國立臺灣大學電機工程學研究所碩士論文

指導教授：張時中博士

以增強式學習法設計機台派工法則之研究

Research on Design of Machine Dispatching Policy Using Reinforcement Learning

研究生：吳欣曄 撰

中華民國九十三年六月

誌　謝

兩年的研究生活不長也不短，卻是我人生的一大轉戾點。首先要感謝的是指導老師張時中教授用心良苦的教誨；在課業上，他讓我學會教科書所沒有的研究方法；在思想上，他給了我一個獨立思考的空間；在生活上，他則對我示範了什麼是正確的人生態度。特別要感謝的是暨南大學的廖大穎教授，不辭辛勞的兩地往返，他在我的研究與論文上提供了很多寶貴的建議。本論文部份係有國科會計畫(編號 NSC 91-2212-E-260-002, NSC 92-2212-E-002-060)的支持下完成。也感謝口試委員陳正剛教授提供寶貴之指正與建議，使本論文能更正確、完整。

研究期間，林偉誠及朱邵儀學長在生活上、課業上和研究上的幫助，每次的討論與建議都讓我獲益良多，這份情誼永銘於心。那些在控制與決策實驗室中曾經朝夕相處的研究夥伴們，宗慶學長、宜學學長、家龍學長、銘辰、豐凡、圻軒、德培、嘉政、家賓、倉弘、以民、旻旻、元祥、加將、宗源、安邦、萬軍、志偉、昭德、冠儀、助理淑宜、怡泓…等等，還有那些曾在我生命留下軌跡的人們，這些研究生涯中的好夥伴，有了你們的參與，讓我枯燥的研究生生活更加多彩多姿。

最後，最該說聲謝謝的是我的父母與家人，沒有他們的支持與鼓勵，是不可能有今天的成果；我願將此榮耀與他們一起分享。

論文摘要

在半導體晶圓的製造線具有多樣化的產品、回流特性的複雜製程、不確定性的機台、客戶導向、高投資成本、生產週期短等特性。如何將不同種類的產品有效地派工(Dispatching)給機台，來富有彈性地製造不同產品，達到準時交貨仍然是一個具有相當挑戰性的研究課題。在同一部機台處理不同製程的晶圓時需要製程轉換，而製程的轉換需要設置時間。一方面，為準時而彈性地製造不同產品，需要適時地調整設置次數;另一方面，減少設置次數可以降低在製品，及減少等候時間。而一種產品製程會不只一次經過同一類機台的回流現象也造成同一產品不同製程競爭機台的情況，因此產能必須恰當配置以達成準時交貨與平衡產流的順暢。本論文研究有設置時間之單一機台派工問題，及可調速率機台之速率切換問題。前者著重在產品平均等候時間與設置次數的取捨，而後者的目標是在等候與高低速的製造成本間作最佳取捨。因此要如何決定下一個加工產品的種類，及切換速度的時機，是本論文所研究的挑戰。

由於現場環境隨時間變化，必須持續調整派工法則。本論文嘗試以增強式學習(Reinforcement Learning)來處理這兩類問題。它將可以隨著環境不同而持續學習，藉由定義合理的回饋(Reward)，估計價值函數(Value Function)來找到最適合的派工決策。針對派工是依據現場狀況做決定的特性，我們假設狀態之間具有馬可夫性質(Markov Property)，將派工問題近似成連續時間的馬可夫決策過程(Continuous-time Markov Decision Process)。

在有設置時間的單一機台派工方面，我們對工作到達率為平穩(Stationary)的問題，用法則疊代法(policy iteration)解出最佳解。但法則疊代法無法解決非平穩(Non-stationary)及機率特性不詳的問題。因此我們嘗試應用Sarsa[RsA98] 的增強式學習演算法，它是一種每個動作都依循著某一特定法則的法則性(On-policy)的學習，具有簡單易計算的特性。它不需要機率特性即可解馬可夫決策過程的問題。在平穩馬可夫決策問題實驗中顯示出隨著學習次數增加，學出來的決策與最佳解有 95%的正確率。我們進而將增強式學習應用在

具有非平穩環境的有設置時間的單一機台派工問題上，並與一個用一樣機率選取不同種類產品的隨機法則來做討論與比較。結果顯示，在非平穩的工作到達率下，增強式學習使平均等待時間穩定在某一定值附近，而隨機法則的平均等待時間則是逐漸增加。增強式學習也比隨機法則的產出增加了 30% ，設置的次數也比隨機法則少許多。研究結果證實增強式學習可以解決法則疊代無法處理的派工問題，即具有非平穩環境的馬可夫決策過程。但是增強式學習的學習速度不夠有效率，且也無法證實非平穩環境的最佳解為何。而若從給予一個由 Kumar 與 Seidman, 1991, 所提出的清除法則開始學習，相對於沒有學習能力的清除法則，增強式學習對於平均等候時間隨學習次數減少。

另外對於具有可調速率機台之速率切換問題，我們考慮生產花費成本與等候權重成本間的取捨，試圖找出最佳切換速率的時機。一如前述之機台派工問題，我們將可調速率機台之速率切換問題表示成馬可夫決策過程，並且使用增強式學習嘗試學出最佳切換時機。結果發現，需要學上千萬次才可以得到最佳解，不夠有效率。另外用法則疊代來佐證最佳切換速度的時機，並探討了各個參數與最佳切換時機的關係:包括工作到達率與高速加工所需成本。實驗顯示出，越高的工作到達率，及越低的高速加工成本，會使得轉換時機發生在等候工作較少時。最後，我們發現在給予最佳法則下，增強式學習對於學習環境改變後的最佳法則，其學習速度可以較從無開始學習到最佳法則快一千倍以上。而對於實際生產線的派工學習，則有待進一步的評估。

Abstract

Semiconductor fabrication is characterized by a variety of products, complex re-entrant flow, machine uncertainty, customer orientation, high investment, and short product life cycle. Effective methods for different lots dispatching that lead to achieve production flexibility and on time delivery still pose significant challenges to both researchers and practitioners. It requires the setup time when changing the processing type in the same machine. In the one hand, it needs appropriate setup for producing different lots flexibly and timely. On the other hand, it can reduce the workload level and waiting time by decreasing setup times. However, the reentrant flow causes the competition among different type lots to the same machine. Hence machine capacity must to be allocated effectively to reach on time delivery and balancing the production flow. We studied the single machine with setup time dispatching problem and adjustable service rate machine problem. The objective of the former is tradeoff average waiting time and setup times, and the latter is tradeoff waiting and service cost. How to choose the next product type and timing to switch the service rate are the challenges we meet.

The dispatching policy must be adjusted continuously because the environment changes over time. We tried to solve the problem using Reinforcement Learning (RL). It can interact with environment and find the suitable policy by Reward function and Value function. We assumed that the states have Markov property and formulate dispatching problem as Continuous-time Markov Decision Process (MDP).

In the single machine with setup time dispatching problem, we used Policy Iteration (PI) to find the optimal policy on the Stationary job arrival environment. But PI cannot solve Non-stationary problems or unknown system dynamics problems. We referred to the RL Sarsa algorithm [RsA98] to apply to our dispatching problem. It is an on-policy learning that learns the value of the policy that is used to make decisions. And it is conceptually and computationally simple to solve MDP without system dynamics. In the stationary case, RL learned 95% correctness of optimal policy with enough learning step. Furthermore, we applied RL to Non-stationary dispatching environment and compared with Random Policy. The Results showed that RL

stabilized the average weighted waiting time but Random Policy did not. RL increased the 30% throughput and decreased switched numbers than Random. This research showed that RL can deal with the dispatching problem that PI cannot. But the learning speed is not effective. We also don't know the optimal policy in the Non-stationary environment. However, starting with given a Clearing Policy that proposed by Kumar and Seidman, 1991, RL makes less average waiting time than Clearing Policy.

In the adjustable service rate machine problem, we considered the tradeoff of the service and waiting cost, and tried to find the timing to switch the service rate. We also formulated this problem as MDP and applied RL to solve. The Results show that RL need to learn 10 million steps for finding optimal switched point. We studied the relationship between parameters, including arrival rate and high service rate cost, and timing to switch. We found that the higher arrival rate and lower high service cost let the switch at fewer waiting jobs. Finally, we found that RL, which had prior knowledge, learned 1000 times faster than one's had not. Besides, it still requires evaluating for real dispatching learning.

目錄

圖目錄

表目錄

第一章　緒論

1.1 研究動機

由於全球電子產品的需求大幅成長，在近幾年 Centrino 風潮，數位照相機等等的數位家電持續增溫。全球的電子產品出貨量持續上升，因此造成半導體製造業的需求動能持續的成長[Dig04]。以至於半導體生產控制的議題越來越受到重視，也充滿許多挑戰與困難，令許多研究者投入於此。在半導體晶圓代工廠中具有多樣化的產品、複雜的製程、機台的不確定性(有當機的可能)，回流特性，設置時間的必要，以及特殊的機台限制。因此擁有快速，有彈性的製造環境，才能在全球激烈的競爭下脫穎而出，達到獲利的目標。所以如何有效派工*(Dispatching)*使得準時交貨，生產週期短，達成顧客要求，是一個生產控制的重要課題。然而降低生產成本，提高生產效率，縮短產品至市場的時間，達成獲利的目的，以至於如何順暢整個生產線，減少在製品的堆積，都是產業界與許多學者亟欲達成，努力的目標。也是此論文想深入探討的動機。

派工的功能是根據現場狀況,產生某一工作站或機台晶圓加工順序。對於整個工廠來說，派工扮演著舉足輕重的角色。一個良好的派工法則可以使工廠的生產週期減少 32.9%[CWH97]。甚至對於平均生產時間，與標準差的改善高達 52%[LRK94]。另外好的派工方法對於產品的平順度(Smoothness)，及設備使用率的改善也有很大的助益[Tho95]。

由於半導體生產環境具有以下的特性，使得良好的派工具有相當的困難度與挑戰性。

a.) 多樣化的產品:

在一個半導體廠裡，有上百種不同的產品在廠中製造著。每種產品的製造流程都不一樣，且每種產品都有特定要使用的製造機台或設備。

b.) 回流性質:

半導體廠具有一個與其他工廠最不一樣的特性，就是回流(Reentrant-Line)。回流的意思是，產品會不只一次經過同一個機台。IC 製造廠的作業是將晶圓廠所做好的晶圓，以光罩印上電路的基本圖樣，再以氧化、擴散、CVD、蝕刻、離子植入...等方法，將電路及電路上的元件，在晶圓上做出，由於 IC 上的電路設計是層狀結構，因此還要經過多次的光罩投入、圖形製作、形成線路與元件等重複程序，才能做出一個完整的積體電路。所以同一個晶圓需要做多次相同製程，造成不同產品常常會需要面臨爭奪同一機台的情況。所以回流特性是使得半導體廠顯得更複雜的主要原因之一。

c.) 多樣化設備與特性:

有上百種機台在一座廠裡面，而且都是高精密的裝置，所以定期的維護與檢測是必要的，也就是說，不預期的機台當機是有可能發生的。因此會影響到生產流程的速度。另外，機台還有些特殊特性，比如說爐管區的整批作業特性，也就是說一定要到某種程度的數量才可以加工。或者是黃光區的光罩限制特性，和光阻液的更換需要設置時間的特性，這些都是影響晶圓廠製造管理的重要因素，也是困難所在。

d.) 龐大的派工排程法則:

有些啟發式法則(Heuristic)存在於實際的工廠中，不同的製程站所用的法則當然也會不同，因此組合出來的法則有上百種，然而工程師在決定選取法則時，都是憑著經驗以及與別的工程師協商，因此面對龐大的排程法則，沒有一套有效率的選擇方式，沒有經驗的工程師也無法選取出適合當下的法則。

這些特點使得晶圓廠的控制策略變得非常複雜與困難，正如 Kumar[LRK94] 所提到的晶圓製造流程因為具有回流特性，所以不像 Flow shops，也因為其流程具有結構性，所以也不像 Job shops。因此針對不同特色的機台，在晶圓廠

裡的派工種類眾多且複雜，而本篇論文致力於以下兩種不同類型的派工問題。

1.有設置時間的派工問題:

有設置時間的派工問題常常出現在各種生產系統。所謂設置是:一個機台可以做不同產品，而轉換製程的時候，需要一些時間或成本來設置(Setup)。半導體廠裡的黃光區(Photolithography)往往被認為是瓶頸(Bottleneck)所在，因其機台非常昂貴，且加工步驟複雜需要設置時間。如上光阻步驟，與光罩步驟皆需要設置，此一過程佔了總生產時間的一部份，且要耗費時間與金錢。另外機器設置在印刷電路板(PCB)產業，是一個相當費時的步驟。製造業者表示，PCB組裝的設置時間最多花費總製造時間的一半。長的設置時間增加存貨的數量，而當機器在設置時，若其後生產線正好是閒置狀態(idle)，則會減少生產線的產能。儘管負責設置的操作員很忙碌，下游的組裝員必須等待設置完成才能開始作業[TrB03]。因此適時的更換製程，以達到產品總等候時間變短，乃至於可以增加生產力，達到如期交貨，是我們此類問題要研究的重點。

2.可調生產速率之機台問題:

在半導體廠中，因為具有回流的現象，同一產品會經過相同製程站多次，而同一種機器可以負責做不同的製程。如圖 1-1 所示，是一個三機台，四個製程站的回流系統。此時就會機台 2 會造成製程站 2 與 4 的資源競爭。實際工廠中同一製程站裡的機器不只有一台，不過我們可以把他叢聚成一台可以調整速率的機台。因為速率的高低代表著分配機器的多寡，高速的處理意味分配較多的機台參與製程，低速則相反。而機台產能分配在整個工廠的生產控制中佔有不可或缺的地位。因為在許多工廠中，往往是以分散式的生產控制系統[HuS98]。而系統裡最小的單位即為一部機台，因此在單一機台中的產能分配往往是影響著整個工廠的產出，生產週期及其他生產要求。因為當特定製程站前的貨堆多的時候，必須分配更多的機台給此類產品進行加工，這樣一來其他製程站相對的少了機台可加工，這變成了不同種類製程站之間的資源競爭。所以在有限的資源下，要如何有效的分配產能，以達到給定的區域性生產要求，是工程師們最想達到目標。

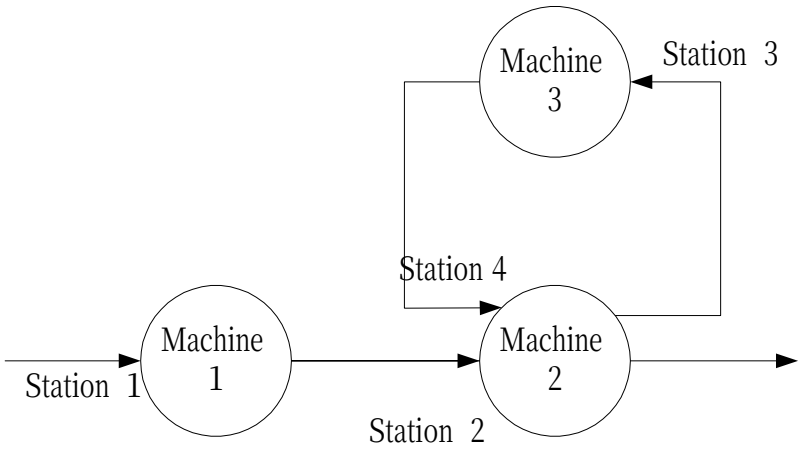


圖 1-1:具有 3 機台 4 個製程站的回流系統

1.2 文獻回顧

關於有關派工與排程的文獻，以解決方案來分類的話，其種類可分為以下五種: 啟發式法則，佇列控制法，模擬解，學習式代理人解，以及增強式學習法解。以下就分別討論此五類文獻所研究的議題及其利弊:

a.） 啟發式法則(Heuristic approaches)

Wein 與 Lu[LRK94][Wei88]指出良好的派工法則，可以使半導體廠的生產週期縮短 44.9%。Lu 提出了 FS(Fluctuation Smoothing)的新派工法則。此法則跟其他已存的法則做比較(如 FIFO(First In First Out)，SRPT(Shortest Remain Processing Time))，在平均生產週期時間減少 22.4%，且生產週期的標準差減少了 52%。[TFL03]提出一種混合式的派工法則，他將機台分成瓶頸(Bottleneck)的機台，非瓶頸的機台，以及批次(Batch)加工的機台三種不同對象。每種不同機台都用不同的派工法則，此方法增加了整個系統的產量，且減少了在製品的堆積(WIP)與生產週期。不過工廠的不確定高，在一些突發事件發生，一些啟發式法則會失敗。

b.） 佇列控制法(Queuing Control approaches)

[Che00]使用排隊(Queuing)的模型，研究單一機台產能配置的問題，他所看的是有當機可能及可以調整平均生產速率的機台，生產速率的控制策略代表產能的配置。將生產速率調整問題表示成一馬可夫鏈，並解出穩態時各狀態的機率，並求出生產需求的統計值。[Lee96]則是將工廠內的動態資源分配，已隨機最佳控制的觀點來研究機台不穩定影響下的動態機台配置及投線策略。他將此問題形成無窮時域馬可夫決策過程(Markov Decision Process)，並使用連續近似法求得最佳機台分配的數值解。

c.） 模擬的解法(Simulation-based approaches)

[HCC00]採用了排序最佳化的方法來動態選取派工法則。 此法可以有效的減少計算時間。為了快速地從資料庫中選取一個夠好的排程法則，在[HCC00]創新地結合排序佳化(ordinal optimization)以及實驗設計(design of experiments)的概念，發展了一個快速的模擬方法。有別於傳統模擬方式求取排程法則精確的績效值，排序佳化比較各個排程法則績效值的相對排名。利用實驗設計方法大幅刪減需要模擬的排程法則個數。應用在晶圓廠模擬的結果顯示大多數的模擬實驗，排序佳化實驗設計模擬可以節省三到四個數量級的時間。在一些例子中，排序佳化實驗設計模擬比起傳統模擬方法，可以加快高達7,000 倍的計算時間。

d.） 代理人(Agent approach)

代理人被廣泛的應用在許多方面，網路，排程，電子商務，機器人學等[Csm03]。 [NaB93] 和 [NaB91]提出了具有知識表達架構的智慧型代理人(intelligent agents)，此代理人可以有模擬製造系統的作決策的能力。[Lin02] 建立了一個派工的知識庫，並且靠著訪談實際工廠中的工程師，推導出所謂決策樹和權重模組的結構，此結構可用來表示實際上工程師派工的知識與經驗。而[Cha02] 設計出具有學習能力的代理人來學習工程師的派工知識，並且決定派工的順序。他們設計了一個軟體代理人，且以配合度(Degree of Matching)驗證了其正確性。但是此種代理人的學習法，是屬於教導式的學習

(Supervised Learning)，直接學習人類給予的知識。不過此法有個缺點，即無法學到人不知道的情況。

e.） 增強式學習(Reinforcement Learning Approach)

增強式學習的基本觀念為心理學在解釋動物或嬰兒的學習過程，他是一種嘗試錯誤(Trial-and-Error)的學習法，不直接靠著指導者來教導知識，所以是一種非教導式的學習(Unsupervised Learning)

[SoW90]應用增強式學習的技術在控制與製造最佳化，有成功的應用。而[HaK98]出發點在動態且難以預測的市場，及製造控管。他看的是顧客導向的製造系統，此系統需要大容量的產品線及有彈性的製造技巧。雖然此研究成功的證明增強式學習可以應用在製造系統上，但是仍然還有兩個障礙。第一:製造者非常的小心翼翼，因此需要強力的理論證明來說服他們。第二:增強式學習以往著重在離散的狀態系統，但是實際上的問題，測量時間與狀態往往是連續的。許多最佳化問題，不論是倉儲控制，排程等都可以表述成半馬可夫決策過程(Semi-Markov decision process)。而主要的目的在於使得總花費達到最小，並找到最佳的策略。[MMD97] 描述了平均回饋的演算法(average-reward algorithm)去找最佳的策略。此演算法在倉儲控制問題裡面，有不錯的表現。[MaT98]描述了機台維護問題，機台在使用一段時間過後會有損壞的可能，此時面臨要保養或者繼續做生產，做保養可能需要花費一些時間，而此時影響到產品的出貨;然而繼續生產則有可能加重機台損壞的可能。因此[MaT98]將此問題表述成半馬可夫決策過程，且說明了一個看似簡單的問題，其狀態的龐大度。而增強式學習也可以應用在狀態不明確和具有時變機率的情形，如電梯排程問題，在[RhA96]中使用增強式學習做出不錯的效能。

1.3 研究範疇

在半導體廠中，具有回流的現象，以及複雜的製程，製程的轉換之間需要設置時間，適時的調整設置次數可以降低在製品，及達到設備最大使用率。因為回流現象使得許多不同製程爭奪同一機台的情況。此時有效的產能必須被配置恰當。而相同機台產能分配的問題，可以看成單一機台可調速率的問題，速率即是產能分配的大小。所以有設置時間之機台派工問題，及可調速率之生產機台問題在此顯得重要且有挑戰性。我們討論有設置時間的單一機台派工問題。機台在轉換製程時需要時間與成本來設置，此設置時間往往佔了總處理時間的一部份而不可忽略。在晶圓代工的複雜製造線上，需要不同製程的晶圓會經過同一部機台許多次，因此不可避免的會有設置時間的產生。切換製程的時間過長可能會造成其他製程的在製品(WIP)增多;過短也會造成不必要的時間與成本浪費。因此要如何決定下一個加工產品的種類，是一項挑戰。因此我們欲設計一派工法則以平衡設置的次數，使得平均等候時間減少，進而讓總產量增加。

在現今工廠中，工程師主要依據他們的經驗來派工，不過工程師無法得知經驗以外的事情。因此結合學習與模擬的方法，可以事先模擬實際狀況，在藉由學習機制學出對應的好決定，而增強式學習即為此類的方法。針對派工是依據現場狀況做決定的特性，我們假設狀態之間具有馬可夫性質(Markov Property)，將此問題近似成連續時間的馬可夫決策過程(Continuous-time Markov Decision Process)，而目標函數(objective)是使得產品的等候時間最小。接著在平穩(Stationary)的環境下，我們用法則疊代法(policy iteration)解出最佳解。由於我們的問題是為一個有限花費，連續時間的且沒有穩態的問題，我們將連續時間的馬可夫決策過程經過一致化(Uniformization)轉換成離散時間的馬可夫決策過程，再利用修改為無穩態問題的增強式學習演算法Sarsa[RsA98]來嘗試解之。平穩馬可夫決策問題實驗中顯示出隨著學習次數增加，學出來的決策與最佳解有 95%的正確率。我們將增強式學習應用在具有非平穩環境的派工問題上，並與一個用相同機率選取產品的隨機法則來做討論與

比較。結果顯示，在非平穩的到達率，增強式學習使平均等待時間穩定在某一特定值附近，而隨機法則的平均等待時間則是逐漸增加。而也比隨機法則的產出增加了 30% ，設置的次數也比隨機法則少許多。也證實增強式學習可以解決法則疊代無法處理的問題，即具有非平穩環境的馬可夫決策過程。但是對於學習速度不夠有效率，且也無法證實時變環境的最佳解為何。另外，增強式學習在給予一個 Kumar 跟 Seidman 所提出的清除法則狀況下開始學習，可使得平均等候時間比原先的清除法則還要小。這代表增強式學習可以學出較好的法則。

在工廠機台的產能分配的問題，我們考慮一部可以調整生產速率且有故障可能的機台來代表。生產速率的高低代表產能配置的大小。高速率的處理雖可加快產品的生產速度，減低在製品的堆積，但是高速率意味要付出較高的製造成本。因為有限的機台產能中，分配較多的機台加工此製程，代表著其他製程的產能要減少，雖然此製程的等待工作數減少，但是相對於其他製程來說，等候的工作數會增加。我們希望的目標是平衡等候工作數與高低速的製造成本。即是找到一個最佳切換速率的時機。因此有效的產能配置能減少在製品的堆積。我們考慮生產花費成本與等候權重成本，來將可調速率機台的問題表示成馬可夫決策過程，並使用增強式學習學出最佳轉換速率的時機，但是需要學上千萬次才可以得到最佳解。此外我們用法則疊代解出最佳切換速度的等候工作數來做佐證。這裡我們探討了各個參數與最佳轉換點的關係。包括工作到達率，高速加工所需成本。實驗顯示出，越高的工作到達率，及高速加工的成本越低，會使得轉換速率的最佳時機發生在較少的等候工作數。最後，我們發現給予最佳法則下，增強式學習對於學習環境改變後的最佳法則，可以較從無開始學習到最佳法則快一千倍以上。而對於實際生產線的派工學習，則有待進一步的評估。

1.4 論文架構

本篇論文的架構如下:第二章我們描述機台派工問題的重要性，以及實際應用上的派工學習依據。而學習法是一種可以補現今人們所不足的應用。當然，我們也介紹了增強式學習的基本理論。接著第三章探討了第一類的派工問題，即是有設置時間的派工問題。在此章我們證實了增強式學習可以應用在派工問題上，更可以進一步應用在非平穩的情況。而在第四章裡，我們則是研究了生產速率調整問題。在此章我們發現了，增強式學習對於事先已有法則建立的快慢程度。最後在第五章作總結論。

第二章

機台派工問題與增強式學習

本論文 2.1 節說明了此篇論文想研究的問題，有設置時間的單一機台派工問題，可調生產速率機台問題。及 2.2 節說明了為何需要學習派工法則，以及一些學習法的文獻，最後 2.3 節我們介紹了增強式學習(Reinforcement Learning)。

2.1 機台派工問題

在這篇論文中，我們著重在兩類問題:一個是有設置時間(Setup Time)的單一機台派工問題，一種是可控制生產速率的機台代表產能分配問題。

所謂有設置時間的單一機台派工問題是:此機台可以做不同的製程(如圖 2-1)，但是轉換製程時，需要花費更換設備或原料的時間與成本。而此設置時間佔了總生產時間的一部分不可忽略，要列入派工需要考慮的因素。例如半導體廠的離子佈植區，做不同的離子參雜時，需要完整的重新設置。而在晶圓代工的複雜製造線上，具有回流現象，晶圓會經過同一部機台許多次，此時不同製程的晶圓同時到達同一機台時，必須決定需不需要更換製程設置，來製造另一製程的晶圓。此區域也常被視為瓶頸區(Bottleneck)，瓶頸區指的是某機台或區域限制著整個產流的流暢，而瓶頸區可從等候區長度，機器使用率與產能滿載程度來判定。如圖 2-2，有六個不同晶圓都會經由一迴圈而再次經過此區域。這時有兩種難以抉擇的情況，1.)維持單一製程以減少設置的次數與時間，但是會造成其他製程的堆貨;2.)更改製程以達到此製程晶圓的準時交期，但必須犧牲時間成本去設置。而整個工廠的物流是否順暢，依賴著這些迴圈的堆貨情形。因此一個良好的派工法則以協助工程師去選取下一個派工的晶圓種類，

及決定何時該轉換製程，是重要且有挑戰的。

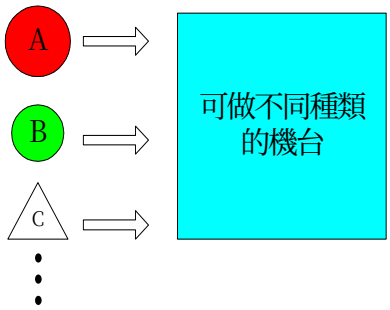


圖 2-1:可做不同種類的機台

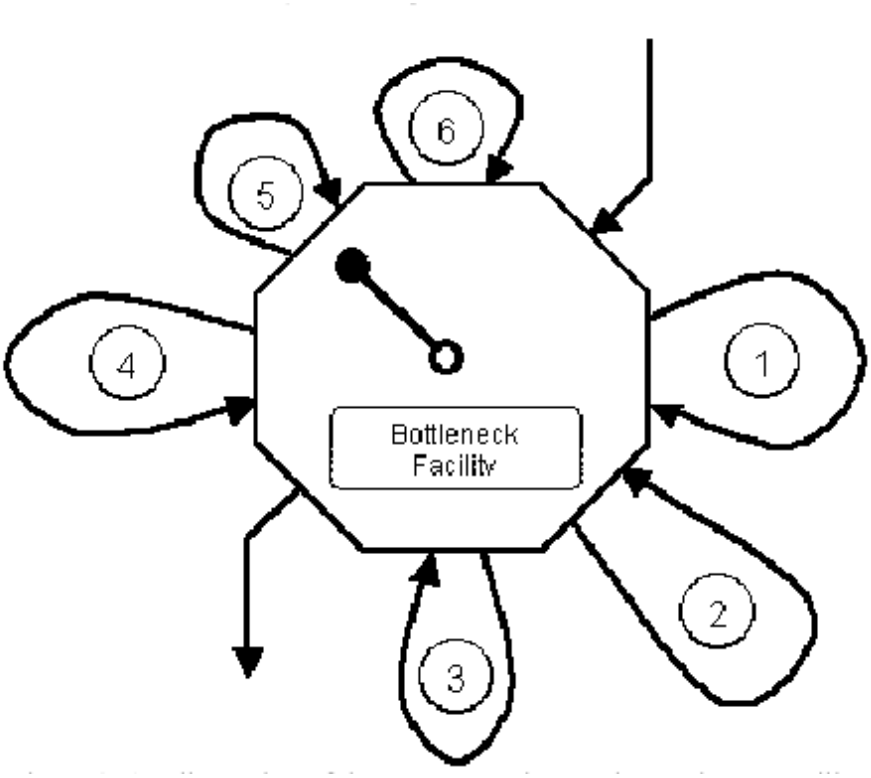


圖 2-2:有設置時間的瓶頸機台(來源[DMN01])

此外，在半導體廠裡常常遇到的另一個派工問題，為多個相同機台的派工問題。同樣的，同一種機台也可做不同製程。在半導體廠中，相同的機台被歸

納為同一區域，而因為晶圓是多層刻劃上去，同一晶圓需要經過多次的相同的區域做不同的製程，使得同一區裡的同一種類的機台必須常常考慮如何分配至每個製程站來達成每個製程的產能平衡。如圖 2-3 所示，一個簡單的派工情況:有五個相同的機台，分別負責做製程 A 及 B。若此時製程站 B 前面的堆貨較多，於是將負責 A 製程的機台分配給 B 製程，讓製程 B 的產流更順暢。所以此派工原則之一是依照等候區的總數決定。分配較多的機台，代表此製程站的生產效率提高，也就是生產速度變快。反之，生產速度變慢。因此我們可以用一部可以調整生產速率的機台，來模擬單一製程站的機台分配。如圖 2-4 所示，一個可調速度的機器，速度隨著等候區總數不同而不同，高速度需要成本較高，因為在有限的機台中，分配較多的機台加工此製程，代表著其他製程的產能要減少，雖然此製程的等待工作數減少，但是相對於其他製程來說，等候的工作數會增加。而等候成本也因等候區的數目而增加，意即越多的產品堆積，會造成下游的影響也越大。因此一個常常被提及的問題是，如何平衡各個製程站的產品堆積數目，使達到最高效益。也就是說平衡高低速與等候區的等候成本，達到成本最佳化，找出調整速率的最佳時機，是在廠內需要且必要的。

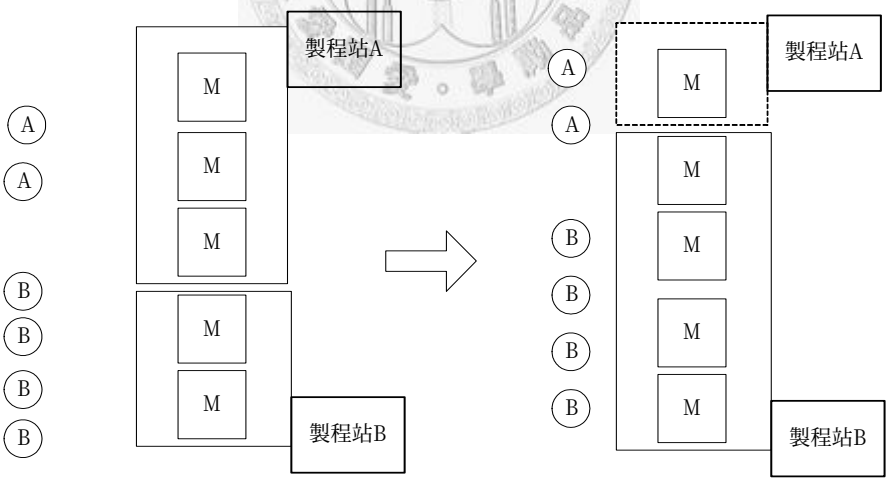


圖 2-3:多個相同機台派工問題

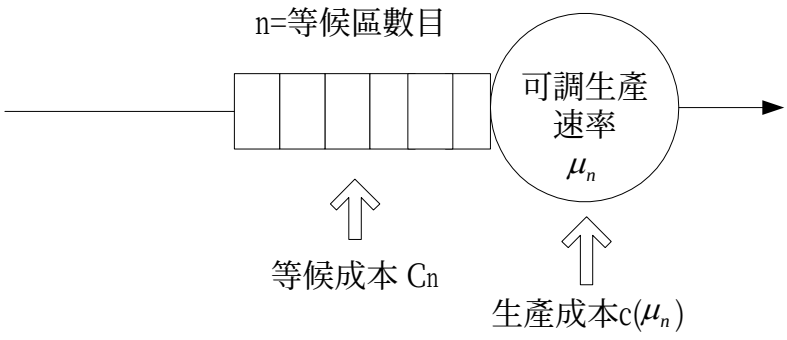


圖 2-4:可調生產速率之機台

2.2 派工法則的學習

在半導體廠生產管理中，決定如何將晶片派工給機台，或將機台派工給晶片，早已是一個重要的研究主題。廠內的設備價格昂貴且需要定期維修，因此如何在一個動態且不確定性高的製造環境中，利用有效的派工來達到準時交貨，及設備最大使用率(Maximum Equipment Utilization)，是一個很重要且有挑戰的問題。

在目前工廠中，工程師主要根據他們的訓練與經驗法則來執行派工。雖然經由適當的管理,這種做法可以獲得優異的生產效能，但是也存在著以下的問題:

1. 良好的實務大部分依賴有經驗的工程師
2. 派工的知識隱晦不明
3. 訓練一位工程師需要花費許多時間
4. 派工知識的紀錄與保存工作很重要但常被忽略

可看出派工知識對於派工的重要，而如何表達派工知識並學習，是許多學者研究的重點。因此許多人工智慧方面的方法被用來解決派工問題。有知識庫

與專家系統，學習代理人，類神經網路與增強式學習等等都被提出。這些方法主要有四項優點:1.)量化了做決策的知識 2.)可以創造出更複雜的派工法則 3.)能依照現在工作，下個工作，資源狀態，物流情形等決定派工法則 4.)可將複雜的資料用簡單的結構表示。然而也存在一些缺點:1.)可能需要大量時間來建立，維護與更新。2.)得到的解答可能是其中一解，而無法知道最佳解。在建立知識庫的第一步為知識的取得，之後將他存為數位資訊。此知識對於派工來說，可以是人類的專業知識與模擬的結果。Ross[Qui86]提出 ID3 的推演方法來學得知識並建立知識庫，而建立決策樹。決策樹為一種分類的方式，將人類的派工法則分類。[Cha02]與[Lin02]經過訪談工程師後，運用決策樹建立了一套知識表達的方式，提出了有學習能力的代理人，可以粹取真實人類派工時的知識。[Yih90]提出一種依循軌跡(trace-driven)的知識粹取方式(TDKA)，可以處理連續的狀態資訊。這方法是先模擬一個派工系統，讓工程師在這系統上模擬決策。而紀錄完整的決策過程，並分析之，而決策與系統的資訊隻間的關係稱為軌跡(trace)。接著分析軌跡並分類規則到不同的群體。分類停止的準則為大部分群體裡面的規則皆使用同一種派工法則。這時候派工法則已經被建立完成。而之後再與工程師做的法則比較。此方法缺點是模擬的過程可能耗費時間的，且模型的建立與實際工廠必須夠吻合，且無法適用高度變化的環境。當系統執行了一段時間後，資訊會改變，此時決策樹的結構會有一定程度的重建。[Chi94]提出了一種有容限(Tolerance-based)的演算法來維持決策樹的穩定，他著重在動態的派工系統，建立了三個模組:1.)離散事件模擬 2.)範例產生 3.)有容限的學習。此方法將不會讓決策樹時時改變結構，必須當新的範例足夠，才會改變。這方法雖然對於連續資料可以有效的歸納，避免一些垃圾資訊，不過在一些製造環境中，有些突發事件的發生是不可容忍的，也不可忽略。

綜合以上各點，我們可知光學習工程師的知識或光靠模擬的資料都是不足夠的，而學習的困難點在於:難以表示未知事件，無法如人類有靈活的推理，有許多範例外的事件無法處理。因此想以一個能持續學習經驗，並結合模擬結果來判斷的學習法，是個有挑戰性的議題。

2.3 增強式學習[RsA98]

本章簡略介紹增強式學習(Reinforcement Learning , RL)的基本理論與演算法。

1. 增強式學習的本質:

增強式學習原本是心理學在解釋動物或嬰兒的學習過程所用的名詞，他是一種嘗試錯誤(Trial-and-Error)的學習法，不直接靠著指導者來教導知識，所以是一種非教導式的學習(Unsupervised Learning)如圖 2-5，訓練記憶住他做出何種決定時，外在環境會給予他何種利益所得到的經驗來學習。例如嬰兒肚子餓了，或尿布濕了不舒服時，經由以往經驗知道哭很大聲將會有人來餵他或幫他換尿布，因此嬰兒就此知道以後只要大哭，就會得到幫助。在訓練動物的時候，也是一樣道理，因為動物並不聽得懂人類的語言，只是記得:在這個狀況下(state)，做這個動作(action)會有肉吃(reward)，做另一個動作將會招來一陣毒打(penalty)。因此，此學習擁有自動的，不需教導的，持續的改善自己的行為的能力。

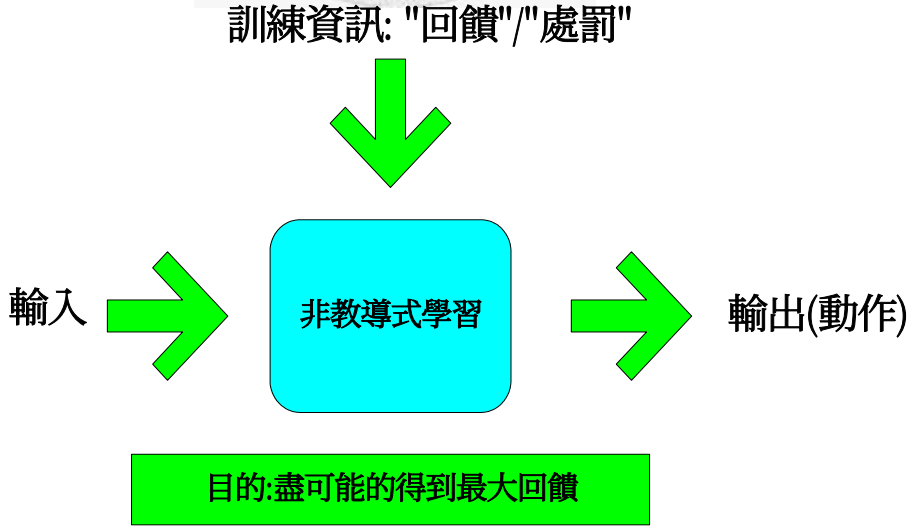


圖 2-5:非教導式學習

2. 增強式學習的基本要素:

圖 2-6 表示出增強式學習的基本結構，扮演學習者(Learner)或決策者(Decision-Maker)的角色稱為代理人(Agent)，而與代理人相互作用的是為環境(Environment)，代理人透過三種不同的訊號與環境相互作用並且學習，分別為狀態(State)，動作(Action)，回饋(Reward)。我們首先定義一些符號:

狀態(State)， s_t**:**

用來表示 t 時間的環境與狀況。$s_t \in S$。S 為在此問題中所有可能狀態的集合。

動作(Action)， a_t**:**

根據當時的狀態，代理人所做的動作。$a_t \in A(s_t)$。$A(s_t)$為一個集合，根據當時的狀態，所能做出動作的集合。

回饋(Reward)， r_t**:**

當作了動作 a 時，環境所賦予一個數值化(numerical)的回饋值，用來評斷此決定的好壞度。

法則(Policy)， $\pi(s,a)$**:**

為一個狀態(state)與動作(action)的函數，為一機率值，也就是規定了每個狀態下該做什麼決定的機率。

價值函數(Value function), $V^{\pi}(s)$:

為在此狀態下以π這個法則往後所得到的總回饋值

代理人接受到此時間點 t 的狀態，s_t，與回饋，r_t，而經由一套學習的法則，決定出該做的動作a_t後，經過環境的反應得到下一個時間點的狀態s_{t+1}與回饋r_{t+1}，再傳回給代理人。如此週而復始的互動，最終目的是最後得到最多的回饋，也是代理人學習機制的目標。用回饋來代表目標的此種構想是增強式學習的一項重要特點。舉例來說，想使一個機器人學習如何走路，操作者在每一個時間點給予一個回饋信號，與機器人前進的距離成正比。或老鼠學習如何走出迷宮，在出口放置乳酪，此時回饋只有在走出迷宮時才得到，其餘皆沒有回饋。這時激勵了老鼠或機器人想盡快走出迷宮，意即，他們總是學著最大化所得到的回饋。所以，如果我們想要代理人，或機器人，甚至動物幫我們做些什

麼事，我們必須設計一個回饋，使得他們最大化所學到的回饋之後，也達成我們的目標。這是增強式學習的目的，也就是利用設計的回饋來實現我們想要的目標。實際上，回饋並不是傳授代理人如何去做的知識，而是與代理人溝通說你想要達到什麼目的，而不是你要如何去達到。

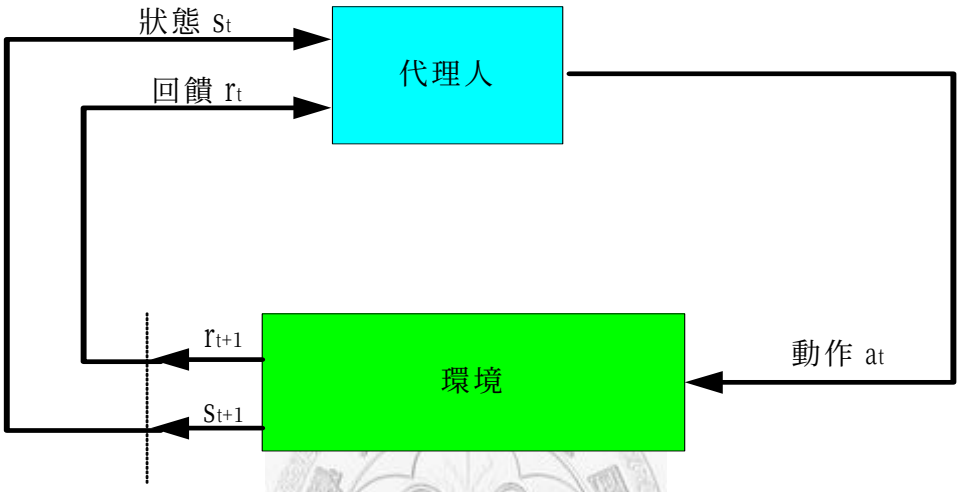


圖 2-6:增強式學習代理人的環境示意圖

接著假設r_T是最終時間所得到的回饋。於是最終的總回饋R_t可以表示成:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + ...r_T \qquad (2.1)$$

但是如果是沒有最終狀態的問題(如本論文所研究的問題)，回饋有可能會發散，於是需要一個折扣因素(discount rate)來阻止最終的回饋到無窮大，此折扣因素也代表了對於未來的回饋值的重視程度，通常此因素，γ，介於 0 與 1 之間。於是有折扣的總回饋可表示成:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \qquad (2.2)$$

另外我們這裡假設是非時變的法則。而增強式學習法的目的就是想學到最大回饋，此時的法則也就是最佳法則，使得每個狀態都是作最佳的決定，造成往後的總回饋可以得到最大。

而狀態價值函數$V^{\pi}(s)$，或動作價值函數(action-value function)$Q^{\pi}(a,s)$的表示如下。假設我們的總回饋是一個有折扣的，無限次的回饋，如下面所示，由於回饋值是一個隨機變數，因此$V^{\pi}(s)$必須取期望值如下:

$$V^{\pi}(s) = E_{\pi}[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} \mid s_t = s] \tag{2.3}$$

同理，$Q^{\pi}(a,s)$也可表示成下面形式:

$$Q^{\pi}(s,a) = E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} \mid s_t = s, a_t = a\right] \tag{2.4}$$

而綜合了以上五種元素，就可以形成增強式學習的數學模型。

3. 增強式學習與動態規劃:

增強式學習有個動態規劃(Dynamic Programming)的遞迴(recursive)特性。此重要的特性，可以讓我們知道增強式學習的本質是靠著學習以往的經驗。式子(2.3)可以展開如下:

$$\begin{aligned} V^{\pi}(s) &= \mathrm{E}\left\langle r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots \mid s_0 = s, \pi \right\rangle \\ &= \mathrm{E}\left\langle r_1 + \gamma V^{\pi}(s_1) \mid s_0 = s, \pi \right\rangle \\ &= \sum_{a} \pi(s,a) \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma V^{\pi}(s') \right] \end{aligned} \tag{2.5}$$

其中，$P_{ss'}^{a}$是指做了動作 a 後，從 s 狀態到 s’狀態的機率，而$R_{ss'}^{a}$則是做了動作 a 後，從 s 狀態到 s’狀態所得到的回饋值。

而最佳的價值函數與最佳法則關係為:

$$
\begin{aligned}
V^*(s) &= \max_{\pi} V^{\pi}(s) \\
Q^*(s,a) &= \max_{\pi} Q^{\pi}(s,a) \\
V^*(s) &= \max_{a} Q^{*}(s,a) \\
\pi^*(s) &= \arg\max_{\pi} Q^{\pi}(s,a) \text{ or } \arg\max_{\pi} V^{\pi}(s)
\end{aligned}
\qquad (2.6)
$$

我們由(2.5)可以清楚看出遞歸的關係，此式子為動態規劃裡的*貝爾曼方程式(Bellman Equation)*。此式可以經由聯立方程式或*法則疊代(Policy Iteration)*解出$V^{\pi}(s)$，但前提是，必須要知道系統的機率特性。

假設$P_{ss'}^{a}$不易得到，或者是時變(Non-Stationary)的，則上述方式不適用。Sutton 提供了一種方法，稱為*差分學習法(Temporal-Difference Learning)*[RsA98]，此法不需要$P_{ss'}^{a}$，而直接作動作，得到一取樣(Sample)回饋。因此我們可以不需知道系統的機率特性，此時估計出的價值函數成為$r+\gamma V(s')$，與先前的 *V(s)*差距稱為差分誤差(TD Error)，如下所示:

差分誤差(TD Error):

$$r+\gamma V(s')-V(s) \qquad (2.7)$$

而學習的法則是依照差分更新法，如下所示:

差分更新(TD Update):

$$V(s) \leftarrow V(s)+\alpha(r+\gamma V(s')-V(s)) \qquad (2.8)$$

其中，α為學習速率(Learning Rate)，代表著對於這次學到的經驗的看重度。由此更新法可以知道，只學習取樣的價值函數與前一個取樣的差距。如果系統是時變的，意即價值函數也是時變的，因此當價值函數改變的時候，取樣的結果也會有所改變，此時經過更新之後，會使得新的取樣更接近改變後的價值函數。於是如果更新(學習)的速度夠有效率的話，那麼此方法將可以應用在時變環境中。

同樣的，Q(s,a)也可以用此種更新法來估算，在文獻中有個成功的差分學習演算法，稱為 Sarsa [RsA98]。如圖 2-5 所示。他是一種法則性(On-policy)的學習，法則性是指選取每個動作都依循著某一特定法則。此法則為一個 S→A 的機率值，意思是即使相同的狀態下每次選取動作有可能不一樣，此方法可以稱為探索(exploration)。

Initialize Q(s, a) arbitraril y
Repeat(for each episode)
Initialize s
Choose a from s using policy derived from Q
Repeat n :
Take action a, observe r, s'
Choose a' from s' using policy derived from Q
$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma Q(s', a') - Q(s, a)]$
$s \leftarrow s' ; a \leftarrow a' ;$
until s is terminal

圖 2-7:Sarsa 演算法

4. 探索(Exploration):

對於動作的選取(action selection)，增強式學習存在一個值得探討的問題，則是探索(Exploration)。探索意思是，去沒去過的地方，做沒做過的動作，這樣的理由是，可以使得我們真正找到一個最佳的答案。如圖 2-8 所示，假如這問題是想達到最低點(minimal)，然而每次都是找最低的情況下，有可能會掉進區域性的低點(Local minimal)，因為在區域性低點的周圍，已經找不到比他更低的點，於是就以為達到整個系統的低點。此時就需要探索的方法，來找到其他的低點，我們用 $\varepsilon-greedy$ 法，此法是指有個很小的機率 ε，去作隨機(random)的動作。然而總不能一值探索，探索的次數應該隨著時間增加而減

少。這樣才能保證收斂到最佳值[TMC00]。在此論文中，對於探索方式的研究不多做探討，我們對ε參數的設計沒有多做研究，沿用了一些經驗值來代替。

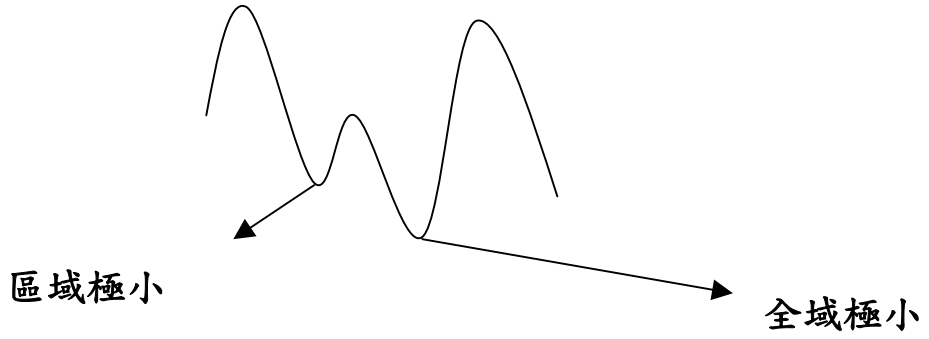


圖 2-8:區域極小與全域極小

5. 結論:

增強式學習有與動態規劃相同的特性，可經由法則的疊代更新而去近似最佳的法則與價值函數。且不像動態規劃法，不需要系統的狀態轉移機率。他是持續的改善效能，學習經驗藉由與環境的互動來估算價值函數。因此，我們更可以推測增強式學習最終可以得到動態規劃的解。而我們考慮的派工問題，由於派工視為一種決策過程，動作為選取下一個產品，狀態為產品的資訊與機台的狀況。而派工的法則則是，在什麼狀態派工什麼產品。。預期學習的輸入與輸出如圖 2-9 所示。

總結增強式學習的優點如下:

a. 對於無法瞭解的環境，可以使用學習經驗來得到不錯的結果。

b. 不侷限於教導者給予的例題，可以自我學習與外在的互動。

c. 結合模擬的觀點，讓學習更具有真實性。

綜合本章，我們認為增強式學習適用於派工問題的原因是:

1.現今的派工知識多是依賴經驗法則，增強式學習正是學習經驗的方法。

2.現場環境多變，且是隨時間改變，必須持續不斷學習改進派工法則。

3.增強式學習有別於一般學習法，是以嘗試錯誤法來持續改進。

4.增強式學習為一種有理解力與自動化的目標導向式學習，以及決策與學習的計算式(Computational)的方法。

5.增強式學習結合模擬與學習的能力。

接下來，我們將針對此兩類類問題做更深入的了解，與用增強式學習設計實驗來驗證是否可行。

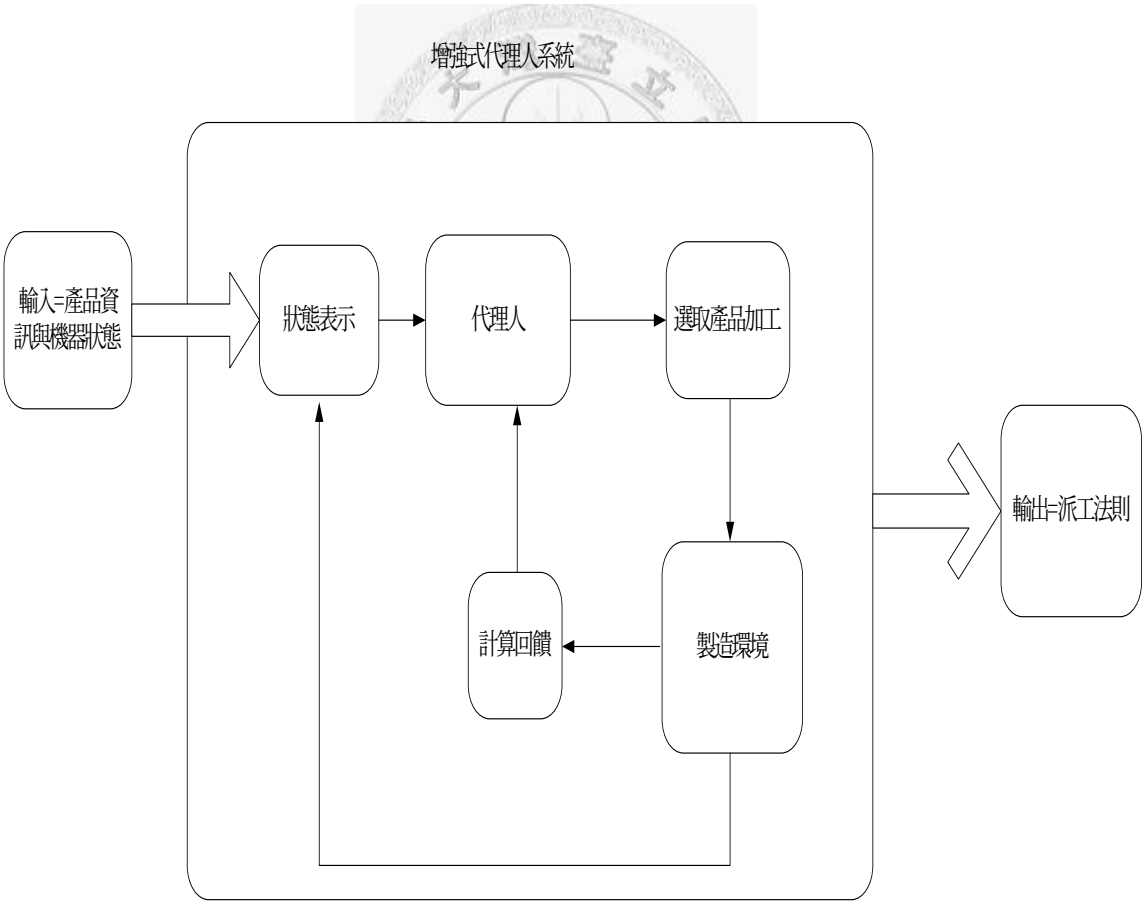


圖 2-9:派工問題的增強式學習之輸入與輸出

第三章

有設置時間單一機台的派工問題

此章我們將討論有設置時間的單一機台派工問題，首先我們將此問題作數學描述，近似成非時變的連續時間的馬可夫決策過程。並將此問題用法則疊代法解出最佳解。然而我們也更改了增強式學習的演算法使得適合應用於此派工問題，並顯示出隨著學習次數增加，可以越來越接近法則疊代的最佳解。最後我們將增強式學習應用在時變的派工問題上，並與一個隨機的法則來做討論與比較，結果顯示，即使是在時變的環境中，增強式學習也能使平均等待時間趨向穩定。

在 3.1 節我們將此派工問題近似成馬可夫決策過程，3.2 比較增強式學習與法則疊代的解。3.3 考慮時變的環境，而在 3.4 討論增強式學習應用於時變環境的成效。

3.1 派工問題之馬可夫決策過程數學描述

1.派工問題描述:

如 2.1 章所介紹過，我們這裡考慮有設置時間的單一機台派工問題，此機台是一個可以做不同製程的機台。而不同的產品有不同的到達率(arrival rate)和處理率(processing rate)，且等候區為有容量限制，另外在做不同產品的時候必須花費設置時間(Setup Time)來轉換製程，於是我們設計一個回饋函數(Reward Function)來代表此種時間效能，於是我們的目標(objective)就是使得產品的等候時間最小(minimize the waiting time)。

為了使問題能近似馬可夫決策過程，而派工的狀態是根據現場狀態做決定，我們可以把此狀態描述成具有馬可夫性質(Markov property)的狀態，也就

是狀態的改變只與前一次狀態有關。而所有的機率分布也都假設成指數分布(Exponential distribution)。我們做了以下假設:

1. 機器同一時間只能做同一件產品。
2. 產品到達時間為指數機率分佈。
3. 產品處理時間為指數機率分佈。
4. 機器設置時間為指數機率分佈。
5. 忽略產品送往機器之間的時間。
6. 每個等候區有最大容納限制。
7. 達到等候區最大容量時，下一個產品會送至其他機台工作，即等候區總數不再增加。
8. 忽略兩個事件(Event)同時發生的機率，即兩個事件發生的機率為零。

2.連續時間馬可夫決策過程(Continuous-time Markov Decision Process):

考慮單一機台 n 種製程的問題，由於派工是根據現場狀態做決定，且我們假設狀態的改變只與前一次狀態有關，所以我們可把此派工問題表示成一個馬可夫決策過程(Markov Decision Process)。

一開始我們先定義一些會用到的參數:

n : 最多有 n 種製程。

i : 產品製程種類，1~n。

μ_i: 製程 i 平均處理完畢率，時間為指數分佈。

λ_i: 製程 i 平均到達率，時間為指數分佈。

ν_s: 平均設置完成率，時間為指數分佈。

C_w^i: i 產品的等候時間權重。

派工指的是，在機器有空出來的時候，選擇派哪個產品去加工。而機器加工產品時間為一個隨機變數，一個派工的決定需要等到下一個產品加工完畢，依照當時狀態所做決定。當派工決策與決策之間的時間間隔為一個獨立的指數分布變數時，我們可以將派工問題形成一個連續時間的馬可夫決策過程。然而在決策與決策之間也會有可能會產生狀態的變化(如圖 3-2)，因此問題的狀態轉移過程會變得複雜，在這裡我們改變做決策的時間點，也就是每改變一個狀態時，就必須做決策，則在真實的情況不用作決策的時間點，此時的決策為不做任何事，使得此問題可以模擬成一個標準的連續時間馬可夫決策過程。然而有設置時間的單一機台派工問題近似成一個標準的馬可夫決策過程具有下列要素，列舉如下:

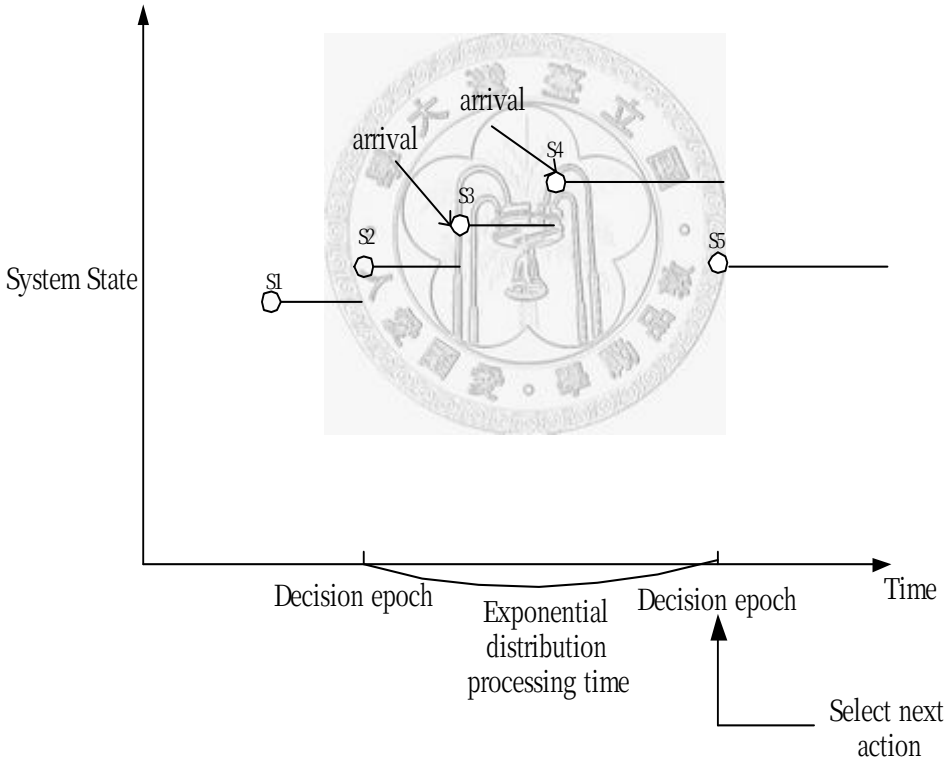


圖 3-1:狀態轉移路徑

a. 狀態(State):

由於派工問題是依照當時狀況來作決定，很自然地，我們考慮的在 t 時間的狀態(State)，表示如下:

$$S(t)=\{N,M,LaT\} \quad (3.1)$$

而狀態裡面變數的詳細定義如下:

N ： 為一個向量變數，N={N1,N2,…Nn}。

N1 ： 製程 1 等候區裡面的個數，為 0~K_T 的整數。

N2 ： 製程 2 等候區裡面的個數，為 0~K_T 的整數。

Nn ： 製程 n 等候區裡面的個數，為 0~K_T 的整數。

M ： 機台的狀態，1 代表正在處理某產品中，0 表示閒置中。

LaT: 表示在下一個決定前，機台處理過的產品種類，為 1~n 的整數。

b.可允許的決策 (Admissible Actions):

我們需要作的決策是依照當時狀況，決定要派哪種產品進機器加工，而產品在等候區是依照到達的順序排列，沒有緊急加工產品的狀況。決策隨著狀態不同而不同，令每個狀態可做的決策 a(s)=u，$u\in\{0,1,2...n\}$，0 代表等待下一個產品，1~n 代表著選擇對應的製程加工。依照狀態可做的決策大可分為兩類，當機器是閒置的時候，可以做派工;或機器正在加工某類產品及沒東西可做，只好選擇等待下一個產品。而機器閒置的時候，可依每種製程的產品有無排列組合再扣掉全部都沒有產品的情況，分為 2^n-1 種情形，每種情形的可允許決策皆不同;分別是 n 類產品皆存在其等候區 s=(N1,N2..Nn,0,LaT)，此時 a(s)=1 or 2 or 3…n;只有某 i 類在等候區，此時 a(s)=i 或 0;以此類推有兩類，三類的情況。

因此定義一個可允許的決定(admissible action)集合$\Phi(s)$，而每個決策$a(s)\in\Phi(s)$。且定義一個集合 u’ ={$\arg_i N(i)\neq 0$}，代表所有 N 向量裡面非零元素的所在位置，因此可允許的決定可以歸納如下:

$$\begin{cases} N1\neq 0，N2\neq 0，\cdots Nn\neq 0，s=(N,0,LaT)，\Phi(s)=\{\ u'\ \} \\ N\text{ 至少有一個元素為零，}s=(N,0,LaT)，\Phi(s)=\{u'\ ,0\} \\ N=0，s=(N,M,LaT)，\Phi(s)=\{0\} \\ s=(N,1,LaT)，\Phi(s)=\{0\} \end{cases} \tag{3.2}$$

比如 n=2 時，當 s=(1,1,0,1)，$\Phi(s)$ ={1,2}; 或 s=(3,0,0,2)時，$\Phi(s)$={1,0};s=(1,2,1,1)，$\Phi(s)$={0}。

c. 狀態轉移機率(State-Transition probability):

定義在狀態 s，作決策 a，經過 t 時間到達狀態 j 的機率為 Q(t,j|s,a)，則

$$Q(t,j|s,a)=P(j|s,a)F(t|s,a) \tag{3.3}$$

P(j|s,a)為狀態 s 作決策 a 到狀態 j 的機率，F(t|s,a)為狀態 s 作決策 a 經過 t 時間會改變到下一個狀態的機率。

因為 State 的轉移是由於某些特定的事件(event)發生，我們的事件集合(event set)E 為:

E={製程 1 產品到達，製程 2 產品到達，…製程 n 產品到達，製程 1 產品加工完畢，製程 2 產品加工完畢,…製程 n 產品加工完畢，製程轉換設置完畢}

在兩個決策之間，由於產品的到達，與產品的加工完成，與設置的完成，會有狀態的變換，狀態變數 N 是由於產品的到達而改變，M 是由於產品的加工完畢而改變，LaT 是由於處理產品不同而改變，然而不同狀態下可以做的決策(admissible action)不同，因此狀態轉移圖(state transition diagram)必須分類來討論。主要依據狀態內變數的不同特性及可允許的決策來分類，而下一個發生的事件決定下一個狀態，總共可比照(3.2)再分為四大類，詳細的圖請見附錄:

Case-1: { N1≠0，N2≠0，…Nn≠0，s=(N,0,LaT)}

這是在機台空出來的狀況下，依(3.2)此時可以做的決策有 n 種，即選擇 1~n 的製程產品;例如在 LaT=1 時的狀態，做了 a(s)=1 時，亦即把製程 1 產品送進機器加工，此時下一個會發生的事件有 n+1 種:製程 1 產品到達，製程 2 產品到達，..製程 n 產品到達，及製程 1 產品加工完畢，而 n+1 種事件發生的時間間隔分佈皆為指數分佈，假設為 T1,T2,…T_{n+1}; 若下一個事件發生的時間為 Te，則 Te 必滿足:

$$Te=\min\ (T1,\ T2,\cdots T_{n+1}\ ) \tag{3.4}$$

經過計算可知，Te 為參數$\{\lambda_1+\lambda_2+....+\lambda_n+\mu_1\}$的指數分佈。所以轉移率(transition rate)為$\{\lambda_1+\lambda_2+....+\lambda_n+\mu_1\}$，所以在這個狀態 s，作 a(s)=1，經過 t 時間到下一個狀態的發生的機率 F(t|s,a)為:

$$F(t|s,a)=1-e^{-(\lambda_1+\lambda_2+...+\lambda_n+\mu_1)t} \tag{3.5}$$

而每個事件發生的機率，如製程 1 產品到達的機率為

$$P\{T1<\min(T2,T3,\cdots T_{n+1})\}=\frac{\lambda_1}{\lambda_1+\lambda_2+...+\lambda_n+\mu_1} \tag{3.6}$$

假設上面事件發生會到達狀態 j，則 P(j|s,a)等於(3.6)式。以此類推可將每個狀態的 P(.)，F(.)個別算出，而求出 Q(.)。結果如圖 3-3 所示;在{.}前面代表是轉移率，後面代表的是轉移機率，兩者相乘即是狀態之間的轉移率。請注意若在 a(s)≠1 的時候，下一個發生的事件則是 n 種產品的到達，與設置完成的事件。同以上的推理可以得到作其他 n 種決策的狀態轉出圖，也可推得 LaT=n 的結果。(請參見 appendix)

Case-2 { s=(N,1,LaT) }

在 M=1 的時候，也就是機器正在加工某一產品時，此時我們只能做 a(s)=0 的決策，也就是什麼都不做的等待下一個事件發生，而下一個事件也是有 n+1 種:製程 1 產品到達，製程 2 產品到達，..製程 n 產品到達，與正在加工的產品完工。如圖 3-4 所示為 LaT=1 的狀況。同理也可以推得其他 LaT 的轉出情況。

Case-3 { N 至少有一個元素為零，s=(N,0,LaT)}

在 N 至少有一個元素為零的時候，我們可以決定要不要繼續等待，或者直接切換做另一個產品。作另一個產品的轉出圖，就與 case-1 一樣，而選擇繼續等待的情況下，下一個可能狀態就是各種產品的到達。

Case-4 { N=0，s=(N,M,LaT)}

在各個等候區裡面都沒有存貨的狀況下，這時候只能作等待。而下一個可能事件的發生為各種產品的到達。

另外，在有某個產品的等候區已滿的情況下，由於我們假設會派去別的機台工作，因此對於以上的各種情況若有等候區滿的情形，則下一個可能的狀態會有點不同，主要是扣除有超過等候區最大容量的狀態發生。

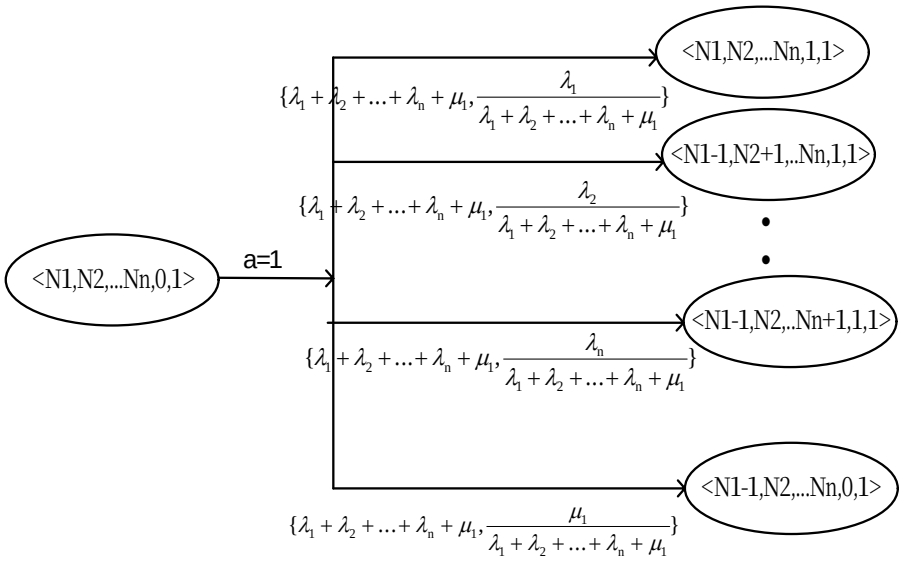


圖 3-2:Case-1, a=1 狀態轉出圖

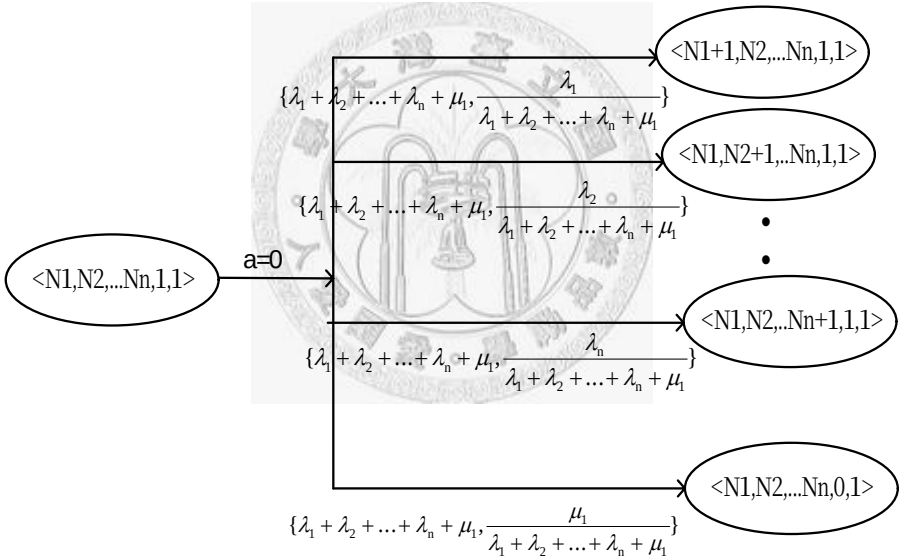


圖 3-3:Case-2, a=0 狀態轉出圖

d. 目標函數(Objective function)

由於我們的目的是要使平均等待時間最小，且每種產品有各自的加權數，乘上有幾個產品在等候區，即是我們的等待成本。由於是連續時間的問題，我們的折扣因素(discount factor)必須為一個 $e^{-\beta t}$ 的值才不會使最後總花費發散。則我們的總單位時間權重等待成本(Total weighted waiting cost per unit time)，如下所示:

$$c(s,a)=\sum_{i=1}^{n}[N(i)\times C_w^i] \qquad (3.5)$$

而 0~τ 秒間期望花費(expected cost) 如下所示:

$$r(s,a) = E_s^a\{c(s,a)\int_0^{\tau} e^{-\beta t}dt\} \qquad (3.6)$$

此 E 是對 t 做期望值，因為 t 為一個隨機變數，也就是狀態轉移的時間。展開如下:

$$r(s,a)=\int_0^{\infty}\sum_{j\in S}[c(s,a)\int_0^{u} e^{-\beta t}p(j|t,s,a)dt]F(du|s,a) \qquad (3.7)$$

因此假設期望值在無窮時間後，存在有值的話，總成本可如下所示:

$$V(s) = E_s^a\{c(s,a)\int_0^{\infty} e^{-\beta t}dt\} \qquad (3.8)$$

若經過τ時間到下一個狀態，則可以表示成:

$$\begin{aligned} V(s) &= r(s,a) + E_s^a\{e^{-\beta\tau}\,V(s')\} \\ &= r(s,a) + \sum_{j\in S}\int_0^{\infty} e^{-\beta t}Q(dt, j|s,a)V(j) \end{aligned} \qquad (3.9)$$

令 $m(j|s,a)=\int_0^{\infty} e^{-\beta t}Q(dt, j|s,a)$ ，而我們的目標函數如下所示:

$$V(s)=\min_{a\in\phi(s)}\{r(s,a)+\sum_{j\in S}m(j|s,a)V(j)\} \qquad (3.10)$$

綜合以上，我們將一個派工問題近似表示成一個馬可夫決策過程。將現場產品資訊與機器狀況表示成馬可夫狀態，而把產品到達時間，處理時間，設置時間假設為指數分布。將派工的決策依照狀態分類，也討論了轉態轉移的變化。因此定義了一個馬可夫決策過程，可使得:1)在非時變的狀況下，以法則疊

代法解出最佳解，2)馬可夫決策過程為增強式學習問題的一種基本環境[RsA98]。

3.2 單一機台，兩種製程:增強式學習之解

現在我們已經有了派工問題的馬可夫決策過程模型，接著將考慮使用增強式學習的方式來嘗試解此問題，在這裡我們考慮單一機台，兩種製程分別為A,B 的問題(如圖 3-1)。

由於 Sarsa 演算法為離散時間，且有目標終止狀態，對於派工問題來說，時間是連續的，且沒有一個終止的狀態。也就是說我們必須做一些改變，使得此演算法適合解我們此項問題。

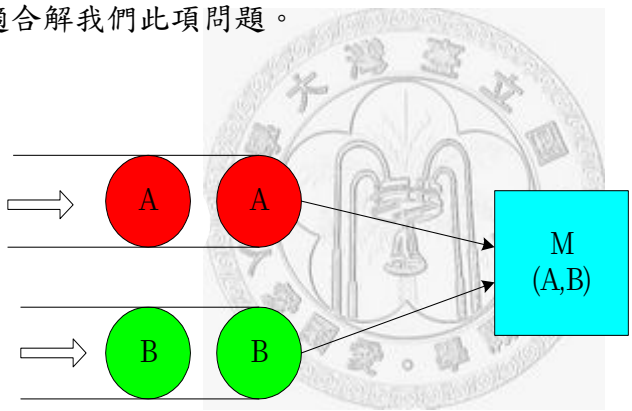


圖 3-4: 單一機台兩種製程派工示意圖

1.連續時間的差分學習法:

現在考慮增強式學習法裡的價值函數，由於我們的問題是連續時間的馬可夫決策問題，且為一個有限花費，無限時間的且沒有穩態的問題。因此我們可以定義每一個狀態的價值函數:

$$V^{\pi}(s) = E\{\int_0^{\infty} e^{-\beta t} c(s,a)dt\} \qquad (3.9)$$

若我們假設到達下一個狀態 s’ 的時間為τ，則在 E{}裡可以寫成:

$$\int_0^{\tau} e^{-\beta t} c(s,a)dt + \int_{\tau}^{\infty} e^{-\beta t} c(s',a)dt = \frac{1-e^{-\beta\tau}}{\alpha} c(s,a) + \int_{\tau}^{\infty} e^{-\beta t} c(s',a)dt$$

$$= R(s,a) + \int_0^{\infty} e^{-\beta(t+\tau)} c(s',a)dt \qquad (3.10)$$

$$= R(s,a) + e^{-\beta\tau} \int_0^{\infty} e^{-\beta t} c(s',a)dt$$

所以可以得到:

$$V^{\pi}(s) = E\{R(s,a) + e^{-\beta\tau} \int_0^{\infty} e^{-\beta t} c(s',a)dt\}$$

$$= E\{R(x,a)\} + e^{-\beta\tau} E\{\int_0^{\infty} e^{-\beta t} c(x,a)dt\} \qquad (3.11)$$

$$= E\{R(x,a)\} + e^{-\beta\tau} V^{\pi}(s')$$

由第二章可以得知，方程式 3.7 式中，可以改寫成 TD-update 的形式:

$$\frac{1-e^{-\beta\tau}}{\alpha} c(x,a) + e^{-\beta\tau} V(s') - V(s) \qquad (3.12)$$

另外，在實際上的工廠並沒有終止穩態(terminal state)，而 Sarsa 演算法停止的準則，將可以改成指定學習的次數後停止。於是假設一個最大學習數 MaxN，當學習的次數到達此數之後，此演算法停止。在這裡我們定義學習的次數是每次的狀態轉移為一次。

綜合以上可以得到連續時間的 Sarsa 演算法(見 Appendix)。使用學習的好處在於，不用事先知道機率特性。Sarsa 演算法中，沒有用到系統的機率特性，意即假設我們對於此系統的產品到達率，處理率等等皆一無所知。

2.基本可行性評估:

以下的實驗想證實增強式學習可以應用於模擬派工問題的馬可夫決策過程。在以非時變馬可夫決策過程表示的派工問題，我們比較了法則疊代[Appendix A]與增強式學習的解。法則疊代法已被證實可以找到非時變的最佳法則。前提是此問題必須存在一個非時變的法則(Stationary policy)[Ber95]，因此我們這部份的環境是非時變的。而連續時間的馬可夫決策過程的解法，需要經過一致性(Uniformization)轉換[Appendix B]，化為一個符合離散時間(discrete time)的馬可夫決策過程，再運用法則疊代(policy iteration)可解出最佳解。所以我們將上述的派工問題轉為離散時間之後，再經由法則疊代解出最佳解。而我們的目的為 1.正確性，2.比較計算時間。

而我們採用的參數值如下:

$\lambda_A = 1.2$ $\lambda_B = 1.5$ $\mu_A = 2$ $\mu_B = 1.6$ $\nu_s = 3$ $\alpha = 0.1$ $\beta = 0.11$ $C_w^A = 1$ $C_w^B = 2$ $K_T = 3$

初始狀態 S_0=[1,1,0,1]

驗證:

為了要證實法則疊代法的正確性，我們先設計一個特例:

假設產品 A，B 的到達率皆很小，即很久的時間間隔才有會產品到達，因此最佳解應該是見一個做一個，且傾向把每個製程都做完才轉換製程。結果如表 3-1 顯示，將資料做了整理，把只做 a(s)=0 的去掉，只探討可以選擇不同 a(s)的狀態。以下的結果可以看出，有一邊等候區是零的狀況下，最佳的法則都是直接做，不等待另一個來臨，而在機器設置部分，可以看出最佳的法則都是盡量避免機器設置，也就是把每個製程都做完，再換作其他製程。

因此我們的法則疊代解法是正確的。於是，可以放心的使用法則疊代法來作為增強式學習法的比較基準。

	State			
optimal policy	Na	Nb	M	LaT
2	0	1	0	1
2	0	1	0	2
2	0	2	0	1
2	0	2	0	2
2	0	3	0	1
2	0	3	0	2
1	1	0	0	1
1	1	0	0	2
1	1	1	0	1
2	1	1	0	2
1	1	2	0	1
2	1	2	0	2
1	1	3	0	1
2	1	3	0	2
1	2	0	0	1
1	2	0	0	2
1	2	1	0	1
2	2	1	0	2
1	2	2	0	1
2	2	2	0	2
1	2	3	0	1
2	2	3	0	2
1	3	0	0	1
1	3	0	0	2
1	3	1	0	1
2	3	1	0	2
1	3	2	0	1
2	3	2	0	2
1	3	3	0	1
2	3	3	0	2

表 3-1:特殊例子的法則疊代解

接下來使用增強式學習法來學習，我們所採用的是 Sarsa 演算法，並設計價值函數一開始皆為零，且初始狀態皆與法則疊代所使用的一樣。實驗顯示出，決策的正確性，隨著學習次數的增加，持續往上增加(如圖 3-5)。這意味著，如果學習的次數夠多，將有一定程度的最佳解。而價值函數與最佳價值函數的誤差值，也如圖 3-6 所示，隨著次數增加，誤差明顯的下降。也就是說，此學習法在應用派工問題中，有近似最佳的解。但是學習次數不夠有效率。

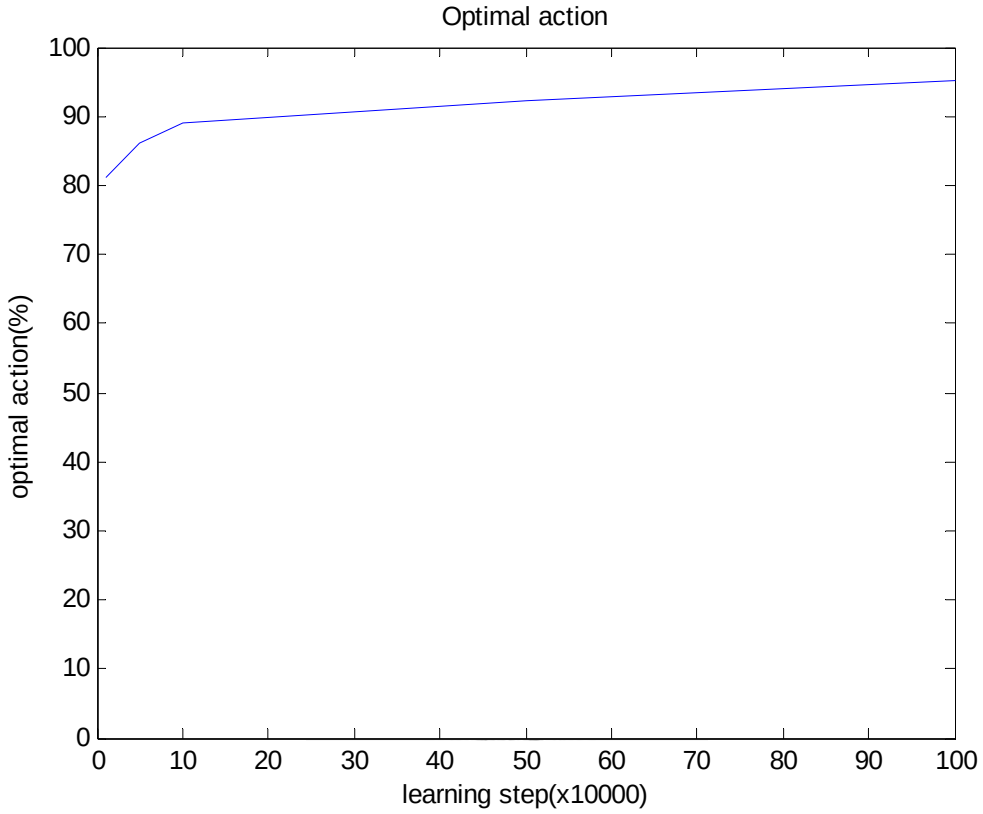


圖 3-5:最佳決策率與學習次數關係圖

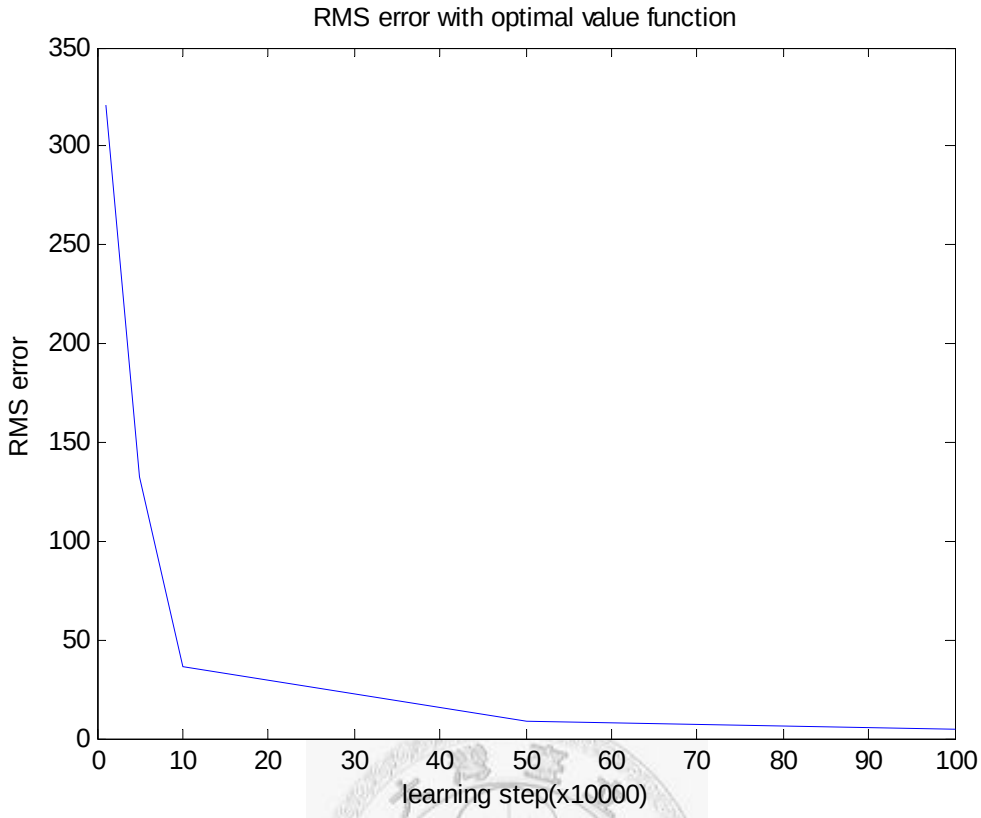


圖 3-6:與最佳價值函數的均方根誤差與學習次數關係圖

3.與啟發式法則的比較:Clearing Policy

接著考慮在[KuS90]所提到的啟發式法則，清空法則(clearing policy)。這個法則是說，延續上一次的製程為主，將此製程的產品全作光，再換到下一個製程，也就是盡量的減少設置次數。對於我們的問題而言，表 3-1 就顯示出清空法則的規則。

我們的做法為:使用此清空法則，與給定等效的清空法則的價值函數作為增強式學習的初始條件，例如 s=[2,2,0,1]，此時應該是作 a(s)=1，因此就令 Q(s,1)<Q(s,2)，以此類推。藉此顯示出增強式學習是否能學出比此清空法則更好的法則，也可看出在我們這個問題中，清空法則與最佳法則的差異性。

實驗結果如圖 3-7 所示，我們使用了 10 萬次的暖機時間，並將彼此的初始狀態設為相同，來比較平均權重等候時間 T_w。定義如下:

$$T_w = [\sum_n (C_w^A \times Na + C_w^B \times Nb)\tau_n]/(totaltime) \qquad (3.13)$$

τ_n 為第 n 次狀態轉移的時間，所以 T_w 即等於(在 t 秒內總權重時間)/t，即是代表了平均權重等候時間。

圖中明顯的看出增強式學習一開始與 Clear 法則效能差不多，經過學習次數增加後，增強式學習有逐漸下降的趨勢。

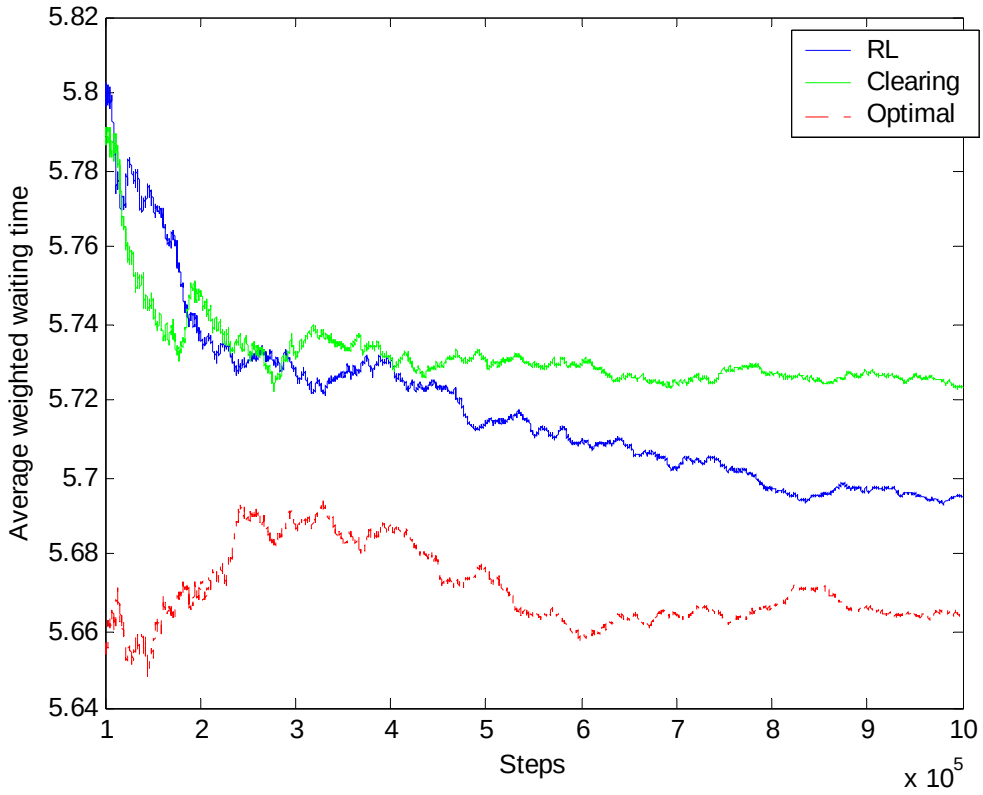


圖 3-7:增強式學習與 Clearing 法則，最佳法則的比較

3.3 具有時變的產品到達率的派工問題

然而考慮到現實層面，產品的需求量是非常不確定的，且是難以預測。在大多數的產業中，產品的需求即代表著訂單的來臨，這無法僅僅以簡單的機率分佈表示。這通常是時變的(Non-Stationary)。許多市場需求的歷史行為，都類似一種隨機行走(random walks)[Gra98]。他會隨著時間演變，而隨機的改變速率成長的方向和大小。此外，當產品的生命週期比較短的時候，此需求的不確定性與隨機性變得更難預測。它往往受到一些外在的因素影響，比如景氣的好壞，產業的發展性等等。所以產品到達率通常是一種時變的分佈，亦即無法使用法則疊代法來求得最佳法則。

我們使用了時變的波松分布描述到達率。我們假設到達率可以用以下表示:

$$\lambda(t) = \lambda_0 + \chi_t \qquad (3.14)$$

此時$\lambda(t)$隨著χ_t改變而變，λ_0為一個常數，χ_t為一個隨機變數。隨著每次狀態的改變，而產生不同的值。在這裡我們為了讓$\lambda(t)$永遠為一個正數，所以χ_t的分佈必須要大於等於$(-\lambda_0)$。修剪式的常態分布(Truncated Normal distribution)可以保證在範圍[a ,b]內取值。在這裡的$\lambda(t)$設計，即是將χ_t設計成一個$[-\lambda_0, \lambda_0]$的修剪式常態分布，其平均值(mean)為 0，變異數(variance)為 1，表示成 truncated_normal[a,b,0,1]。因此$\lambda(t)$隨著時間會在$[0, 2\lambda_0]$範圍裡變動。

3.4 時變問題數值實驗與討論

接下來設計了一系列的實驗，來驗證增強式學習是否有能力針對時變的系統，做持續的學習與效能改善。而我們採用三種指標評估系統效能:

1.平均權重等候時間(Average weighted waiting time)，T_w:

2. 相同狀態轉移次數下，所獲得的產量。

3. 相同狀態轉移次數下，所獲得的設置的次數。

我們選用隨機派工法則(Random Job Selection)來作為對照，隨機法則即是，以相同機率隨機的選擇工作。將可以看出增強式學習法在此三種效能評估中，對於時變環境的學習效果。

以下分別使用了三個不同 Case(如表 3-1)，為了了解時變到達率的影響，其中只改變到達率，其他參數如前一節的設定。每個狀況都有 10 萬次的暖機(Warm-up)時間，以避免初始狀態所造成的影響，而初始狀態則為各有一個產品，機器狀態為設置在產品 A，如下所示:

S_0=[1 1 0 1]

結果討論:

Case a-1:有時變到達率的產品 B，包含極小 λ_B

由圖 3-8 可看出增強式學習比隨機法則的結果好許多，不論在平均權重等待時間上，或者是產量與切換次數，都有著比隨機法則好的效能。平均權重等待時間上，增強式學習大約少了一倍左右的時間。產量的部分，增強式學習則是提升了約 30%的產出。切換次數也有較少的趨勢。而我們注意到等待時間在學習次數的二到六十萬之間有下降的趨勢。這是因為此時波松到達率剛好到達

一個很小的值(見圖 3-9)，也就是說此時 B 產品的到達時間間隔變得很長。因此在此時，只有 A 產品會到達，所以會有平均權重等候時間下降的現象。但是這樣不是一個正常的現象，也就是說，到達率不應該極小到接近零。現實情況中，應當不會有這麼長的一段時間沒有貨來臨，所以我們將 B 的到達率 $\lambda(t)$ 稍微做了一下改變，如 Case a-2 所示。

Case a-2:有時變到達率的產品 B，沒有極小 λ_B

如圖 3-13，到達率呈現相當高度的變化，且沒有出現很長一段時間極小的狀況。而此時的平均等待時間由圖 3-12 可看出，增強式學習的平均等待時間還是比較小，且隨機法則在次數二十萬左右逐漸往上，而增強式學習則保持在穩定的狀況下。這顯示出，增強式學習可以隨著環境變動，持續的作調整，使得派工的法則皆能讓等待時間達到穩定。而在產出的部分也增加了 30%(如圖 3-14)。而如圖 3-15，切換次數的部分，則可以明顯看出，增強式學習皆比較小，在十萬到五十萬之間，增強式學習切換次數增加了七倍多，而隨機法則則增加了十倍多。區段性的可以看出，增強式學習學到了如何降低切換次數。

Case b-1:有時變到達率的產品 A 與 B

在這個情形下，將產品 A 與 B 的到達率都改為時變。見圖 3-17，3-18，這情況下，平均等待的時間皆比前兩個情況還來的小。也就是說，兩個產品都是時變的狀況下，等待時間會比較小。在此情形下，得到的結果也是增強式學習比較好。且等待時間上差了四倍多，意即，增強式學習在較高度的時變環境中，較能顯現出學習的效果。

	λ_A	λ_B
Case a-1	1.2	$\lambda(t)$=1.5+χ_t， χ_t~truncated_normal [-1.5,1.5,0,1]

Case a-2	1.2	$\lambda(t)$=1.5+χ_t ， χ_t~truncated_normal [-1.499,1.5,0,1]
Case b-1	$\lambda(t)$=1.2+χ_t ， χ_t~truncated_normal [-1.199,1.2,0,1]	$\lambda(t)$=1.5+χ_t ， χ_t~truncated_normal [-1.499,1.5,0,1]

表 3-2:實驗數據

Case a-1:

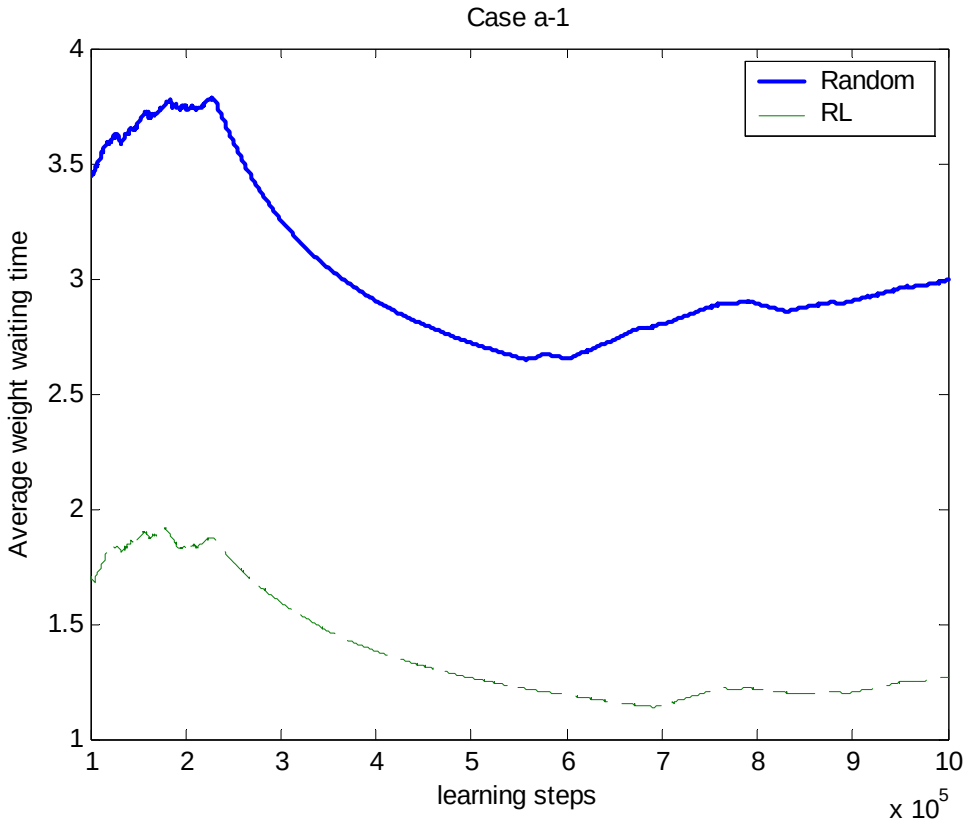


圖 3-8:Case a-1 平均權重等候時間比較圖

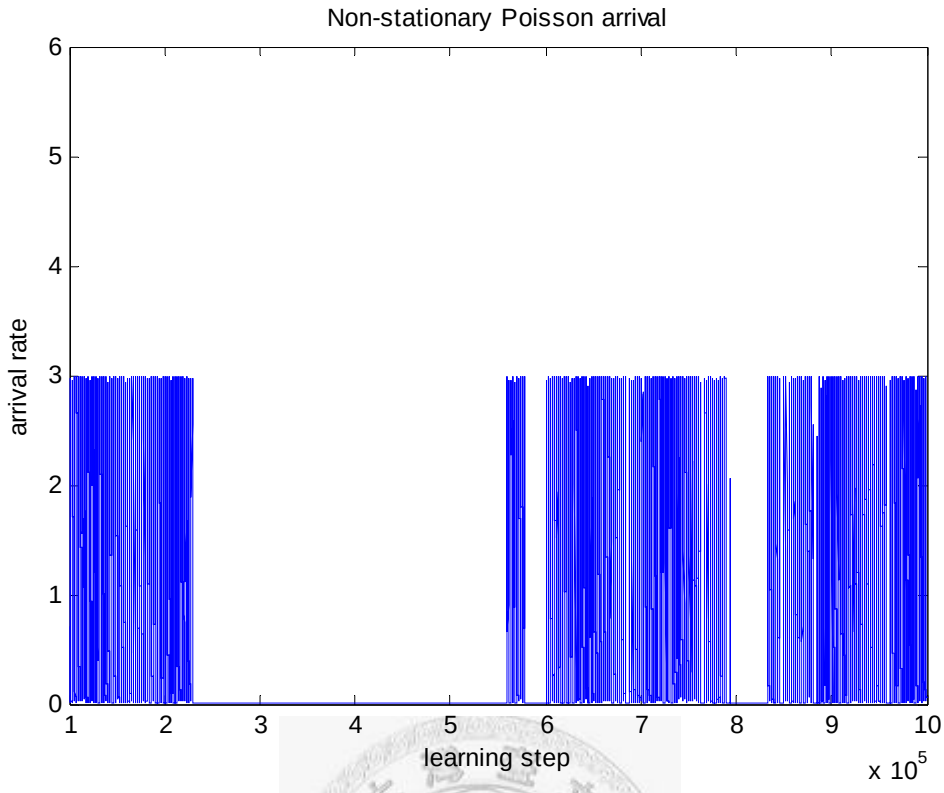


圖 3-9: Case a-1 時變到達率

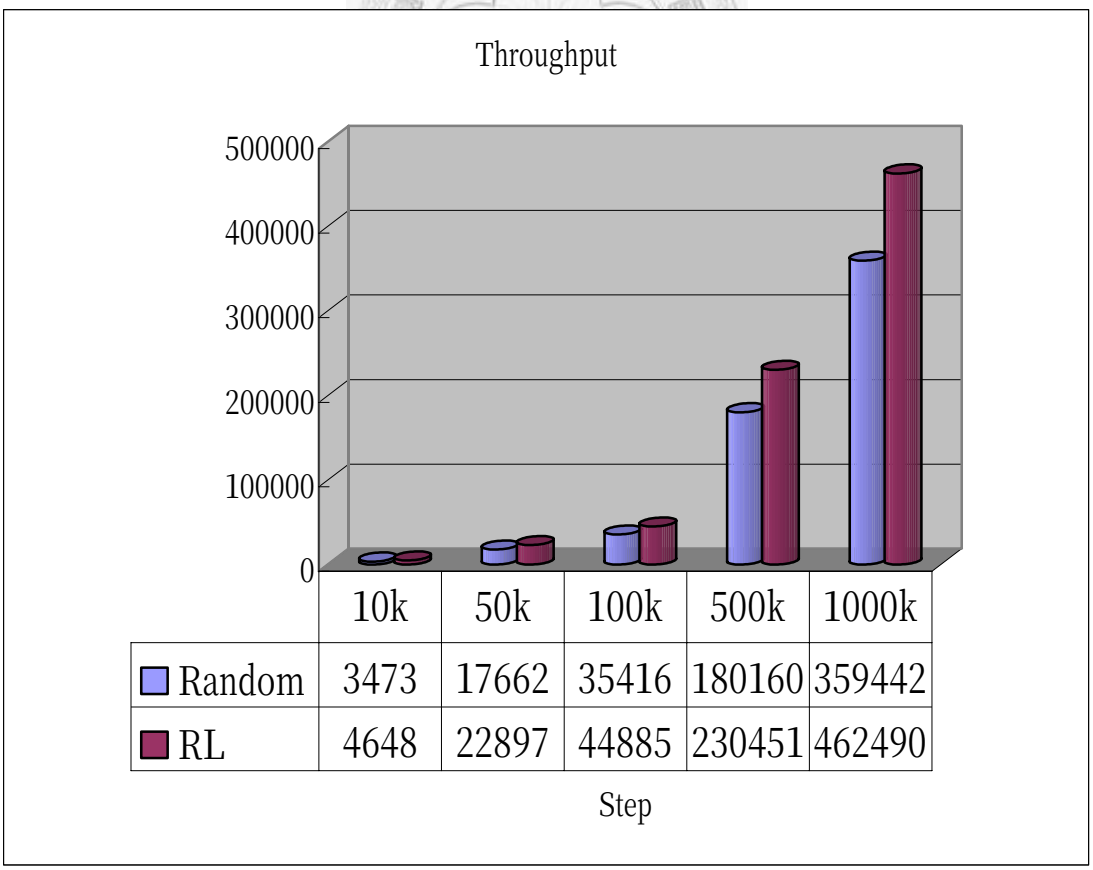


圖 3-10: Case a-1 產量比較圖

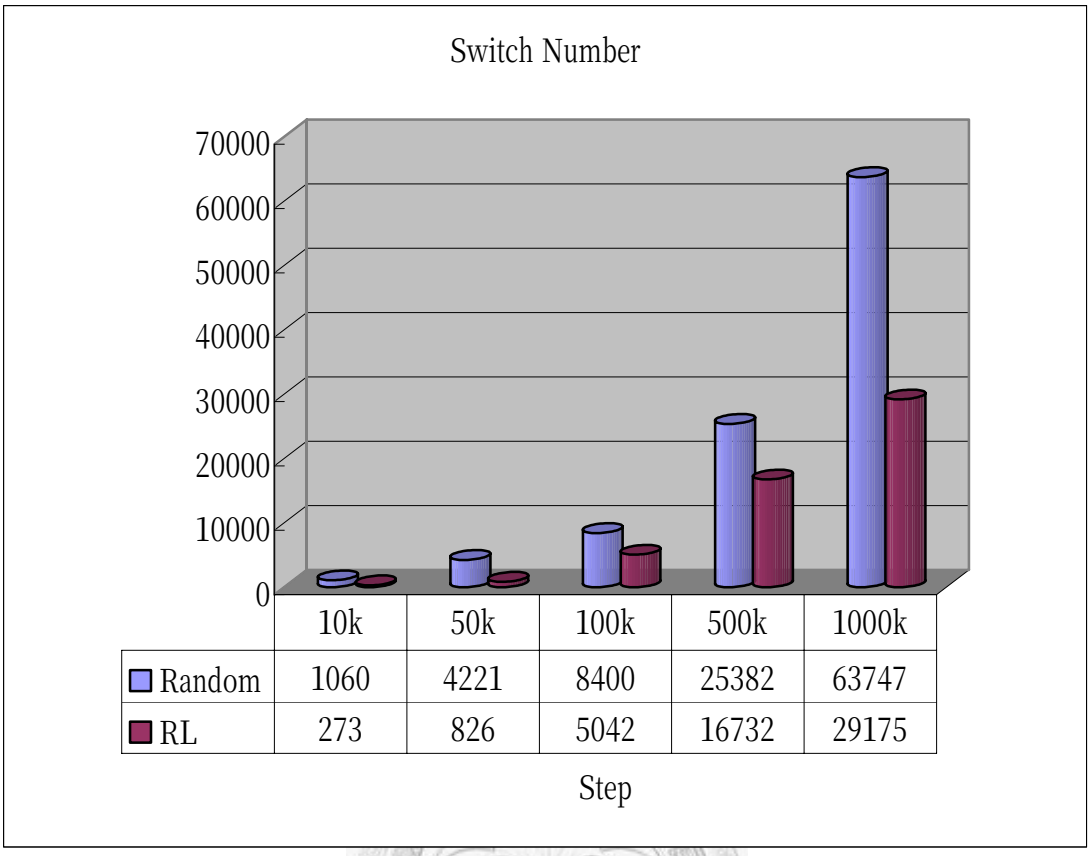


圖 3-11: Case a-1 切換次數比較圖

Case a-2:

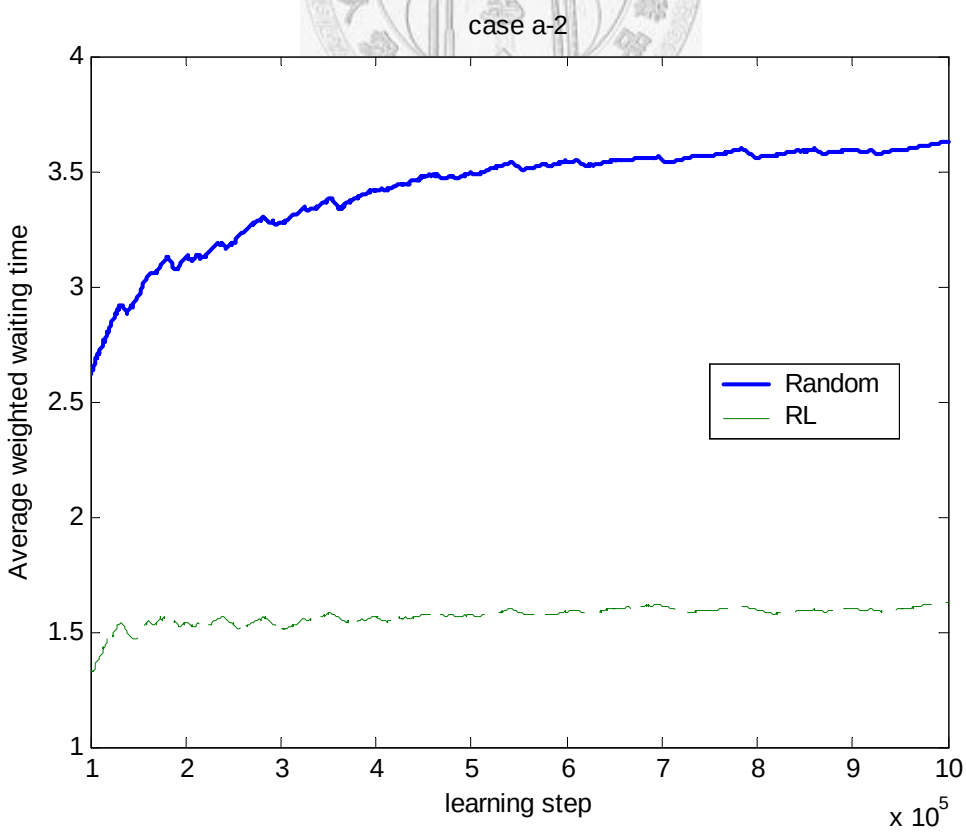


圖 3-12: Case a-2 平均權重等候時間比較圖

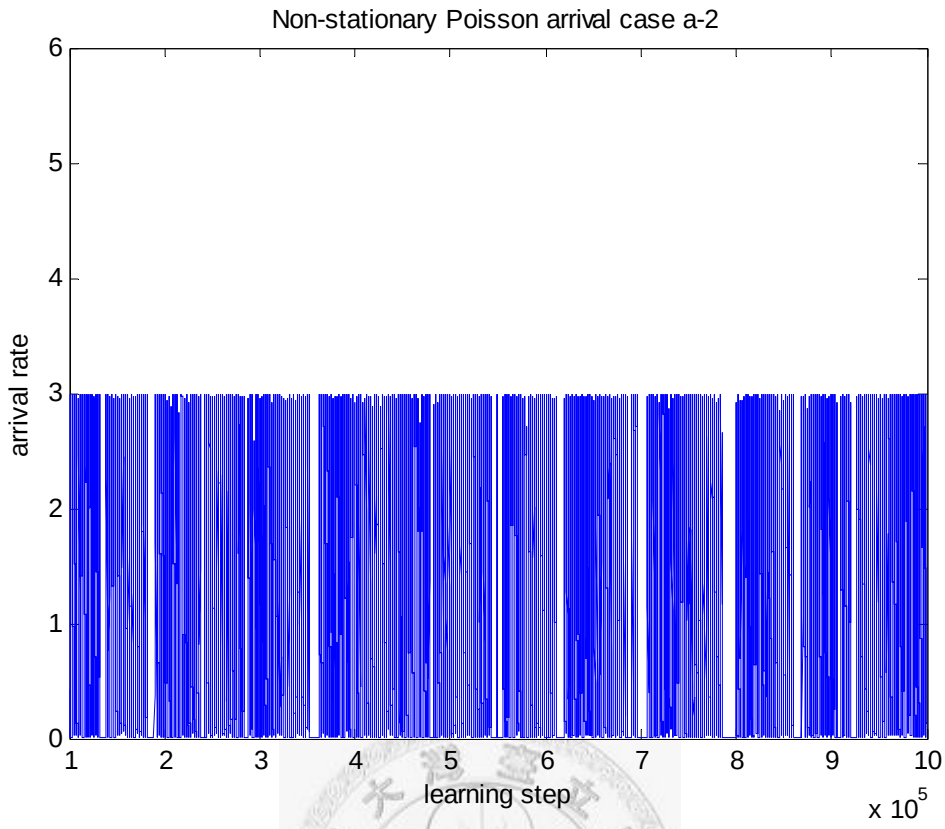


圖 3-13: Case a-2 時變到達率

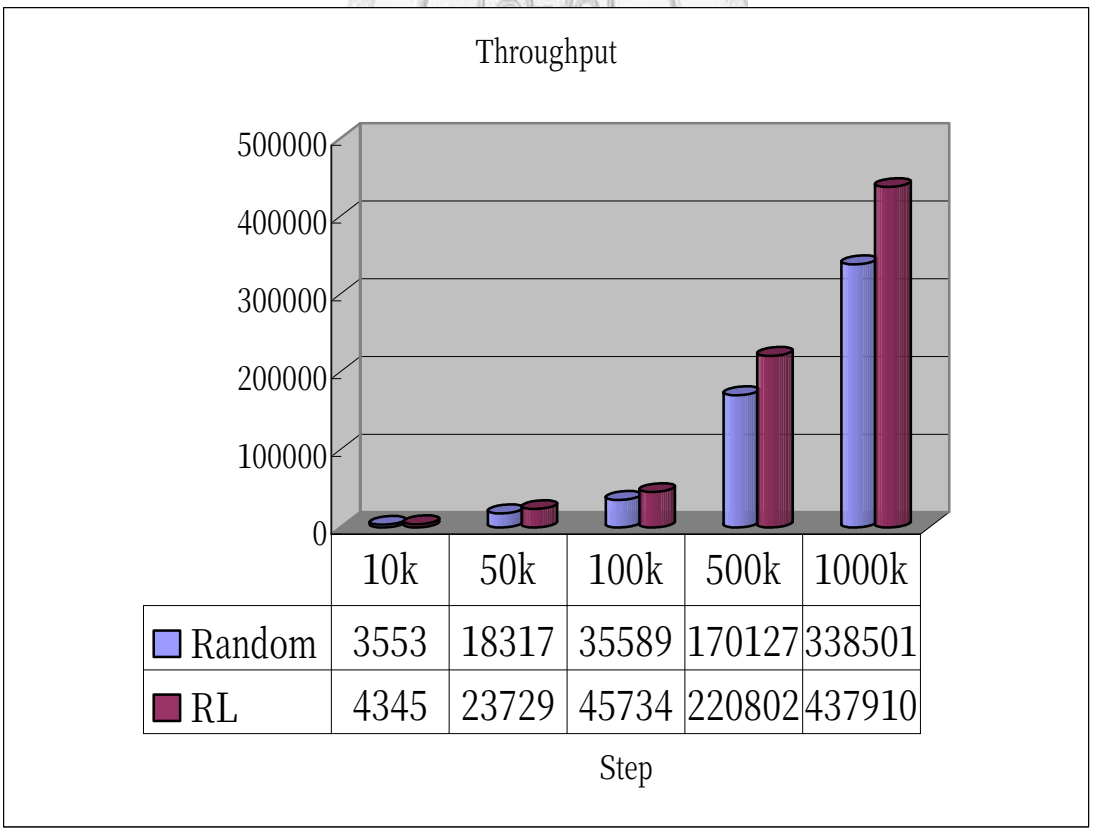


圖 3-14: Case a-2 產量比較圖

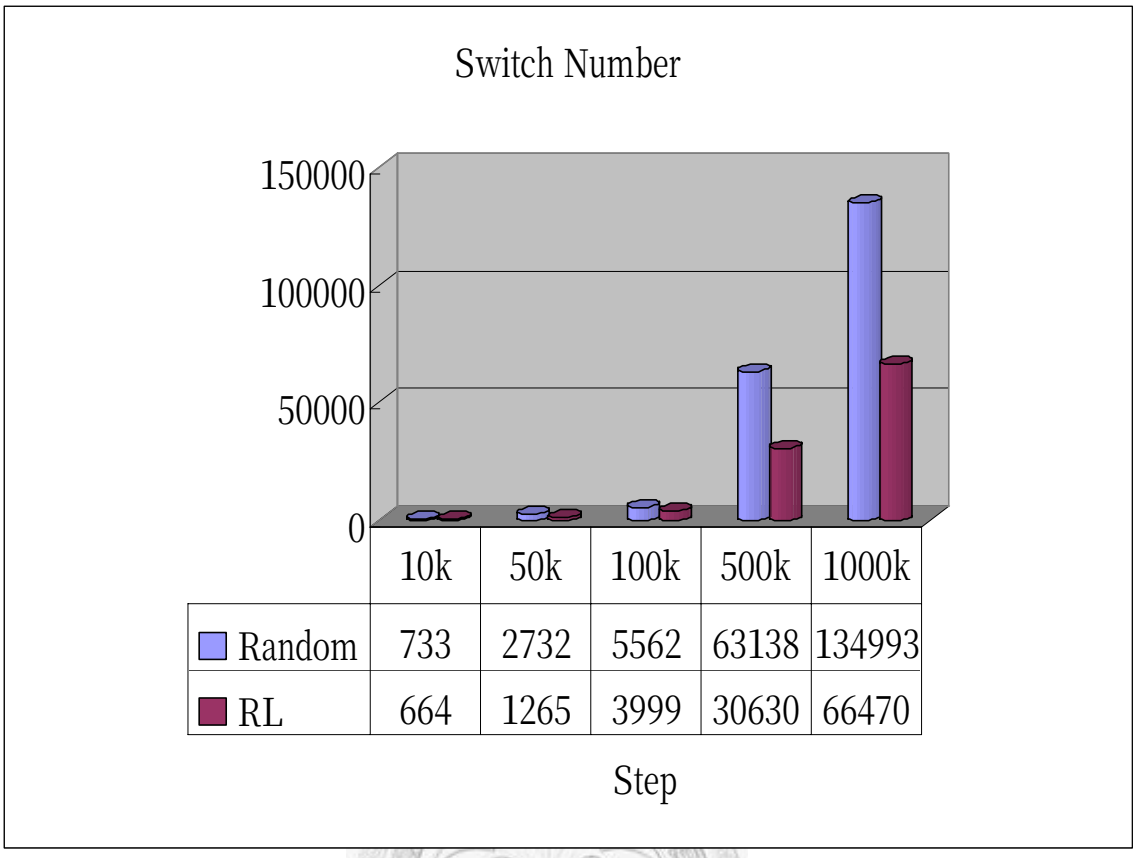


圖 3-15: Case a-2 切換次數比較圖

Case b-1:

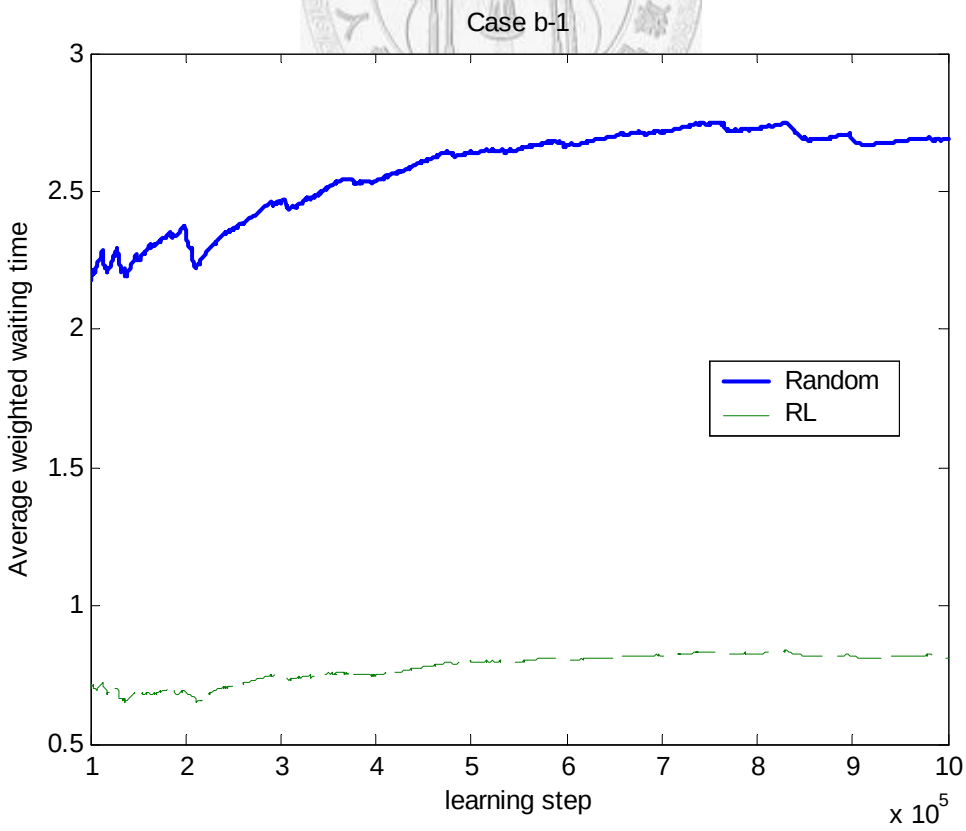


圖 3-16: Case b-1 平均權重等候時間比較圖

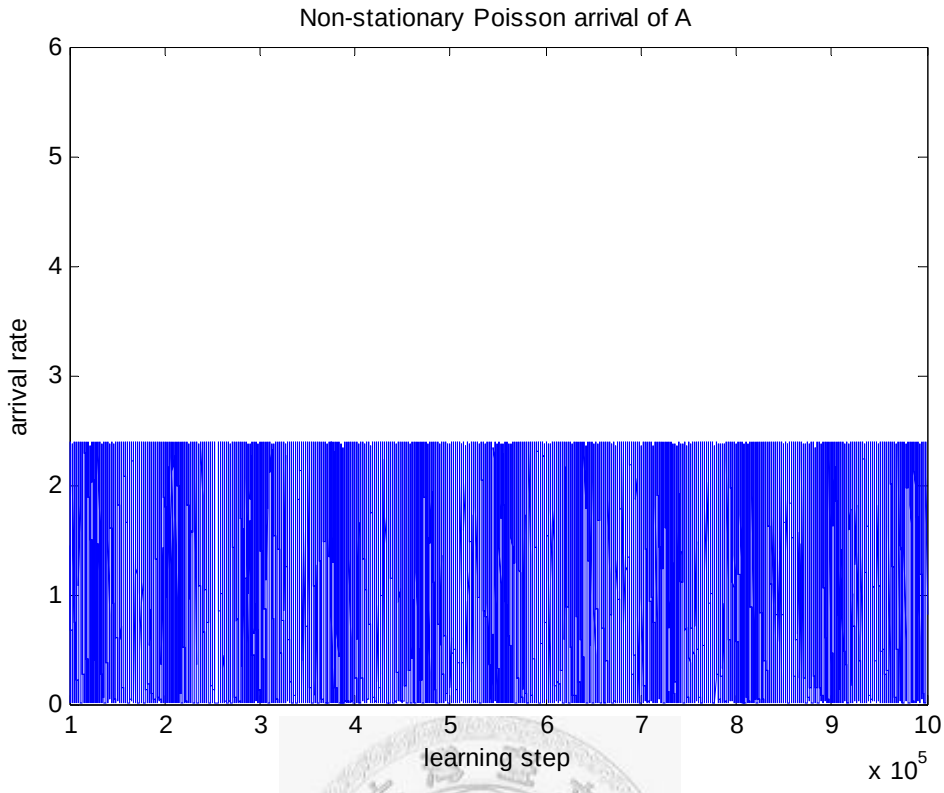


圖 3-17: Case b-1 A 產品時變到達率

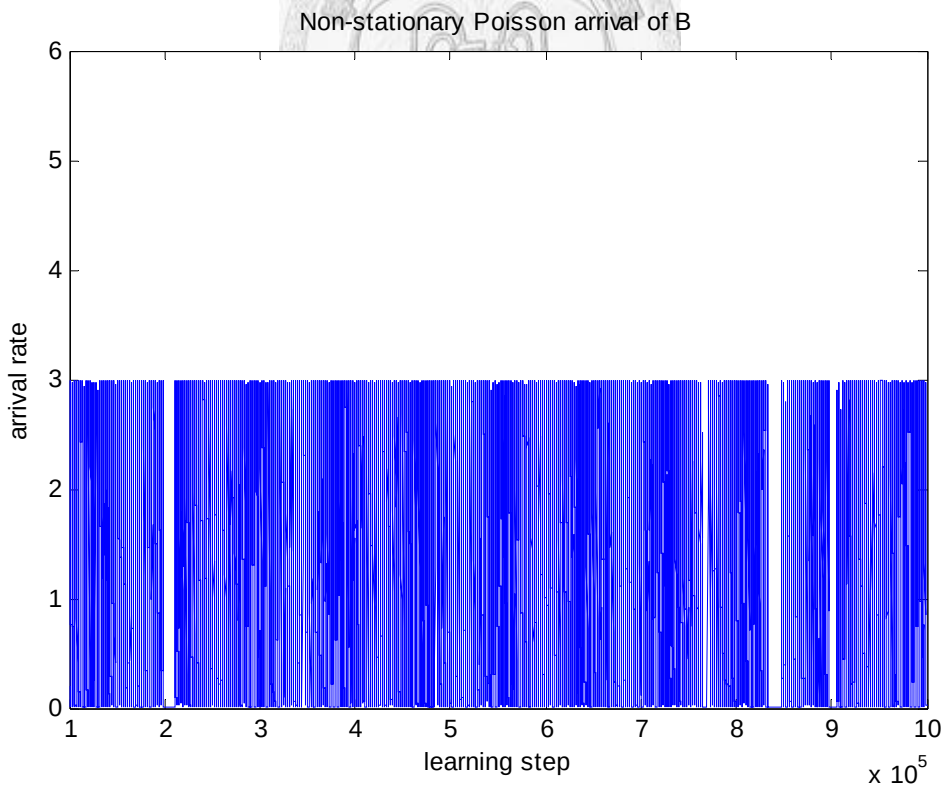


圖 3-18: Case b-1 B 產品時變到達率

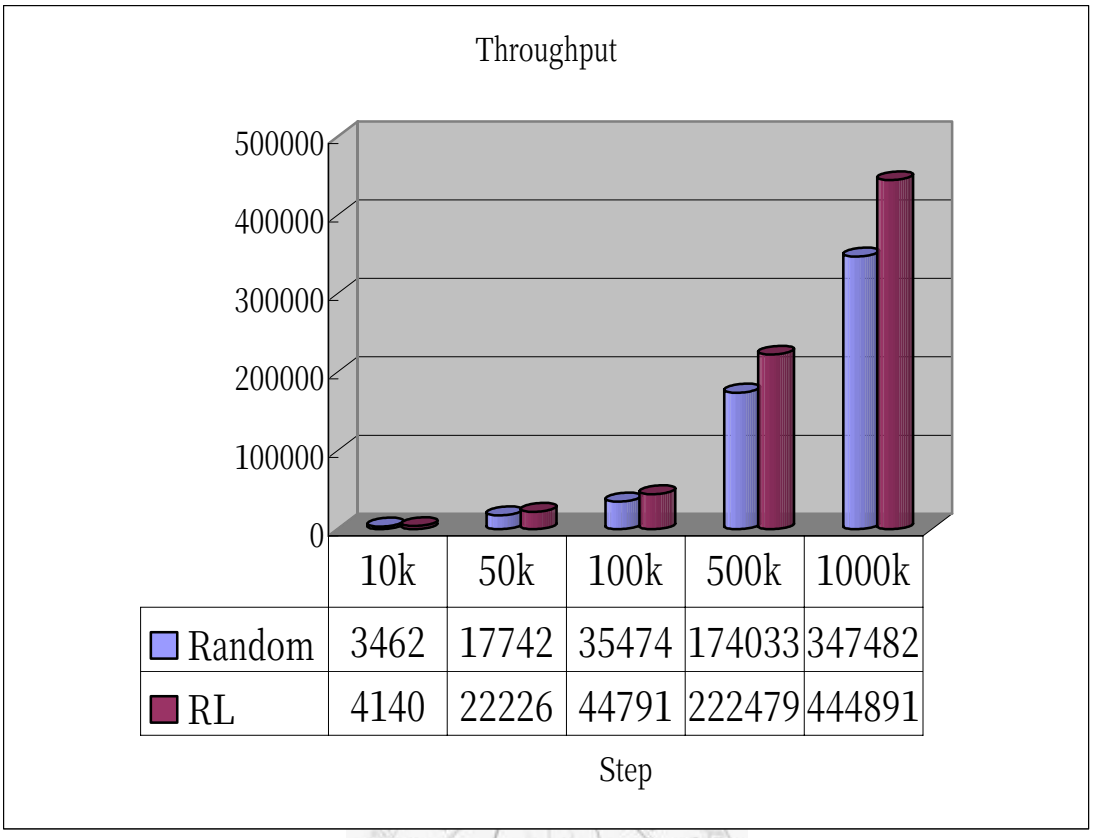


圖 3-19: Case b-1 產量比較圖

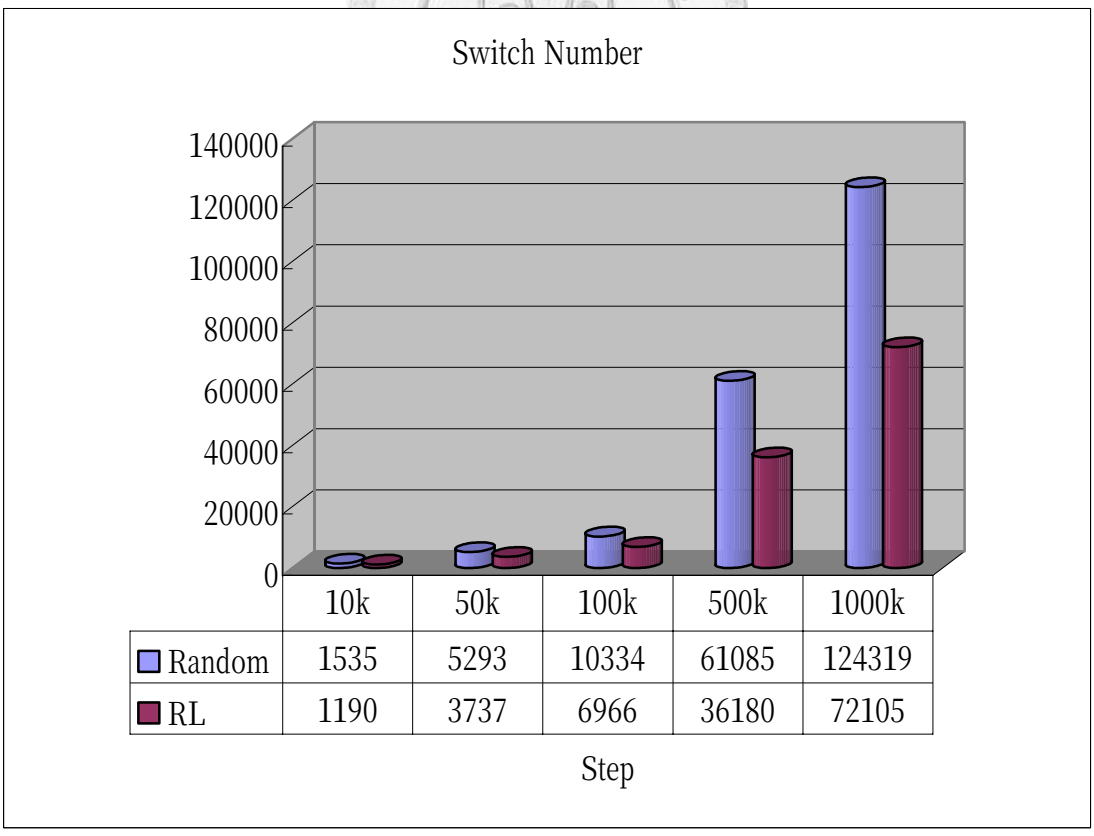


圖 3-20: Case b-1 切換次數比較圖

3.5 總結

本章研究了有設置時間的機台派工問題，定義了回饋函數，成功的表述成馬可夫決策過程，並且應用法則疊代法求得最佳解。之後應用了增強式學習學出非時變的進似最佳法則，在 95%左右的正確率下，我們認為增強式學習可以成功的應用於此類派工問題。並也證實了連續時間 Sarsa 演算法的正確性，以及探討了已有一個啟發式法則之後再學習的成效。接著我們把增強式學習應用在時變的環境中，得到了等待時間，產量，與切換製程次數的統計值。與隨機法則比較下，增強式學習都有其過人之處。在理論上，我們研究了增強式學習的演算法，並加以改良。在實際上，應用在派工的問題有不錯的表現。

在單一機台兩種產品的問題中，表示成馬可夫決策過程，就有一定的複雜度。如要擴展到多機台，多產品，則可以想見狀態的龐大性。此時要考慮的是學習的速度，以及狀態的表示法。因此對於大型問題，此演算法能否應用還是個未知數，且在時變環境中，能否與具有代表性的啟發式派工法則比較，仍是一個需要探討的議題。

第四章

機台生產速率控制

本章我們研究單一機台產能配置的方法，以達到給定的區域性生產要求。我們所假設的系統是一個有當機可能以及可以調整平均生產速率的機台。我們希望的目標是平衡等候工作數與高低速的製造成本。即是找到一個最佳切換速率的時機。我們針對加入生產速率的控制策略的系統，針對此問題的特性，加入了回饋函數，使得問題成為一個馬可夫決策過程。並且探討使用增強式學習應用的可行性。這裡我們探討了各個參數與最佳轉換點的關係。包括工作到達率，高速加工所需成本。而我們發現給予最佳法則下，增強式學習對於學習環境改變後的最佳法則，可以較從無開始學習到最佳法則快一千倍以上。

本章 4.1 節描述了機台生產速率控制問題，4.2 節以法則疊代法解出最佳切換點，4.3 節應用增強式學習求得最佳切換點。

4.1 生產速率控制問題之描述

1. **區域性的生產要求與控制生產速率之問題:**

我們所假設的系統是一個有當機可能以及可以調整平均生產速率的機台。生產速率的控制策略代表產能的配置。當機台是在正常運作下且機台前的等待工作總數超過生產速率切換值 K 時，機台以高的速率運作，反之，則採用低速率。在[Che00]文獻中，針對加入生產速率的控制策略的系統，以馬可夫分析方法加以分析，得到生產績效，如等待工作總數(Queue Length) 和系統內總工作數(System Size)，與生產速率之控制策略中控制參數切換值的關係。透過以上分析，[Che00]得到許多生產績效指標的統計量，除了平均值(Mean)以外，也包含變異量(Variance)。由數值結果可以顯示出生產速率之控制策略對於等待工

作總數和系統內總工作數的平均值及變異量的控制有很好的效果。透過以上的分析和推導，我們不只能夠瞭解生產速率之控制策略的特性，並且得到了生產速率之控制策略和給定生產要求之間清楚的數學關係，並且進一步瞭解藉此控制策略來達成給定的生產要求的可行性。

由[Che00]的實驗顯示出，越大的 k 值，會造成越大的平均等待工作總數。意即，在等待工作總數越小時，把速度調高會有越好的效果。這是一個很直觀的結果。那是不是不管如何都把速度調高就會是最佳的策略了嗎?很不幸的，天下沒有白吃的午餐，將速度調高，意味著要分配較多的機台來生產。不過機台的數目有限，分配較多的機台加入此製程，代表著其他製程的產能要減少，雖然此製程的等待工作數減少，但是相對於整個廠來說，要付出較高的成本。

於是我們考慮一個會當機的機器，發生當機的時間為指數機率分布，修復的時間也為指數分布，工作的到達是一波松分布。機器生產速率是可以調整的，分為 μ_1, μ_2，而 $\mu_1 < \mu_2$，兩者皆為指數機率分佈。而生產速率的大小代表著產能的分配，也就是意味著分配較多的機器投入生產此一製程的工作。然而，較多的機器會有較高的成本，堆貨增加所要付出的成本也較高，於是我們必須要在這兩者之間取平衡。也就是我們必須要達到使堆貨情形少的情況，換句話說，設計一最佳轉換點 k 值使我們總成本最低。如圖 4-1 所示，為一個可以調整處理速率的系統，而我們的決策為，當等候區的產品達到某一程度的數目後，則將速度調高或調低，以達成我們所要的需求。

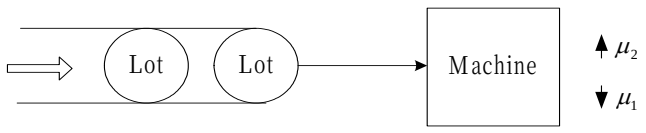


圖 4-1:可調速率之生產系統

在建立數學模型前，我們先對於此問題特性作以下的假設，在機器壞掉的時候，我們做決定的準則如下:

a.壞掉以後，馬上進行修復。

b.修復完成後，要重新計算加工時間，且要重新決定要調高或低速率。

接著，我們考慮此問題的數學模型:

2. **連續時間的馬可夫決策過程:**

由於工作到達的時間分布，處理的時間分布，機器當機與修復的時間分布皆為指數分布，所以基本上所有狀態的轉移時間皆為指數分布，是一個馬可夫過程。所有機率分佈的參數定義如下:

λ ： 產品的平均到達率，時間為指數機率分布

μ_1 ：機器的平均處理速率，為較低速

μ_2 ：機器的平均處理速率，為較高速

f ： 機器的平均當機率，時間為指數機率分布

r ： 機器的平均修復率，時間為指數機率分布

接著我們同樣把此問題表述成馬可夫過程，而馬可夫過程的主要要素分述如下:

a. 狀態(State)

我們狀態的定義延續陳依君論文的定義，如下表示:

$S(t)=\{N,M\}$ (4.1)

而狀態變數的詳細定義如下:

N:目前系統裡面的產品數目為整數。且系統有容量上限 K_T，即 $0 \le N \le K_T$

M:目前機器的狀態，0 代表當機中，1 代表正常。

b. 可允許的決策(Admissible Actions)

我們可做的決策有三種，即在機器正常下切換到兩種速率下工作，與機器當機後的修復。定義為 a(s)=u，u∈{0,1,2}，1 代表切換至較低速的速率處理，2 則是使用高速率的處理速度。每個決策是依照狀態決定的，因此定義一個可允許的決定(admissible action)集合 $\Phi(s)$，而每個決策 $a(s) \in \Phi(s)$。如下所示:

$$\begin{cases} s=(N,0)，\ \Phi(s)=\{0\} \\ N \neq 0，\ s=(N,1)，\ \Phi(s)=\{1,2\} \end{cases} \qquad (4.2)$$

c. 狀態轉移機率(State-Transition probability)

因為只有在機器正常下才能做決策，機器當機的時候只有一個動作，就是修復它。而轉移機率的算法如第三章所提到算法，這裡只表示出轉移率的部份所以我們的狀態轉移圖如圖 4-2 所示，:當機器是正常狀態下，可以選擇調高或調低速度。假如選擇調低速度，即 a=1，則下一次事件的發生有三種，一種是新的產品到達，此時等候區總數加一，一種是做完產品，此時等候區總數減一，或者最後一種，機器當機。同理，也應用在選擇高速處理 a=2 的狀況。如果機器是在當機狀態下，此時只有修復。此時有兩種事件可能發生，一是修復完成，回到機器正常的狀態，二是有新產品到達，等候區數目加一。最後一種特殊的情況，是沒有東西可以做的時候，此時也不用作決定，只有等待下一個產品到達。

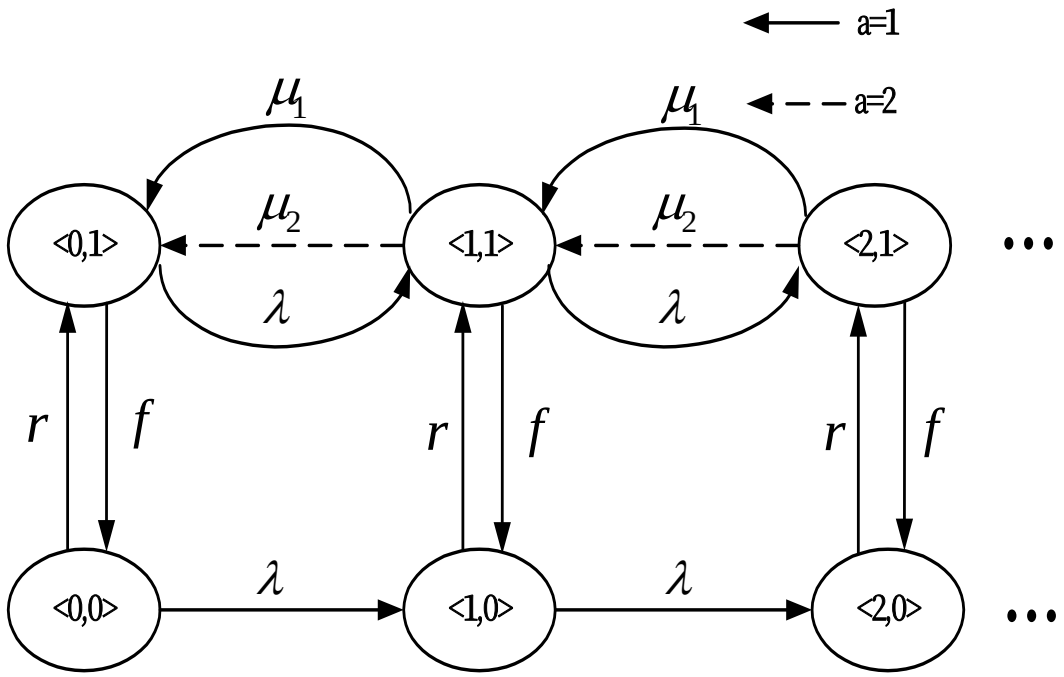


圖 4-2:狀態轉移圖

d. 目標函數(Objective function):

在這邊的 Reward 有兩種，一種是等候區的等待數目乘上時間，與我們生產時所要花費的成本。假設到達下一個狀態的時間為τ，則我們的 reward 可寫成:

$$c(s,a_{\mu i}) = [(C_w \times N)^2 + C_{\mu i}] \qquad (4.2)$$

$$r(s,a) = E_s^a\{\int_0^\tau c(s,a)e^{-\alpha t}dt\} \qquad (4.3)$$

這裡的平方項，是指等待數目越多其重要性會越高。其中C_w是單位等待數目的權重，$C_{\mu i}$是單位時間所需花費的生產成本。

而我們的目標函數與 3.10 相似，意即找出最佳的法則。

我們的目的是要找出最佳的轉換點，也就是要相對應每個狀態，要有最佳的策略，而這些最佳的策略可以預測的，是在等候區的數目到達某一數量 k 之前都做低速處理，超過 k 之後做高速處理。也就是找到最佳的轉換點 k*。

4.2 法則疊代數值實驗結果

經由以上的馬可夫決策過程的表述，同樣的，我們可使用法則疊代法可以找到一個非時變的最佳法則。意即每個狀態下的最佳策略的一組解，因此我們可以知道最佳的轉換點。參數的數值將沿用[Che00]的實驗數值，我們加入了 C_w，$C_{\mu 1}$，$C_{\mu 2}$ 三個新變數，分別是代表等待的單位成本，低速與高速處理的單位成本。此三種成本的制定沒有一定標準，視其重要程度而定，也可以說由決策者自行決定，以下就來觀察此三個變數對結果會有什麼影響。

我們實驗參數值如表 4-3 所示，分別改變 μ_2 的值，表示高速與低速的差異程度。接著對於每個情形分別改變到達率的大小，$C_{\mu 2}$ 的大小來作實驗，來觀察此兩個變數對於轉換點的影響。由表 4-4 可以看出，對於到達率越來越小的情況下，轉換點有變大的趨勢。意即，對於產品的需求量較小的情況下，生產資源分配可以容忍較多的堆貨再考慮調高速率，或者是說分配較多機台生產。

這邊可以進一步以馬可夫鏈分析堆貨情形的機率大小，來看看或許有些堆貨的情形是否是以極小機率發生。得知機率值後，配合最佳高低速轉換點，可以得知在最佳解的狀況下，使用高低速率的最佳比例。

以 Case a-1 來說，λ=1 的時候，k*=2。我們可以用穩態平衡方程式如 (4.4)，來求得各狀態的機率。方程式的觀念即，當穩態存在時，每個狀態流進總數與流出要相同。其中機率的表示方式為，$P_{N,M}$，代表狀態(N,M)的機率。而邊界條件為 $\sum_{M=0}^{1}\sum_{N=0}^{K_T} P_{N,M} = 1$。

$$
\begin{cases}
-(f+\lambda)P_{0,1} + rP_{0,0} + \mu_1 P_{1,1} = 0; \\
-(f+\lambda+\mu_1)P_{n,1} + rP_{n,0} + \mu_1 P_{n+1,1} + \lambda P_{n-1,1} = 0, \quad 1 \le n < k^*; \\
-(f+\lambda+\mu_1)P_{k^*,1} + rP_{k^*,0} + \mu_2 P_{k^*+1,1} + \lambda P_{k^*-1,1} = 0; \\
-(f+\lambda+\mu_2)P_{n,1} + rP_{n,0} + \mu_2 P_{n+1,1} + \lambda P_{n-1,1} = 0, \quad k^*+1 \le n \le K_T - 1; \\
-(f+\mu_2)P_{K_T,1} + rP_{K_T,0} + \lambda P_{K_T-1,1} = 0; \\
\\
-(r+\lambda)P_{0,0} + fP_{0,1} = 0; \\
-(r+\lambda)P_{n,0} + fP_{n,1} + \lambda P_{n-1,0} = 0, \qquad 1 \le n \le K_T - 1; \\
-rP_{K_T,0} + fP_{K_T,1} + \lambda P_{K_T-1,0} = 0.
\end{cases}
\tag{4.4}
$$

由以上方程式算出，λ=1，k*=2 時，等候區有一個貨的時候，機率為 0.2476。這表示出在此狀況裡，理論上將有約 75%的機會皆使用高速處理。見表 4-6 可以清楚看出每個狀況下的使用高速處理的機率 $P_{\mu 2}$。

另外表 4-5 可看出，$C_{\mu 2}$ 與最佳轉換點的關係。表中顯示 $C_{\mu 2}$ 越高，最佳轉換點越大，即高速處理的成本越高，越可以容忍等候區堆積個數的增加。換成機台分配的角度來看，此區要要求較多的機台資源需要較高的成本，就是相對於別區，此區的出貨重要性較低。這時候就不會再分配額外的機台資源於此區。

Case	K_T	λ	μ_1	μ_2	f	r	C_w	$C_{\mu 1}$	$C_{\mu 2}$
a-1	100	1	1.2	1.5	0.001	0.1	1	2	5
a-2	100	1	1.2	1.8	0.001	0.1	1	2	5
a-3	100	1	1.2	2.4	0.001	0.1	1	2	5

表 4-1:實驗參數

Case a-1										
k*	2	2	2	3	3	3	3	4	4	4
λ	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Case a-2										
k*	1	2	2	2	2	2	2	2	2	3
λ	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Case a-3										
k*	1	1	1	1	1	1	1	1	2	2
λ	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1

表 4-2:到達率改變與 k*的關係

Case a-1										
k*	2	3	5	6	6	7	8	9	11	12
$C\mu 2$	5	10	15	20	25	30	35	40	45	50
Case a-2										
k*	1	3	3	4	5	5	6	7	7	8
$C\mu 2$	5	10	15	20	25	30	35	40	45	50

Case a-3										
k*	1	2	3	3	4	4	4	5	5	5
$C\mu 2$	5	10	15	20	25	30	35	40	45	50

表 4-3:加工成本與 k*值的關係

Case a-1										
$P_{\mu 2}$	0.75	0.7	0.63	0.32	0.25	0.18	0.12	0.03	0.02	0.01
λ	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Case a-2										
$P_{\mu 2}$	1	0.66	0.6	0.54	0.47	0.4	0.33	0.25	0.17	0.02
λ	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Case a-3										
$P_{\mu 2}$	1	1	1	1	1	1	1	1	0.17	0.09
λ	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1

表 4-4:在最佳轉換點下使用高速的機率

4.3 以增強式學習的結果

接著，此節探討以增強式學習來針對此問題求解。首先，我們以 case a-1 的為例。考慮問題的實際性，我們用方程式(4.4)算出λ=1，k*=2 時，等候區大於 10 個的機率只有 0.0331。於是我們將 case a-1 的最大容量縮為 10 個。

分別對 case a-1 的λ=1，λ=0.7 這兩個狀況做學習。而探索的參數ε隨著學習次數減小。經過多次試驗與經驗得知，發現ε的值，用分段的遞減方式，會得到較好的結果。我們ε選用第一段 0.1，第二段 0.05，第三段 0.01，而最後為 0。兩個 case 的每段經過的學習次數大小如下所示:

$$\varepsilon \begin{cases} 0.1 & step < 1000000 \\ 0.05 & step < 3000000 \\ 0.01 & step < 5000000 \\ 0 & step < 10000000 \end{cases}, \lambda = 1 \qquad (4.5)$$

$$\varepsilon \begin{cases} 0.1 & step < 5000000 \\ 0.05 & step < 8000000 \\ 0.01 & step < 10000000 \\ 0 & step < 15000000 \end{cases}, \lambda = 0.7 \qquad (4.6)$$

在學習的速度上，此問題的顯得不夠有效率，而對於λ越小越明顯。在λ=1 的情況下，學習次數需要 1000 萬次，才得到最佳的轉換點。而在λ=0.7 的情形下，則需要 1500 萬次才能得到最佳的轉換點。

State		λ=1，step=10000000	
N	M	optimal value function	RL value function
0	0	222.02	189.53
1	0	261.89	240.5
2	0	308.39	286.73

3	0	360.29	361.49
4	0	415.96	407.53
5	0	473.35	501.17
6	0	529.94	520.29
7	0	582.7	538.96
8	0	628.11	594.49
9	0	662.28	656.67
10	0	681.17	701.14
0	1	67.608	65.264
1	1	74.89	76.164
2	1	88.68	90.153
3	1	109.9	117.91
4	1	139.57	157.38
5	1	178.15	186.74
6	1	225.31	224.56
7	1	279.55	289.42
8	1	337.34	342.04
9	1	391.85	402.96
10	1	430.46	454.23
RMS=87.57			

表 4-5:狀態價值函數一覽表

State		λ=0.7， Step=15000000	
N	M	optimal value function	RL value function
0	0	134.85	132.85
1	0	170.73	174.14
2	0	215.24	232.91
3	0	267.44	297.77
4	0	325.99	322.72
5	0	388.97	326.85
6	0	453.75	442.84
7	0	516.79	538.81
8	0	573.5	586.13
9	0	618.11	628.02
10	0	643.86	678.08
0	1	32.04	25.44
1	1	36.928	28.672

2	1	46.634	35.293
3	1	61.788	51.278
4	1	83.677	84.448
5	1	113.38	116.42
6	1	151.61	161.06
7	1	198.34	200.67
8	1	252.06	230.41
9	1	307.74	310.68
10	1	352.11	375.85
RMS=92.9855			

表 4-6:狀態價值函數一覽表二

在已有最佳法則下之學習:

由於從零開始學習的速度不佳，我們這裡考慮在已有最佳法則之下，改變到達率，來觀察還需學習多少次才能找到最佳轉換點。我們做了兩個實驗，皆是 Case a-1 所使用的參數值。一個是λ=0.7 所學出來的最佳法則，與價值函數下，將λ由 0.7 改成 1，看再需要多少次的學習，才能到達最佳的轉換點。另一個實驗則是相反狀況，將λ由 1 改成 0.7。

結果如表 4-9，前 10 次實驗的結果皆顯示，在已有法則下，可以較快學習出最佳轉換點，但是由於某些狀態的到訪的次數不足，會造成與最佳價值函數的誤差變大。而做了 1000 次實驗結果顯示:

1.λ=0.7→1 的情形: 平均 n=1824.815，平均 RMS=331.8896

2.λ=1→0.7 的情形: 平均 n=3007.379，平均 RMS=320.068

結果發現，到達率由低轉為高，學習的次數比較小，這又再次顯示出，到達率較小，學習次數需要較多次，才能到達最佳轉換點。

λ=0.7→1			λ=1→0.7	
n	RMS		n	RMS
1592	343.05		2384	318.04
436	316.66		3092	319.28
678	318.46		1728	330.47
740	309.96		1110	335.25
2178	270.89		1632	315.12
625	302.57		1895	324.73
798	302.43		4562	306.69
674	325.46		3933	315.53
628	337.39		2971	313.01
544	314.52		3739	326.31
777	330.74		2166	318.67

表 4-7:前 10 次實驗之結果

4.4 總結

在本章我們考慮一個有當機可能的機器，我們新加入了生產花費成本與等候權重成本。將此問題重建成為馬可夫決策過程，並且利用法則疊代法解出最佳法則，意即是最佳高低速轉換點。在這裡我們探討了各個參數與最佳轉換點的關係。包括到達率，高速加工所需成本，以及在最佳轉換點下使用高低速的比率。

此外我們使用增強式學習嘗試解決這類問題，在多次試驗下，發現探索的參數ε對於學習結果有很大的影響，我們找出了一組可以學出最佳解的ε，不過因為此類問題有些狀態到達的機率很小，例如當機器當機的狀態，造成學習速度不夠有效率。但是在已有最佳法則的狀態下，改變到達率之後，則可以花較少的學習次數達到最佳解，顯示增強式學習有隨著環境的變遷而學習的能

力。但是對於從無開始學習到最佳法則的情況，我們認為此增強式學習演算法仍不夠有效處理此類問題。

第五章 結論

本論文研究有設置時間之單一機台派工問題，及可調速率機台之速率切換問題。前者著重在產品平均等候時間與設置次數的取捨，而後者的目標是在等候與高低速的製造成本間作最佳取捨。由於現場環境隨時間變化，必須持續調整派工法則。本論文嘗試以增強式學習來處理這兩類問題。它將可以隨著環境不同而持續學習，藉由定義合理的回饋，估計價值函數來找到最適合的決策。我們假設狀態之間具有馬可夫性質，將兩類問題皆近似成連續時間的馬可夫決策過程。

我們嘗試應用 Sarsa 的增強式學習演算法來解這兩類問題，且用法則疊代法解出工作到達率為平穩的狀況下的最佳解。在平穩有設置時間的單一機台派工問題實驗中顯示出隨著學習次數增加，學出來的決策與最佳解有 95%的正確率。接著我們將增強式學習應用在具有非平穩環境的有設置時間的單一機台派工問題上，並與一個用一樣機率選取不同種類產品的隨機法則來做討論與比較。結果顯示，在非平穩的工作到達率下，增強式學習使平均等待時間穩定在某一定值附近，而隨機法則的平均等待時間則是逐漸增加。增強式學習也比隨機法則的產出增加了 30% ，設置的次數也比隨機法則少許多。研究結果證實增強式學習可以解決法則疊代無法處理的派工問題，即具有非平穩環境的馬可夫決策過程。但是我們發現增強式學習的學習速度不夠有效率，且也無法證實非平穩環境的最佳解為何。而我們從給予一個由 Kumar 與 Seidman, 1991, 所提出的清除法則開始學習，相對於沒有學習能力的清除法則，增強式學習會使平均等候時間隨學習次數減少。

另外對於具有可調速率機台之速率切換問題，我們考慮生產花費成本與等候權重成本間的取捨，試圖找出最佳切換速率的時機。且使用增強式學習嘗試學出最佳切換時機。結果發現，需要學上千萬次才可以得到最佳解，不夠有效率。另外用法則疊代來佐證最佳切換速度的時機，並探討了各個參數與最佳切換時機的關係。實驗顯示出，越高的工作到達率，及越低的高速加工成本，會使得轉換時機發生在等候工作較少時。最後，我們發現在給予最佳法則下，增

強式學習對於學習環境改變後的最佳法則，其學習速度可以較從無開始學習到最佳法則快一千倍以上。而對於實際生產線的派工學習，則有待進一步的評估。

參考文獻

[All03] http://www-all.cs.umass.edu/rlr/

[Bel57] R. E. Bellman “Dynamic Programming” , Princeton University press, 1957

[Ber95] Dimitri P. Bertsekas, “Dynamic Programming and Optimal Control”, Athena Scientific ,Belmont, Massachusetts,1995

[Che00] Chen I. C. ,” Stochastic Production Flow Control: Single Station Case” *NTUEE Master thesis* 2000

[Cha02] Chang Y. Z. ”A Learning Agent for Supervisors of Semiconductor Tool Dispatching” *NTUEE Master Thesis* 2002

[Chi94] C. Chiu , ”A Learning-Based Methodology for Dynamic Scheduling in Distributed Manufacturing Systems ”, Ph.D. Dissertation, Purdue University,1994

[CrB96] Robert H. Crites, Andrew G. Barto, ”Improving Elevator Performance Using Reinforcement Learning”, *Advances in Neural Information Processing Systems 8*.1996

[CrB97] Robert H. Crites, Andrew G. Barto, ”Elevator Group Control Using Multiple Reinforcement Learning Agent”, *Kluwer Academic Publishers,Boston.*1997

[CWH97] D. W. Collins, K. Williams, and F. C. Hoppensteadt , "Implementation of Minimum Inventory Variability Scheduling 1-Step Ahead Policy in a Large Semiconductor Manufacturing Facility, " *Proceedings of the 1997 6th International Conference on Emerging Technologies and Factory Automation,* 1997.

[DMN01] Z. Duwayri , M. Mollaghasemi, D. Nazzal, ”Scheduling setup changes at bottleneck facilities in semiconductor Manufacturing ” , *Proceeding of the 2001 Winter Simulation Conference*

[Gra98] Stephen C. Graves, ”A Single-Item Inventory Model for a Non-Stationary Demand Process”, 1998

[HaK98] Stephan ten Hagen and Ben Krose “Reinforcement learning for realistic Manufacturing process” *In Proceedings of CONALD’98,* June 1998, Pittsburgh

[HCC00] B.-W. Hsieh , C.-H. Chen ,and S.-C. Chang, “Dispatching Rule Selection for Semiconductor Wafer Fabrication by Ordinal Optimization and Simulation*” IEEE Trans. On Robotics and Automation*, July, 2000

[KuS90] P. R. Kumar, Thomas I. Seidman, “ Dynamic Instabilities and Stabilization Methods in Distributed Real-Time Scheduling of Manufacturing Systems” *IEEE Trans. On Automatic Control, Vol. 35, No. 3*, March 1990

[Lee96] Lee M. C. , “Machine Allocation Wafer Release Policy in Semiconductor Fabrication” *NTUEE Master Thesis June 1996*

[Lin02] Lin Y. T. , ”Design and Implementation of a Dispatching Knowledge Representation Model for Semiconductor Tools” *NTUEE Master Thesis 2002*

[LTC01] Loo Hay Lee, Loon Ching Tang, and Soon Chee Chan, **“**Dispatching Heuristic for Wafer Fabrication” *Proceedings of the 2001 Winter Simulation Conference*
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

[LRK94] S. H. Lu, D. Ramaswamy, and p. r. Kumar, ”Efficient Scheduling Policies to Reduce Mean and Variance of Cycle-Time in Semiconductor Manufacturing plants, ”*IEEE Transactions on Semiconductor Manufacturing,* Vol. 7, No. 3, August 1994.

[MaT98] S. Mahadevan, and G. Theochaurus.,”Optimizing Production Manufacturing using Reinforcement Learning” *Eleventh International FLAIRS Conference*, pp. 372-377, AAAIPress.1998

[MiS99] Manfred Mittler and Alexander K. Schoemig ,” Comparison of Dispatching Rules for Semicondutor Manufacturing Using Large Facility Models” *Proceedings of the 1999 Winter Simulation Conference*
P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

[MMD97] Sridhar Mahadevan ,Nicholas Marchalleck, Tapas K. Das and Abhijit Gosavi “Self-improving factory simulation using continuous-time average reward reinforcement learning” *In Proceedings of the Fourteenth International Machine Learning Conference.* Morgan Kaufmann 202-210,1997

[NaB91] G. Nadoli, John E. Biegel, ”Decision Making Agents in Manufacturing

Systems Simulation" IEEE ,1991

[NaB93] G. Nadoli, John E. Biegel, "Intelligent Manufacturing-Simulation Agents Tool(IMSAT)",*ACM Transactions,Vol. 3*,No. 1,Jan 1993,P42-65

[Nbu00] http://www.nbu.bg/cogs/events/2000/Readings/Petrov/rltutorial.pdf

[Qui86] J. Quinlan, "Induction of decision tree" *Machine Leaning*,1:81-106,1986

[RsA98] Richard S. Sutton and Andrew G. Barto, "Reinforcement Learning: An Introduction", *Cambridge, Mass. : MIT Press*, c1998

[Rss92] Richard S. Sutton, "Reinforcement Learning: A Special Issue of Machine Learning on Reinforcement Learning", *Reprinted from Machine Learning Vol. 8, Nos.3-4*(1992)

[SoW90] D. A. Sofge and D. A. White,"Neural network based process optimization and control." *In Proceedings of the 29th conf. On Decision and Control*, page 3270-3276, Honolulu, Hawaii, 1990. IEEE

[TFL03] C. H. Tsai, Y. M. Feng & R. K. Li ," A Hybrid Dispatching Rules in Wafer Fabrication Factories " *Department of Industrial Engineering and Management Ta-Hwa Institute of Technology* ,2003

[Tho95] M. Thompson, ""Using simulation-based finite capacity planning and scheduling software to improve cycle time in front end operations," *Proceedings of the 1995 IEEE/SEMI Advanced Semi. Manuf. Conference and workshop,1995*

[TMC00] T. Jaakkola, M. L. Littman, and C. Szepesvari "Convergence Results for Single-Step On-Policy Reinforcement Learning Algorithm " *In Machine Learning Journal, vol 38(3), pages 287-308, 2000.*

[TrB03] Sheri Coble Trovinger, Roger E. Bohn," Setup Time Reduction for Electronics Assembly: Combining Simple (SMED) and Sophisticated Methods" , The Information Storage Industry Center University of California,2003

[Wei88] L. M. Wein, "Scheduling Semiconductor Wafer Fabrication" *IEEE Transactions on Semiconductor Manufacturing*, Vol.1 , pp. 115-130 , August 1988

[YcI00] Yon-Chun Chou and I.-Hsuan Hong, "A Methodology for Product Mix Planning in Semiconductor Foundry Manufacturing" *IEEE Transactions on Semiconductor Manufacturing*, VOL. 13, NO. 3, AUGUST 2000

[Yih90] Y. Yih , ”Trace-driven knowledge acquisition(TDKA)for rule based real-time scheduling systems” *Journal of Intelligent Manufacturing*, 1(4):217-230,1990

附錄 A

法則疊代
Policy Iteration

1. *Initialization*

   $V(s) \in \Re$ *and* $\pi(s) \in A(s)$ *arbitrarily for all* $s \in S$

2. *Policy Evaluation*

   Repeat

   $\Delta \leftarrow 0$

   For each $s \in S$ *:*

   $v \leftarrow V(s)$

   $V(s) \leftarrow \sum_{s'} P_{ss'}^{\pi(s)}[R_{ss'}^{\pi(s)} + \gamma V(s')]$

   $\Delta \leftarrow max(\Delta, |v - V(s)|)$

   until $\Delta < \theta$ *(a small positive number)*

3. *Policy Improvement*

   policy - stable $\leftarrow$ *true*

   For each $s \in S$ *:*

   $b \leftarrow \pi(s)$

   $\pi(s) \leftarrow arg\ min_a \sum_{s'} P_{ss'}^{\pi(s)}[R_{ss'}^{\pi(s)} + \gamma V(s')]$

   If $b \neq \pi(s)$, *then policy - stable* $\leftarrow$ *false*

   If policy - stable ,then stop ; else go to 2

附錄 B

Uniformization:

考慮一個連續時間的馬可夫鏈，$X=\{X_t : t\geq 0\}$，而機率結構具有以下特性:

$$P\{X_{t+u}=j \mid X_u=s\}=P\{X_t=j \mid X_0=s\}\equiv P^t(j\mid s) \quad for\ all\ u\geq 0$$

其中 j，s 皆屬於可數集合 S。

我們假設留在狀態 s 的時間機率分布為 exponential，而參數為 $\beta(s)$，且 $0\leq \beta(s)<\infty$。

定義從 s 跳到 j 的機率為 $q(j\mid s)$，而 Q 是由 q 所組成的機率矩陣。

定義一個生成元(infinitesimal generator)A，為一個 $|S|\times|S|$ 矩陣，其中元素如下:

$$A(j\mid s)=\begin{cases}-[1-q(s\mid s)]\beta(s) & j=s\\ q(j\mid s)\beta(s) & j\neq s\end{cases} \qquad (B.1)$$

而此生成元與狀態轉移機率滿足科莫葛若夫方程式(Kolmogorov equation):

$$\frac{d}{dt}P^t(j\mid s)=\sum_{k\in S}A(j\mid s)P^t(k\mid s) \qquad (B.2)$$

由以上可知，具有相同生成元的隨機過程，也具有相同的機率結構。

因此我們可以藉由轉換生成元，而將不易分析的隨機過程，轉換成比較容易分析，如連續時間的馬可夫鏈，我們定義另外一個隨機過程 $\{\tilde{X} : t\geq 0\}$ 如下詳述:

選一常數 $c<\infty$ 滿足 $\sup_{s\in S}[1-q(s\mid s)]\beta(s)\leq c<\infty$，

而停留時間的指數機率分布參數為 $\tilde{\beta}(j)=c$，轉移機率為:

$$\tilde{q}(j|s)=\begin{cases}1-\dfrac{[1-q(s|s)]\beta(s)}{c} & j=s\\ \dfrac{q(j|s)\beta(s)}{c} & j\neq s\end{cases} \qquad \text{(B.3)}$$

然而新的生成元滿足:

$$\tilde{A}(j|s)=\tilde{q}(j|s)c \ \ for \ \ j\neq s \ \ and \ \ \tilde{A}(s|s)=-[1-\tilde{q}(s|s)]c \ \ for \ j=s \qquad \text{(B.4)}$$

所以可得 $\tilde{A}=A$，表示 $\tilde{X}$ 與 X 有相同的機率結構。而 $\tilde{X}$ 稱為 X 的 Uniformization

附錄 C

有設置時間的派工問題法則疊代的最佳解

$\lambda_A = 1.2$ $\lambda_B = 1.5$ $\mu_A = 2$ $\mu_B = 1.6$ $\nu_s = 3$ $\alpha = 0.1$ $\gamma = 0.9$ $C_w^a = 1$ $C_w^b = 2$

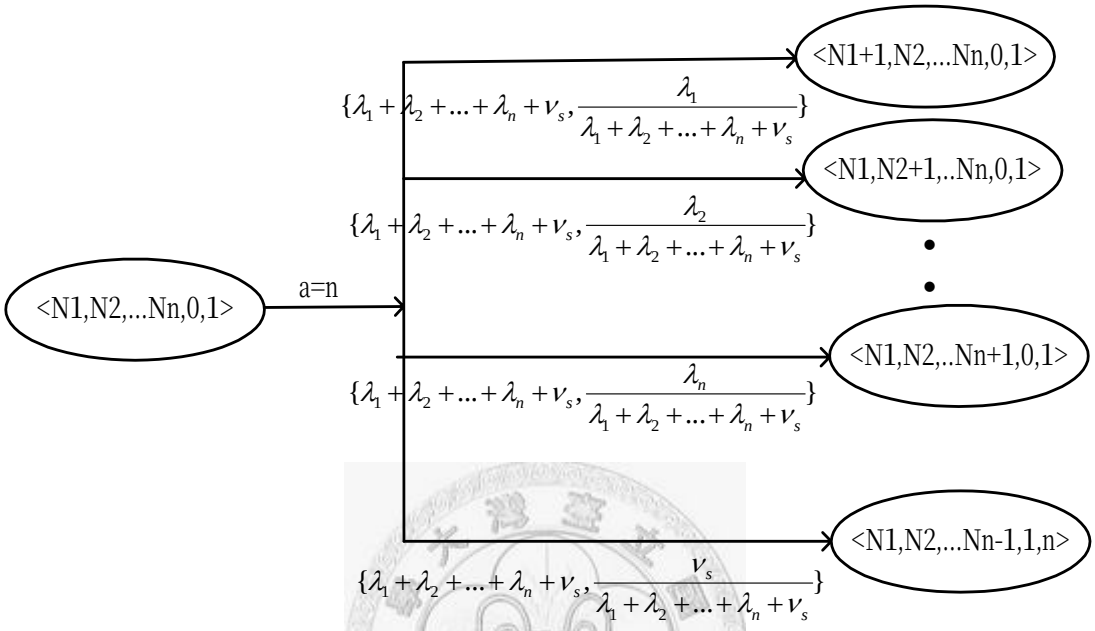
Na	Nb	M	LaT	optimal policy	optimal value function
0	0	0	1	0	44.642
0	0	0	2	0	44.274
0	0	1	1	0	45.637
0	0	1	2	0	45.607
0	1	0	1	2	46.954
0	1	0	2	2	46.061
0	1	1	1	0	48.822
0	1	1	2	0	48.651
0	2	0	1	2	50.098
0	2	0	2	2	49.105
0	2	1	1	0	51.818
0	2	1	2	0	51.838
0	3	0	1	2	52.455
0	3	0	2	2	52.291
0	3	1	1	0	53.6
0	3	1	2	0	54.052
1	0	0	1	1	45.844
1	0	0	2	1	46.099
1	0	1	1	0	47.495
1	0	1	2	0	47.76
1	1	0	1	1	49.03
1	1	0	2	2	48.214
1	1	1	1	0	51
1	1	1	2	0	50.915
1	2	0	1	1	52.026
1	2	0	2	2	51.368
1	2	1	1	0	53.875
1	2	1	2	0	54.133
1	3	0	1	1	53.808
1	3	0	2	2	54.587
1	3	1	1	0	55.422
1	3	1	2	0	56.353
2	0	0	1	1	47.703

2	0	0	2	1	48.005
2	0	1	1	0	49.384
2	0	1	2	0	49.575
2	1	0	1	2	50.971
2	1	0	2	2	50.029
2	1	1	1	0	52.866
2	1	1	2	0	52.662
2	2	0	1	2	53.986
2	2	0	2	2	53.115
2	2	1	1	0	55.793
2	2	1	2	0	55.839
2	3	0	1	1	55.63
2	3	0	2	2	56.293
2	3	1	1	0	57.359
2	3	1	2	0	58.042
3	0	0	1	1	49.591
3	0	0	2	1	49.484
3	0	1	1	0	50.693
3	0	1	2	0	50.69
3	1	0	1	2	52.029
3	1	0	2	2	51.313
3	1	1	1	0	53.878
3	1	1	2	0	53.694
3	2	0	1	2	55.191
3	2	0	2	2	54.317
3	2	1	1	0	56.961
3	2	1	2	0	56.837
3	3	0	1	1	57.566
3	3	0	2	2	57.46
3	3	1	1	0	58.831
3	3	1	2	0	59.027

附錄 D

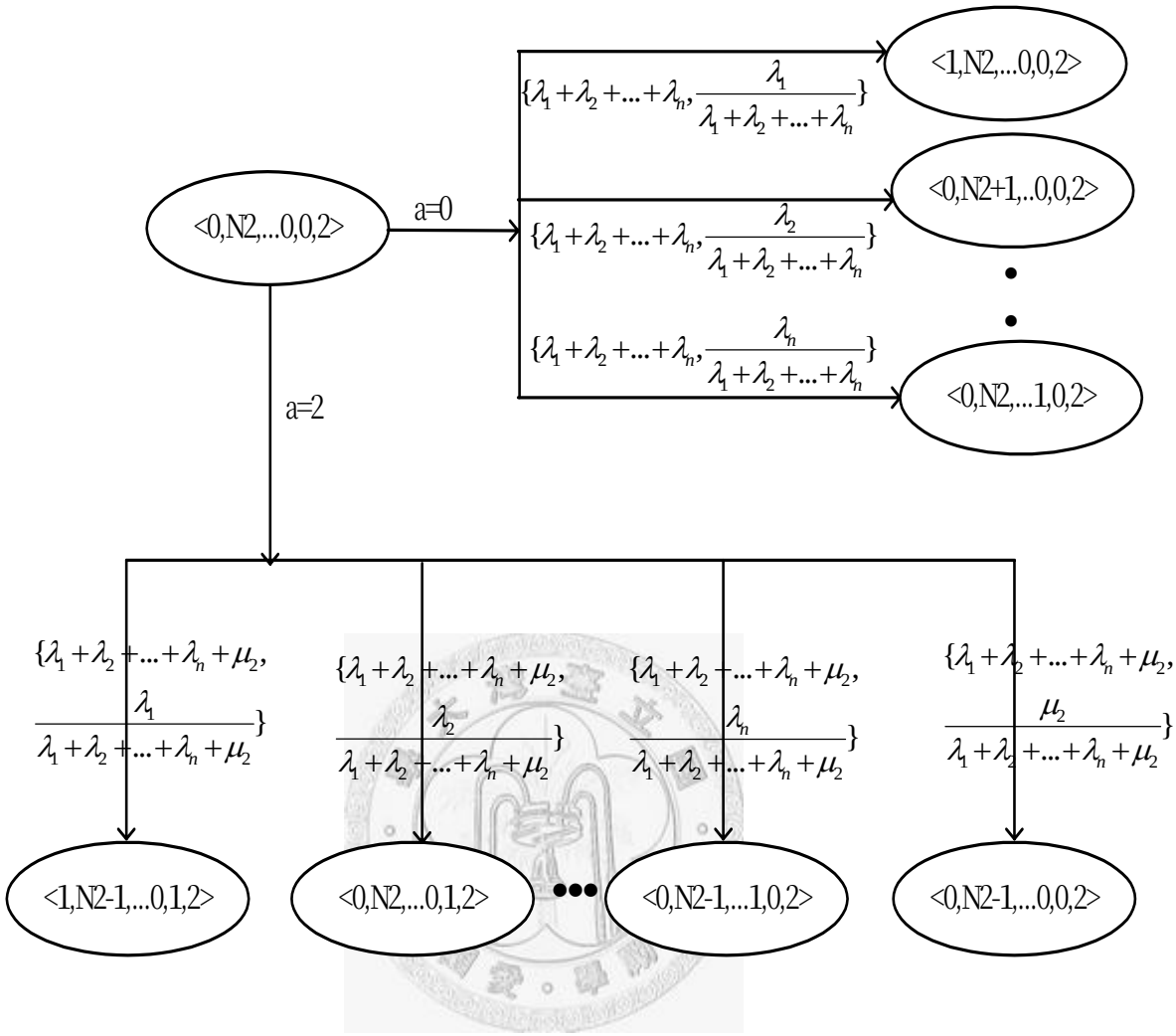
狀態轉出圖(第三章)

Case-1　a=n 的時候



Case-2　LaT=n 的時候

Case-3　只有 N2 有產品時



Case-4　都沒有產品時

附錄 E

轉換過的 Sarsa 演算法

Initialize Q(s,a) arbitrarily

Repeat(for each episode)

Initialize s

Choose a from s using policy derived from Q(ε -greedy)

Repeat n :

Take action a, observe r, s'

Choose a' from s' using policy (ε -greedy)

$$Q(s,a) \leftarrow Q(s,a) + \alpha[\frac{1-e^{-\beta\tau}}{\beta} c(x,a) + e^{-\beta\tau} Q(s',a') - Q(s,a)]$$

$s \leftarrow s' ; a \leftarrow a' ;$

until $n > Max\_num\_lot$