO 的結果

投入數	等待搬送時間(s)	等待機台時間(s)	搬送時間(s)	平均處理時間(s)	生產時間(s)	平均生產時間(s)
40	174.75	12.1	997.35	1184.2	51184	43.22242864
41	140.0487805	4.024390244	974.5365854	1118.609756	52088	46.56494342
42	131.3095238	0.761904762	976.0238095	1108.095238	55473	50.06158144
43	162.0697674	2.279069767	992.9069767	1157.255814	55651	48.08876251
44	193.6363636	4.113636364	993.5227273	1191.272727	56999	47.84714591
45	236.0444444	19.22222222	1010.266667	1265.533333	59542	47.04893852
46	285.3043478	26.30434783	1024.73913	1336.347826	62496	46.76626757
47	276.6808511	9.106382979	1003.212766	1289	64930	50.37238169
48	356.5625	45.33333333	1051.041667	1452.9375	69220	47.6414161
49	296.9387755	28.69387755	1019.918367	1345.55102	72887	54.16888613
50	375.36	49.34	1055.48	1480.18	71358	48.20900161
51	437.7647059	77.76470588	1074.529412	1590.058824	82431	51.8414783
52	387.9230769	30.28846154	1047.865385	1466.076923	75097	51.2230967
53	380.6981132	52.20754717	1061.849057	1494.754717	77856	52.08613769
54	665.4259259	563.7592593	1151.388889	2380.574074	95811	40.2470148
55	481.6363636	81.50909091	1070.672727	1633.818182	85580	52.38036946
56	457.0178571	43.125	1064.803571	1564.946429	88621	56.62877552
57	489.4210526	79.43859649	1048.210526	1617.070175	92705	57.32899005
58	902.3275862	453.4827586	1180.155172	2535.965517	94100	37.10618278
59	1553.457627	4150.966102	1319.508475	7023.932203	221368	31.51624953
60	816.5	213.8333333	1142.333333	2172.666667	163751	75.36867137
61	1148.786885	1659.327869	1225.065574	4033.180328	112452	27.88171886
62	1798.645161	6921.33871	1366.774194	10086.75806	164668	16.32516602
63	1910.873016	6967.507937	1370.555556	10248.93651	555096	54.16132684
64	1727.765625	6304.9375	1322.828125	9355.53125	554434	59.26269553
65	1924.446154	6736.292308	1389.492308	10050.23077	508122	50.55824206
66	2178.818182	8205.80303	1454.560606	11839.18182	633805	53.53452711
67	2915.044776	12675.76119	1577.671642	17168.47761	897526	52.2775531
68	2455.661765	11206.07353	1509.838235	15171.57353	954934	62.94231763
69	3252.971014	16838.08696	1616.188406	21707.24638	955578	44.02115236
70	3018.228571	20264.9	1565.685714	24848.81429	1739190	69.99086476
71	4005.56338	35039.85915	1705.957746	40751.38028	2733474	67.07684454
72	3878.611111	33314.44444	1696.541667	38889.59722	2697624	69.36621083
73	4446.780822	44451.0137	1782.150685	50679.94521	2963869	58.4820877
74	3672.256757	38483.44595	1674.121622	43829.82432	2856211	65.16592398
75	4354.626667	51301.52	1754.04	57410.18667	4135304	72.03084052
76	4686.723684	59505.85526	1792	65984.57895	4936029	74.80579673
77	4580.532468	58538.54545	1764.454545	64883.53247	5484402	84.52687133
78	4681.717949	65296.66667	1769.589744	71747.97436	6267866	87.35948375
79	4617.088608	70933.25316	1780.974684	77331.31646	6076677	78.57976921
80	4565.9	75027.825	1774.05	81367.775	7396544	90.90262085

投入數與機台使用率

投入數	Ma機台	Mb機台	Mc機台	Md機台	RGV車	Stocker
40	31	30.66667	27	11	37	19
41	31.5	31.33333	28	11	37	19
42	32.5	32	29	11	38	19
43	33.25	33	29	11	40	20
44	34	33.33333	30	12	41	21
45	34.75	34	31	12	42	22
46	35.5	35	31	12	43	23
47	36.5	35.66667	32	13	44	23
48	37	36.33333	33	13	47	25
49	37.75	37.33333	34	13	46	24
50	38.5	38	34	13	48	26
51	39.5	39	35	14	50	27
52	40.25	39.33333	36	14	50	26
53	41	40.33333	37	14	52	27
54	41.5	41	37	14	57	31
55	42.25	41.66667	37	15	54	29
56	43.75	43	39	15	55	29
57	44	43.66667	39	15	56	29
58	44.75	44	40	15	62	34
59	44.75	44	39	15	68	39
60	48	47.66667	44	17	68	37
61	47.25	46.66667	43	17	70	39
62	46.5	46	40	15	71	40
63	48.25	47.66667	42	16	75	42
64	48.75	48.33333	43	16	75	43
65	50.5	50	45	18	80	46
66	51	50.33333	45	18	88	51
67	50.5	50	45	17	89	53
68	52.25	52	46	19	93	55
69	52.25	51.66667	46	18	95	56
70	51.25	51.33333	46	18	95	56
71	49.5	49.66667	44	17	98	59
72	50.25	50	45	18	98	59
73	48	49.33333	43	17	99	60
74	50.5	51.33333	45	18	97	58
75	49.5	49.66667	44	17	98	59
76	48.5	49.66667	42	17	99	60
77	49.25	49.66667	41	16	99	60
78	48.5	49.66667	41	16	99	60
79	49	49.66667	40	16	99	61
80	49	49	41	16	99	60

附錄二 物件化編碼基因演算法的結果

代數	生產時間-最小值(s)	生產時間-最大值(s)	生產時間-平均值(s)
0	313605	3771663	1586181
1	267381	2994424	1066807
2	244796	2613316	716472
3	244796	2628215	558301
4	242968	2620620	507176
5	237921	2003582	529532
6	237921	2006458	466120
7	237921	2659415	501696
8	237921	2522841	539991
9	237921	2372503	564545
10	237921	2433150	526793
11	236132	2455878	523345
12	233773	2463699	520818
13	217190	2914941	512801
14	217190	2631960	454298
15	210412	1982060	453907
16	210412	2534797	459819
17	210412	1466071	432513
18	210412	2541926	527216
19	210412	2254288	520166
20	210412	2489011	453689
21	210412	2655470	498647
22	210412	2998631	556804
23	210412	2404758	459809
24	210412	2200093	446687
25	210412	2425764	441525
26	210412	2604245	517404
27	210412	2498579	522907
28	210412	2289719	476797
29	210412	2724668	512982
30	210412	2896586	482254
31	210412	2507211	423278
32	210412	2321207	475670
33	210412	1924361	432476
34	210412	2328842	446584
35	210412	2905252	540607

36	210412	2129351	492433
37	210412	2683372	567188
38	210412	1966922	529837
39	210412	3244302	561048
40	210412	2450647	481742
41	210412	3450528	504579
42	210412	2461366	440648
43	210412	2786371	503853
44	210412	2096938	456163
45	210412	2233935	503471
46	210412	2564510	462017
47	210412	2460006	538927
48	210412	3130443	559435
49	210412	2376779	517144
50	210412	1662296	504383
51	210412	2848101	558025
52	210412	2823007	553058
53	210412	2498103	513531
54	210412	2833728	560053
55	210412	2026051	474504
56	210412	2322997	500225
57	210412	2975897	573328
58	210412	2351114	522250
59	210412	2338195	524263
60	210412	3470810	595168
61	210412	2096987	477658
62	210412	1794922	472615
63	210412	2150949	492027
64	210412	2742127	532705
65	210412	2157018	483591

附錄三 在投入數 69 下搬送策略與生產時間

投入數 69 最後一代的結果

	w1	w2	w3	w4	w5	生產時間(秒)
1	0.06445612	0.41509296	0.18599089	0.03984787	0.29461216	210412
2	0.06445612	0.41509296	0.18599089	0.03984787	0.29461216	210412
3	0.06445612	0.41509296	0.18599089	0.03984787	0.29461216	210412
4	0.05892711	0.35094898	0.23854588	0.03720688	0.31437115	251284
5	0.05371655	0.46783222	0.17188417	0.03121716	0.2753499	251711
6	0.06856809	0.37759566	0.20346786	0.03567737	0.31469102	253489
7	0.06470749	0.35651507	0.20857624	0.04033714	0.32986406	261380
8	0.06541539	0.39379858	0.23590019	0.03149023	0.27339561	265225
9	0.15113291	0.30089686	0.18342663	0.02548995	0.33905365	265695
10	0.10795556	0.37781927	0.23031851	0.0320063	0.25190036	272002
11	0.09306002	0.36742382	0.21872561	0.02056281	0.30022774	275913
12	0.27306656	0.24509251	0.19386313	0.02444751	0.26353029	290793
13	0.16207476	0.29196009	0.25399215	0.0256205	0.2663525	295730
14	0.07860716	0.38166893	0.18467831	0.0313992	0.3236464	299822
15	0.14986275	0.34020733	0.20457184	0.02455812	0.28079996	302233
16	0.22780377	0.35154967	0.16071118	0.0174122	0.24252318	304991
17	0.06897218	0.34447142	0.25321607	0.03394344	0.29939689	306376
18	0.07108929	0.36478737	0.16345044	0.03501866	0.36565424	306971
19	0.06223148	0.37611845	0.24614787	0.01825463	0.29724757	310232
20	0.06771681	0.41034178	0.21201496	0.05091495	0.2590115	310268
21	0.19787725	0.32366671	0.18736097	0.02932959	0.26176548	310550
22	0.09196842	0.39964871	0.19896683	0.03308724	0.2763288	318140
23	0.13526706	0.26586355	0.20005162	0.03279242	0.36602535	322484
24	0.057955	0.42627407	0.2180006	0.03136065	0.26640968	326477
25	0.08048636	0.12481513	0.32489004	0.01675436	0.45305411	326955
26	0.13841757	0.37843613	0.16799431	0.046557	0.26859499	327925
27	0.06868936	0.477611	0.15886809	0.0335893	0.26124225	334971
28	0.09216989	0.24045428	0.21706252	0.02384189	0.42647142	335040
29	0.06113046	0.43754502	0.22294304	0.04588817	0.23249331	336616
30	0.52131513	0.27028116	0.07561134	0.01302294	0.11976943	337812
31	0.06575672	0.3742717	0.22225095	0.03716376	0.30055687	337879
32	0.06530575	0.34804723	0.23546979	0.02502648	0.32615075	338228
33	0.07455129	0.29284221	0.23458064	0.02569791	0.37232795	338731
34	0.07455129	0.29284221	0.23458064	0.02569791	0.37232795	338731
35	0.07805432	0.41817225	0.18855444	0.01990829	0.2953107	339044
36	0.06288985	0.38343387	0.23374117	0.03248193	0.28745318	342346
37	0.13036372	0.29904968	0.23335143	0.03013342	0.30710175	343671
38	0.12743913	0.38648042	0.21444255	0.03710114	0.23453676	343765
39	0.08053948	0.32581943	0.21949447	0.04822804	0.32591858	346312
40	0.1121519	0.3288244	0.26408131	0.03325042	0.26169197	347852
41	0.13152209	0.30170694	0.23542492	0.02151549	0.30983056	349632
42	0.07185043	0.35884644	0.26378294	0.02177275	0.28374744	351442
43	0.05902106	0.32841611	0.19937961	0.05253645	0.36064677	356115
44	0.06670698	0.38960703	0.24489989	0.02226257	0.27652353	357752
45	0.1066219	0.41945959	0.14916547	0.03354222	0.29121082	361005
46	0.06662824	0.35170025	0.24367874	0.0411907	0.29680207	364575
47	0.06359921	0.36330018	0.24014878	0.04225638	0.29069545	364575

48	0.10215898	0.2530403	0.25228677	0.03918839	0.35332556	379172
49	0.1127774	0.32825933	0.28649523	0.0523588	0.22010924	389688
50	0.19445345	0.32704307	0.02973943	0.02007151	0.42869254	391015
51	0.15115832	0.33912224	0.16458553	0.03403676	0.31109715	392054
52	0.09346972	0.29062289	0.24441773	0.04078346	0.3307062	395946
53	0.06805081	0.31089009	0.25695788	0.03614461	0.32795661	398236
54	0.0562679	0.40898462	0.20520924	0.03926148	0.29027676	402027
55	0.2046861	0.16958546	0.26545796	0.02588316	0.33438732	402487
56	0.06177879	0.20494537	0.1814023	0.01557485	0.53629869	405882
57	0.17585142	0.21897511	0.20006466	0.04319106	0.36191775	406275
58	0.10623488	0.32530889	0.22909462	0.03455657	0.30480504	411792
59	0.08756783	0.3109158	0.18930325	0.0390293	0.37318382	413343
60	0.29185236	0.19666966	0.19213729	0.0346622	0.28467849	415220
61	0.13391958	0.38979749	0.23384739	0.03639778	0.20603776	415853
62	0.15272375	0.35336095	0.17149598	0.07222261	0.25019671	421032
63	0.46863021	0.17099919	0.1605514	0.02421578	0.17560342	425709
64	0.1428748	0.37846911	0.16958082	0.03633207	0.2727432	426751
65	0.20689858	0.28419478	0.21600753	0.01732219	0.27557692	431132
66	0.04090064	0.59197559	0.1474954	0.03268217	0.1869462	432772
67	0.17778434	0.35541891	0.21577286	0.0304212	0.22060269	434087
68	0.17842883	0.42136542	0.02466906	0.0456889	0.32984779	436671
69	0.07592504	0.40199353	0.18998184	0.03328137	0.29881822	440517
70	0.08985748	0.41717373	0.19425298	0.02434131	0.2743745	442552
71	0.13017074	0.35797595	0.18388391	0.03339652	0.29457288	456986
72	0.07364165	0.40942717	0.18286195	0.07144478	0.26262445	457570
73	0.05476164	0.34805773	0.18247028	0.02681655	0.3878938	462336
74	0.10588937	0.27504192	0.36549832	0.01499627	0.23857412	473938
75	0.08622381	0.40030403	0.33003996	0.02314017	0.16029203	476891
76	0.06594369	0.32410368	0.18248456	0.05200701	0.37546106	491504
77	0.15179933	0.15740929	0.34922065	0.02905027	0.31252046	496184
78	0.12292148	0.3503314	0.10200433	0.02880484	0.39593795	500914
79	0.06750147	0.39567003	0.19477837	0.0335185	0.30853163	501191
80	0.08128586	0.29736731	0.21788773	0.04077264	0.36268646	514715
81	0.42231116	0.15409776	0.14468262	0.03819201	0.24071645	520994
82	0.07216149	0.31104447	0.36962625	0.04121196	0.20595583	563020
83	0.12724765	0.31078934	0.20432739	0.02500692	0.3326287	595229
84	0.16693891	0.31375161	0.22359883	0.04954524	0.24616541	609167
85	0.17066198	0.18548115	0.2307416	0.04738495	0.36573032	628074
86	0.11064802	0.25382257	0.23831441	0.13655812	0.26065688	927379
87	0.11517762	0.38921366	0.21239874	0.02266201	0.26054797	975838
88	0.06533594	0.39555327	0.29252969	0.01560926	0.23097184	1035010
89	0.03168653	0.1594798	0.1282719	0.51151678	0.16904499	1142190
90	0.06674698	0.38853589	0.21714539	0.02074674	0.306825	1205181
91	0.1588699	0.37599726	0.02196489	0.25290986	0.19025809	1247735
92	0.05498969	0.42553539	0.20742267	0.02237005	0.2896822	1259890
93	0.4109456	0.20753162	0.1525248	0.02747047	0.20152751	1276702
94	0.06337913	0.50100518	0.18288319	0.02063034	0.23210216	1356176
95	0.03833539	0.18574263	0.12211586	0.47858504	0.17522108	1359610
96	0.10681862	0.21354726	0.12293292	0.35646363	0.20023757	1545899
97	0.03926375	0.22939324	0.43722508	0.09144759	0.20267034	1550404
98	0.20692883	0.16539058	0.43050673	0.02220095	0.17497291	1773419
99	0.02429145	0.16570499	0.10472282	0.30829757	0.39698317	1960050
100	0.06724402	0.52245692	0.13497667	0.01739349	0.2579289	2742127

附錄四 不同投入數的結果

投入數 70

代數	最小時間(s)	最大時間(s)	平均時間(s)	代數	最小時間(s)	最大時間(s)	平均時間(s)
0	361038	4413504	2011445	11	249876	3326546	619314
1	288167	3608412	1598670	12	249876	3008689	529287
2	288167	3164825	1081407	13	247712	3428204	546316
3	284346	4127323	868853	14	247712	2519130	532789
4	249876	2566007	671785	15	247712	2867153	629318
5	249876	3833912	709650	16	247712	2716615	597673
6	249876	2967349	606197	17	247712	3026140	481784
7	249876	2864329	645774	18	247712	2492289	507538
8	249876	2089816	528002	19	247712	2301674	548898
9	249876	2875754	559970	20	247712	2729032	595874
10	249876	3984369	547287				

投入數 71

代數	最小時間(s)	最大時間(s)	平均時間(s)	代數	最小時間(s)	最大時間(s)	平均時間(s)
0	445821	4546732	2461859	10	291520	3451953	708310
1	366025	4380332	1929706	11	262675	3189581	699140
2	301557	4565053	1468517	12	262675	3077342	727552
3	301557	4416843	1152230	13	262675	3680265	754301
4	301557	4241149	887814	14	262675	4333266	826546
5	301557	3180601	751653	15	262675	3660440	688482
6	301557	4047339	779353	16	262675	3267623	667215
7	291520	6038094	700317	17	262675	3414345	848607
8	291520	3240153	686703	18	262675	3728031	886434
9	291520	3634995	693396				

投入數 72

代數	最小時間(s)	最大時間(s)	平均時間(s)	代數	最小時間(s)	最大時間(s)	平均時間(s)
0	426691	6016647	3063825	11	314784	4734246	863889
1	426691	5656473	2547494	12	314784	4607075	903954
2	409316	4643240	2014266	13	314784	3610411	1033649
3	409316	4086609	1460687	14	314784	3589575	1065561
4	400343	6264325	1275599	15	314784	3701146	912549
5	365392	4982573	1138176	16	314784	3525444	1046722
6	348993	3487240	849430	17	314784	4440050	1041940
7	348993	3139049	891293	18	314784	5069415	1134714
8	348993	4088228	868582	19	314784	3563013	938441
9	348993	3779417	975022	20	314784	3634987	924832

10	314784	3626687	981255	21	314784	3354631	1018151

投入數 73

代數	最小時間(s)	最大時間(s)	平均時間(s)	代數	最小時間(s)	最大時間(s)	平均時間(s)
0	720179	6489352	3699135	11	428036	5005108	1220679
1	575306	6423380	2963081	12	428036	4605740	1177406
2	471716	5403990	2172254	13	428036	4094159	1151451
3	455388	4753040	1799985	14	421082	4330669	1313031
4	449303	5436257	1561236	15	421082	4697879	1193562
5	449303	5158475	1487922	16	421082	4459838	1247815
6	449303	5237335	1603437	17	388087	4887068	1192743
7	443151	4565352	1261876	18	388087	4824044	1119383
8	428036	5473230	1155737	19	371934	4628041	1214561
9	428036	4976499	1391587	20	371934	5619004	1208762
10	428036	5652086	1317952	21	371934	3578975	859413

投入數 74

代數	最小時間(s)	最大時間(s)	平均時間(s)	代數	最小時間(s)	最大時間(s)	平均時間(s)
0	646379	7394909	4283448	11	439419	5947258	1568295
1	646379	9839753	3599482	12	439419	8161083	1356127
2	573152	6259383	3021612	13	439419	4215175	1170571
3	516585	6837094	2479740	14	439419	4372396	1495301
4	516585	4840316	2226616	15	439419	4541254	1203660
5	516585	4851375	2076936	16	439419	4349304	1247141
6	439419	5020835	1854302	17	439419	4913516	1452105
7	439419	5072754	1522898	18	439419	6144684	1469699
8	439419	6230304	1727242	19	423198	4881460	1198837
9	439419	7307007	1713019	20	423198	7299705	1399225
10	439419	4499429	1523945	21	423198	4998167	1492505

投入數 75

代數	最小時間(s)	最大時間(s)	平均時間(s)	代數	最小時間(s)	最大時間(s)	平均時間(s)
0	864433	10305689	5175063	11	659133	8969534	3559714
1	822206	10123889	4125056	12	659133	7813687	3296021
2	725825	7555800	3520451	13	639637	6309547	3200331
3	725825	8527691	3605298	14	639637	7196354	3078200
4	695966	9436802	3442526	15	639637	7165725	3095469
5	695966	8426633	3350276	16	639637	7713642	2898487
6	659133	6003550	3240844	17	639637	5463300	3104636
7	659133	8189881	3187524	18	639637	10550168	2967762
8	659133	5735655	3229289	19	639637	5018186	2722297
9	659133	5828615	3177471	20	639637	6864923	2821042
10	659133	5976000	3343462				

簡歷

學歷

2004 年 9 月 ~2006 年 6 月 交通大學 電機學院與資訊學院專班 資訊學程

1996 年 9 月 ~2000 年 6 月 元智大學 資訊工程學系

經歷

2004 年 8 月~目前 中華映管股份有限公司 CIM 工程師

2001 年 6 月~2004 年 6 月 新加坡商軟思股份有限公司 程式設計師

2000 年 8 月~2001 年 7 月 台灣科技大學 國科會助理

2000 年 3 月~2000 年 9 月 福特汽車 程式設計師

1999 年 7 月~2000 年 9 月 資策會技術研究處實習資策會技術研究處實習