第二章　文獻回顧

班表設計(timetabling)、排班(crew scheduling)及輪值(crew rostering)是捷運營運規劃的一部分。班表設計是產生特定時間特定點開始以及在特定時間到達不同站之不同路線列車班次；排班產生詳細的每日任務（任務卡），讓不同場站的司機員駕駛所有由時刻表產生之列車班次。每張任務卡是一個司機員一天的工作；輪值是指派每一場站特定司機員下幾個星期一連串的任務卡[23]。排班問題及輪值問題兩者是所謂的人員排班問題，在文獻上通常會被描述成 0-1 整數規劃問題，這類問題是屬於 NP-Hard 高複雜度之組合問題，而在文獻上大多以在合理時間內找到較佳可行解之啓發式解法為主，最佳化模式通常用於規模較小的問題上，但在應用上不易處理實際面臨的各種限制條件。本研究從整數規劃及限制滿足問題的觀點來探討人員輪值問題，將對人員排班問題、人員排班問題型態、人員排班問題之解法及限制滿足問題與限制規劃法進行回顧與說明。

2.1　人員排班問題

Lau[9]認為人員排班就是在滿足管理者、勞方、政府等各單位的目標與政策下將人力資源適當的安排於所需的作業項目中，在組織營運時，將員工安排至各項工作以提供服務已經成為一種不可或缺的管理活動。

由於各組織間特性不同以及需求條件的差異，人員排班的種類也會不同。Beasely 與 Cao[1]依產業特性將人員排班問題區分為：航空公司組員排班、大眾運輸人員排班及一般人員排班等三種型態。

2.1.1　航空公司組員排班

航空公司組員排班通常分為勤務組合排班與組員派遣等兩個部分。勤務組合排班問題是在勤務航段資料以及勤務組合限制已知的情形下，以最小總成本為目標組成連續性的勤務任務組合；而組員派遣問題則是將所有的勤務組合在符合組員派遣限制下分派給所有的組員。

Lavoie 與 Minoux[10]針對空服員勤務組合產生問題使用變數產生法求解，求解過程是先以人工方式產生可行變數作為主問題集合涵蓋模式的起始解，之後利用單體法自現有變數集合中找到最佳解；子問題定式為最短路徑問題，利用標籤設置演算法找到對主問題目標值有貢獻的可行勤務組合變數，將其加入主問題中重新求解，以此程序改善目前解，直到無法改善才停止。

王國琛[15]針對後艙空勤組員排班問題進行求解，係在已知草擬班表及民航法規、工會規定、航空公司空勤派遣規則等限制下，以空服員營運成本最小為目標，應用限制規劃方法及數學規劃方法建立一套以限制規劃為基礎之變數產生法，求解大型後艙空勤組員排班問題。

唐依伶[19]探討公平性空服組員派遣問題，其將問題列式為限制滿足問題，運用限制規劃方法，經由問題變數與限制函數之設計，將所考量的各項限制模式化求解。研究中為提升效率適當加入「以最小累積飛行時數者優先指派」，並以「分階段指派」啓發式方法進行指派，求解公平性組員派遣問題。

2.1.2 大眾運輸人員排班

大眾運輸人員排班問題，較常見到的有公車、鐵路以及捷運相關人員（站務員及司機員）的排班。這類人員排班問題特性和航空公司組員排班問題特性相似，因此，一般在學術上求解方法也是類似：將大眾運輸人員員排班問題分為組員排班及組員派遣問題兩個部分。組員排班為考量法規規定及休假等限制下，利用乘務資料排出可行的工作班。將可行的工作班在符合組員派遣限制下分派給所有人員，即完成組員派遣。組員派遣問題在本研究中稱為組員輪值問題。本研究所要探討的捷運司機員輪值問題即是屬於大眾運輸人員排班問題的部分。

Chu and Chan[4] 針對香港輕軌捷運問題進行探討，以作業研究模式結合電腦輔助求解此問題，並發展一決策支援系統，將複雜的排班過程予以自動化。求解過程可以分為三個階段，第一階段，針對每一車旅次，先以最短路徑加上路徑限制式(path constraint)去產生可行之任務鏈；第二階段，先將第一階段產生之任務鏈，利用最小加權對稱配對方法(minimum weighted symmetric matching method)，在滿足工作班可行性的條件下，兩兩配對組合成工作班；最後一階段是改善式啟發式解法(Improvement Heuristics)，針對第二階段無法配對成工作班的任務鏈，或是有特殊因素考量以手動方式微調班表的情形，其使用一連串改善式啟發式解法來形成另外新的工作班。研究結果顯示提升部分排班績效，班表在不到半小時內快速產生。

蘇昭銘、張靖[25]利用數學規劃方法構建捷運系統站務人員排班模式，分為兩階段，第一階段模式主要在決定站務人員之上班日與休假日，第二階段模式主要在決定站務人員上班日之上班時段。兩階段目標式均包含絕對值之非線性轉換型態，透過變數轉換程序，可將其轉換成一整數規劃模式，故在求解上可使用現成套裝軟體（如 LINDO）進行求解工作。

盧宗成[22]探討捷運司機員排班問題，其將問題建構為集合分割問題模式，並設計一求解演算法。演算法主要使用變數產生法，其求解方式將主問題定式為集合涵蓋問題，子問題為有限制之最短路徑問題，利用單體法與資源受限制之最短路徑演算法分別求解主、次問題。

張育彰[21]的研究對象是台鐵列車駕駛員排班作業，將列車駕駛員排班問題分為(1)可行工作班集合產生階段與(2)排班與輪班整合求解階段，前者以網路產生啓發式產生可行工作班集合當作下一階段求解搜尋空間；後者將排班問題視為一集合涵蓋問題，而將輪班定義為不對稱的推銷員旅行問題，使用基因演算法來求解，在所有工作班均需執行過一遍的概念下，求取總週期最小等目標。

Caprara et al.[3][11]藉由使用由混合線性整數規劃放鬆形成的啓發式求解義大利鐵路輪值問題得到好的求解時間。他們目標問題是給定任務的數目之後，求解涵蓋所有每日需執行到的任務且極小化總週期的輪值表。

Ernst et al.[6][11]研究澳洲鐵路－National Rail 相關輪值問題。司機員必須駕駛長距離的列車，因此每一班次在 National Rail 來說就是一個班別，而司機員有可能需在外住宿。這樣情況和航空人員管理類似，但仍有些差異。National Rail 所有可能的來回旅程統計出來並使用它作為投入資料，因此許多來回旅程的硬限制可以放進來回產生程式(roundtrip generation program)。他們有輸出來回旅程及每一場站司機員數目的輪值泛系統的模式。之後，他們以場站為基礎建立輪值模式並使用混合線性整數規劃(MIP)模式。然而，為了在合理時間求解這模式，他們被迫放鬆不可欠缺的限制式以及集中在結果為連續解來得到整數解；這是有疑問的實例，但是在他們實務背景下可能是適當的。

Emden-Weinert, Kotas, and Speer[5][11]研究德國有軌電車及公車司機員的輪值系統。他們提供一套不同於地鐵的輪值系統，將司機員及場站的任務分群並分別發展輪值表。一些地鐵場站對於試用多種的輪值表也有興趣，因此指派任務到同一場站不同群司機員以及指派司機員到不同群的工作就被應用到這些地鐵場站。在他們案例中只有休假日及班別種類的型式。他們使用混合線性整數規劃(MIP) 指派班別到輪值表的空格內，而這個方法有個缺點就是求解時間會隨著問題規模成指數增加。十二名的司機員輪值表花了二十八小時仍求不出最佳解，而六名司機員的輪值表使用 CPLEX 6.5.花了六個小時求出一個解。

Sodhi and Norris[11]的研究對象是倫敦地鐵司機員輪值問題，將求解步驟分成二階段混合線性整數規劃、圖論(graph-theoretic)及手動操作求解。第一階段是輪班模式，先建立網路，再利用混合整數規劃求解，最後用圖論啓發式得到各司機員所值的班別及休假日。第二階段指派問題是利用混合整數規劃求解出司機員值班時應執行的任務。其求解速度相當地快，和最佳解差距在 2%之內。這篇研究的貢獻為提供一種簡單、彈性且可能應用在非地鐵系統的方法、理論進步的可能性及增加人員輪值的作業研究文獻。

捷運專有名詞說明[16][22]：

1.車旅次(vehicle block)：指單一列車所執行之最小工作量，通常為車輛從某一機廠出發至另一場站的機廠，或是回到同一機廠的旅次。

2.任務(task)：司機員在同一車旅次內所執行的最小工作量，其端點為換班點(relief point)或是機廠。

3.任務錬(piece)：司機員在同一車旅次內連續服務的一組任務。

4.工作班(shift 或 workday)或任務卡：單一司機員每日所服務的一組任務錬。

上述各名詞間關係示意圖如圖 2.1 所示：

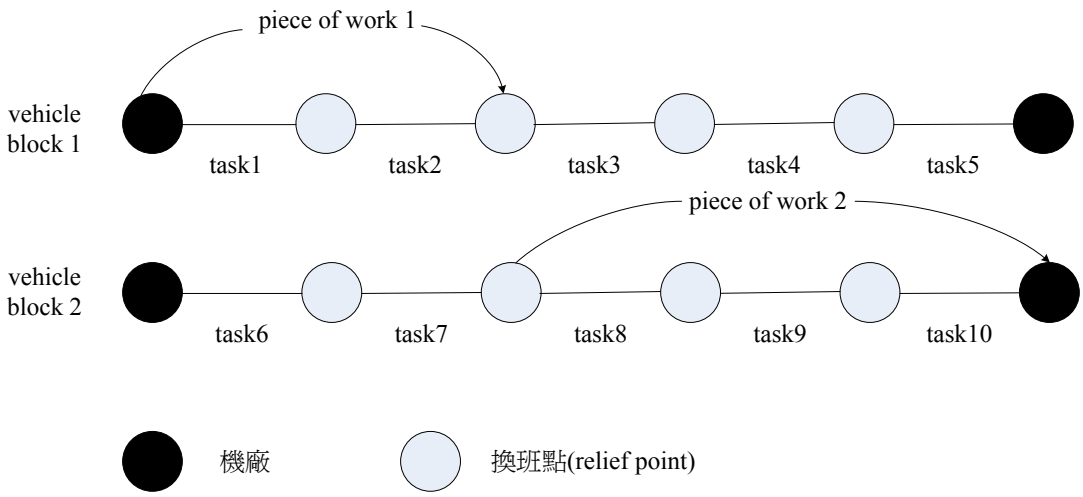

圖 2.1　捷運司機員排班問題名詞定義示意圖
資料來源：[16][22]

2.1.3　一般人員排班

除了上述兩類排班問題之外的排班問題都可歸為一般人員排班，如 104 接線生、護理人員、警察人員、客服人員、作業人員等，此類問題與之前提到的航空人員排班或是大眾運輸人員排班問題的龐大規模比較，是屬於中、小型之人員排班問題，因此，文獻上大多使用啓發式解法或其他適合的演算法進行求解。

王勇華[14]利用數學方法構建人員排班模式，採用啓發式解法在合理時間內求得近似解，組合文獻已有之方法，首先以 Morris & Showalter 之 MS 演算法求得一起始可行解，再以 Easton & Rossin 之 EAS 演算法產生相同成本解，最後加上 Henderson & Berry 之 LPI 演算法以及自行發展之 EXCHANGE 演算法進行解的改善，成為一個組合的啟發式解法 MEX；並以台灣電信北區管理局 104 東區查號台的接線生排班為應用實例。

蘇啟超[26]針對民航局飛安查核工作排程與檢查員排班問題進行探討，其將飛安查核工作排程與人員指派問題定義成資源限制專案排程問題，並應用限制規劃與數學規劃分別建構工作排程與人員指派求解模式，求解查核任務排程及人員排班之限制滿足問題與組合最佳化問題，以獲得可行的查核工作排程與查核人員班表。

李俊德[17]以護理人員排班問題進行研究，探討高複雜度之全年無休排班問題，將其構建為限制滿足問題模式，利用限制規劃法求解，並以實際醫院病房作為其研究個案。在滿足多重複雜限制條件下，排出績效良好的排班表及輪班表。

由前述文獻整理可知，目前國內在大眾運輸人員排班研究方面較多關於鐵路人員排班及輪值，而捷運司機員排班方面，目前只有盧宗成[22]，其論文較著重於組員排班，對於捷運司機員輪值問題並無深入研究，加上，捷運司機員輪值問題和鐵路司機員輪值

問題差異性較大，因此，本研究針對捷運司機員輪值問題做深入的研究。

2.2 人員排班問題型態

人員排班問題可以依據不同的排班問題有不同解題型態，主要型態分為三種：休假排班問題(Off-day Scheduling Problems)、值勤排班問題(Shift Scheduling Problems)及休假值勤排班問題(Tour Scheduling Problems)[14][18][23]。

2.2.1 休假排班問題

班表的規劃為一個星期，且每一天人員的需求已知。通常人員一星期的工作天數必小於企業單位一星期營運的天數時，因此須安排人員的休假日。而當休假日一決定，即可知道人員之工作日。休假的型態有很多種，例如一星期中有一天休假或兩天休假，後者又可分為連續或不連續假，至於是何種型態，則視營運單位法規訂定。

2.2.2 值勤排班問題

問題僅規劃一天內的班表，意即決定人員在這一天內那個時段必須工作。最簡單的方式是指派非重疊性班次(Nonoverlapping shifts)，如早、中、晚三班。然而當需求隨著時間而變化很大時，非重疊班次雖然能夠滿足尖峰時段的需求，但在非尖峰時段會產生很大的人力浪費，為了改善這種情形，於是就有重疊性排班(Overlapping shift)的產生。如此一來，班次型態就更多，增加了問題的複雜度。

2.2.3 休假值勤排班問題

此問題規劃的班表長度為一計畫週期，所排出之班表不但包含那一天為休假日，並包含工作日的工作時段，因此上述兩個問題－執勤班次排班問題及休假排班問題，皆為休假值勤排班問題的子問題。如此使得班次型態變得非常龐大，問題求解更加不易。

本研究的捷運司機員輪值問題主要在決定司機員上班日、休假日、上班的班別及所應執行的駕駛任務，故屬於休假值勤排班問題。

2.3 人員排班問題之解法

關於人員排班問題，在國內外均被廣泛討論，其相關文獻很多，學術領域定義它為一組合最佳化問題。由過去學者的研究中，其求解方法基本上可以分為兩種，一為最佳化演算法(Optimal Solution Algorithm)，另一為啓發式演算法(Heuristic Algorithm)。

2.3.1 最佳化演算法

最佳化演算法是在求解條件與限制式已知條件下，對於問題的目標求取其可行解空間內最佳的一個解。目前對於人員排班問題的作法通常是將整個問題構建成集合分割(Set Partitioning)或集合涵蓋(Set Covering)模式，並配合最佳解策略加以求解。然而最佳

化演算法最主要面臨的瓶頸為演算時間較長，對於大型人員排班問題，往往需要耗費過多的求解時間，甚至求不出可行解。關於最佳解求解方法，文獻上主要有三種：分支定限法、分枝切面法與變數產生法。

（一）分枝定限法(Branch and Bound Method)

分枝定限法為傳統上最常用來求解 0-1 整數規劃問題最佳化的方法，而由 Hiller 與 Lieberman[8]的說明可知，分技定限法的基本觀念是分割以及解決。其內容如下表 2.1。

表 2.1　分枝定限法求解步驟

節點選擇 (Node Selection)	在分枝限定法過程中，可能有許多未能求得整數之分枝端點，如何從這些未選節點當中擇一繼續進行求解工作即為節點選擇，節點選擇機制有最佳目標搜尋（Best-Bound Search）、深度搜尋（Depth-First Search）、廣度搜尋（Breadth-First Search）、隨機搜尋（Random Search）等方式。
分枝 （Branch）	當處理二元變數時，最簡單的方式就是將合理解集合固定一個變數值 x，將原本問題分割為 $x_1 = 0$ 及 $x_1 = 1$ 的兩個子問題， 以分枝樹來看，即針對 x_1 變數產生了兩個分枝。
定限 （Bound）	對每一個子問題，找出其線性放鬆後之最佳解即為該子問題之鬆弛解。對於整數規劃而言，通常鬆弛解即為放鬆掉整數限制後之最佳解。
解決 （Fathoming）	當一個子問題被解決之後即可不再考慮，而分枝限定法即是透過解決所有分枝端點找到問題最佳解，其有三種方式可以被解決：1.子問題為最佳解為整數解、2.子問題為不可行解、3.子問題之最佳解較目前暫時最佳解（Incumbent）差。

資料來源：[15]

（二）分枝切面法(Branch-and-Cut Method)

分枝切面法可視為結合分枝定限法與切面法。根據 Young[13]的定義，分枝切面法是在計算分枝端點前嘗試加入新的有效不等式(Valid Equation)，用以縮小放鬆整數限制後的可行解範圍，加快分枝定限求解速度。對一個極小化問題而言，其最佳解之上限通常是以暫時可行解中最好的一點來表示，而下限解則以一個放鬆的不可行解來代表。

（三）變數產生法(Column Generation)

變數產生法其求解概念是將原來大規模問題的主問題，分解為受限制主問題與子問題兩個部分，再利用線性規劃中的對偶理論(Dual Theory)將受限制主問題與子問題緊密連接起來，其運作方法是先透過人工方式或啓發式解法求得一組可行解，然後放鬆受限制主問題並求得一組對應每個限制式的對偶變數向量，接著將此組對偶變數向量放入子問題中，使其能運用對偶理論來產生對目前受限制主問題目標值有貢獻之新變數並加入受限制主問題中，藉由受限制主問題與子問題互動逐步改善受限制主問題之線性規劃最

佳解，直到無法再改善為止，所得之解即為放鬆後受限制主問題之最佳解。相關文獻有[10]、[15]、[22]等。

2.3.2 啓發式演算法

基於最佳演算法無法在特定時間內求得可行解的原因下，一般常見的人員排班問題的處理會選擇設計一適合的啓發式演算法進行求解，以尋求一可行解，雖然不盡然為最佳解，但其以縮短求解時間而達到效率佳往往更符合實務所需。關於人員排班問題常會使用的啓發式解法有禁制搜尋法（Tabu Search, TS）、模擬退火法(Simulated Annealing, SA)、基因演算法(Genetic Algorithm, GA)[21]等。

人員排班問題除了上述提到解法之外，近年來也有學者開始利用限制滿足問題(Constraint Satisfaction Problem, CSP)處理排班的問題。限制滿足問題主要的特色是可以將排班問題中的各種限制分成硬性限制與軟性限制，進而使用限制規劃求解。所謂硬性限制是一定要遵守的限制，而軟性限制則是盡量遵守的限制，在必要的時候可以違背軟性限制[20]。目前在國內文獻中已有學者就航空公司組員排班問題（相關文獻有[15]、[19]）及一般人員排班問題（相關文獻有[17]、[18]、[26]）採用限制規劃方法求解，獲得結果也相當不錯，但在大眾運輸人員排班問題方面，目前尚未有人使用限制規劃法求解。本研究除了利用整數規劃求解司機員輪值問題外，也將採用限制規劃方法求解部分捷運司機員輪值問題，關於限制規劃方法將於後面作深入的介紹。

2.4 限制滿足問題與限制規劃

限制規劃（Constraint Programming，CP）是一種空間搜尋技術，緣起主要來自電腦科學（Computer Science）在人工智慧領域（Artificial Intelligence，AI）的發展，根據學者 Brailsford 等人[2]對限制規劃的定義「運用電腦程式發展的模式化語言，讓使用者能簡單輕易地描述限制滿足問題，並透過精緻的電腦演算法有效率地求解限制滿足問題」。

簡單而言，限制規劃就是應用電腦演算法之執行來求解限制滿足問題，這些電腦演算法最早是在邏輯語言（如：Prolog）中所設計，但是傳統語言中的邏輯程序（logical solution procedure）並不能有效率地處理限制的關係，因此，便逐漸發展出一些針對求解限制滿足問題的限制式邏輯語言（Constraint Logical Programming），像是 CHIP 語言（Constraint Handing in PROLOG）便是由 Prolog 修改而來，利用限制式與限制式之間的關係來加速求解效率。近年來隨著電腦科技為解決實務問題而發展的物件導向概念日漸盛行，限制式邏輯語言也開發出具有物件導向式的限制規劃程式庫，常見的有 ILOG 公司的 Sover（C++）、Scheduler（C++）、Jsolver（Java）以及 COSYTECH 的 CHIPv5（C++）。

2.4.1 限制滿足問題

所謂限制滿足問題是由一群變數（variables）與限制（constraints）所構成，「在給

定一組決策變數 X={x1,x2,…,xn }、各決策變數相對應之有限值域 D={D1,D2,…,Dn}、一組限制式 C={c1,c2,…,cm}條件下，尋找滿足各項限制式之一組或多組可行解」[2]，其中，n 表示決策變數之個數，m 表示限制式之個數，有限值域具有上、下限值為每一變數可能值所構成之集合。從 CSP 定義來看，包含了變數、有限值域、限制式三個部分，一般即以（C,X,D）來描述一個 CSP 問題：

1.變數：CSP 中有多個變數 x1,x2,…,xn，這些變數構成的集合為 X={x1,x2,…,xn}。

2.有限值域：CSP 中每一個變數都有其可能值 d1,d2,…,dk，每一個可能值所構成的集合為該變數之值域 Di={ d1,d2,…,dk }，各變數值域集合所構成之有限值域為 D={D1,D2,…,Dn}，值域中的值不一定是數值型態，可以是其他型態，如：字串…。

3.限制式：CSP 中的限制式是用來描述變數之間的關係。CSP 當中的限制式可能影響任意個變數，若某個限制式剛好影響所有的變數，稱該限制式具有 arity 的性質，若 CSP 中的限制都只影響兩個變數，稱之為二元限制式（binary constraint）。

在問題的本質上，限制滿足問題不同於最佳化問題，其求解目的在於求得一組或多組可行解，而所謂的可行解是指「每一個變數 xi 從其對應值域 Di 中挑選出一個值 di，能夠滿足所有限制式。每個變數的值所構成的集合{ d1,d2,…,dn }即為可行解」。Brailsford[2]等學者指出，在 CSP 中通常會尋找幾種不同的解答：

1.只要有解即可，CSP 求解可能出現多組解，但是只要其中任何一個即可滿足需求。

2.所有的解，亦即找出 CSP 中所有的可行解。

3.最佳解：許多人常常誤解 CSP 問題並非求解最佳解，但可透過加入目標限制式，用以評估多個可行解，以獲得最符合需求的解。

2.4.2 限制規劃法

傳統 OR 領域所探討的大多屬於有明確目標式的最佳化(Optimization)問題，但對於許多諸如排班(scheduling)、班表(timetabling)、派遣(rostering)的組合性(combinatorial)問題，因其牽涉許多複雜與多變的限制條件，最佳化模式的應用效率有限。從 1999 年首度舉行跨領域的 CP-AI-OR 國際研討會以來，許多的文獻均肯定限制規劃有助於班表、組員或路線的組合問題。

限制規劃主要緣起於人工智慧(Artificial Intelligence, AI)在電腦程式語言方面的發展，一般而言，「限制規劃」指的是應用電腦演算法之執行來求解限制滿足問題。主要目的乃在於簡化理論與實務問題解決的差距，讓一般對於程式語言不熟悉的使用者，亦能很快的發展其問題模式並進行求解。

限制規劃求解為一種二層式(Two-Level)的架構，其中一層為描述限制滿足問題的限制式元件，另一層為則用來求解限制滿足問題的程式求解元件。各元件的相關內容，茲詳述如下：[12]

1.限制式元件(Constraint Component)：

限制式元件又稱為限制式模式化元件(Constraint Modeling Component)，其功能乃在於提供一套親和的模式化語言，讓使用者能直覺地(intuitive)、輕易地宣告決策變數(decision variables)與限制式。然而，由於實務問題之限制式種類相當多(如 logical constraint, arithmetic constraint, cardinality constraint, all-different constraint, atleast constraint, atmost constraint 等)，所以對於任何實務問題可能使用之限制式，限制規劃中皆存在一個其對應的限制式元件以供使用者進行問題之模式化(Modeling)動作，因此，就問題的描述方便性而言，限制規劃較傳統之數學規劃更容易讓使用者所使用。

2.程式求解元件(Programming Component)：

為了使限制規劃能有效率地求解限制滿足問題，程式求解元件中必須包含一個智慧型的限制式系統邏輯推理元件；相對地，為了使限制規劃能有系統地來求解限制滿足問題，程式求解元件中必須包含一個系統化的空間搜尋元件。然而，藉由此二元件的相互充分配合，促使限制規劃的求解效率相當的不錯。

限制規劃較適合限制程度高的組合最佳化問題(如：含時間窗限制的車輛排程問題、含容量限制的車輛排程問題等)與限制滿足問題(如：排課問題、班表設計問題、勤務組合產生問題、排程問題等)，在求解此類型問題時，限制規劃之求解效率通常較傳統作業研究來的好；然而，對於限制程度低的組合最佳化問題而言，由於作業研究有其紮實之數學理論基礎，所以在求解此類型問題時，作業研究之求解效率通常較限制規劃來的佳。根據此分析，將作業研究與數學規劃之適用問題類型整理成表 2.1 所示。

表 2.2 OR 與 CP 方法所適用之問題類型

問題特性	求解方法	可行解空間特性
限制程度高的問題 (fully-constrained Problem)	CP	可行解空間為斷斷續續的(fragment)
限制程度中的問題 (median-constrained Problem)	CP + OR	介於上、下兩者之間
限制程度低的問題 (less-constrained Problem)	OR	可行解空間相當大且非常連續

資料來源：[15]

綜合歸納應用限制規劃方法來求解限制滿足問題之優越特性，有下列三項優點：[2]

1. 有效率之求解機制：由於在限制規劃的求解機制中，充分地結合了一致性檢驗技術(Consistency Checking Techniques)、限制式遺傳機制(Constraint Propagation)與智慧型的搜尋演算法(如：Forward Checking Algorithm、Look- Ahead Checking Algorithm)，使得限制規劃的求解效率相當高。

2. 模式建立之方便性：由於限制規劃語言屬於程式語言中之「宣告式語言

(Declarative Language)」，使得使用者可以很容易地運用限制規劃語言來建構其限制滿足問題模式。

3. 可處理多樣之限制式：限制規劃可以處理各種實務問題之限制式類型而其中有許多是數學規劃所無法直接使用的，如：布林限制式(Boolean Constraints)、邏輯限制式(Logical Constraints)、序列限制式(Sequence Constraints)等，所以對於不熟悉數學規劃理論的使用者，仍然可很直覺地(Intuitively)、容易地(Easily)運用限制規劃來建構其限制滿足問題或組合最佳化問題。

2.4.3 限制規劃之求解演算法

限制規劃中的求解演算法主要是由一致性檢驗技術、系統化之空間搜尋演算法所構成，其運作機制是在每個分枝節點中運用一致性檢驗技術來縮減少變數之值域(Variables' Domain Reduction)，以減少可行解之搜尋空間，以期達到運用電腦演算法系統化且有效地求得限制滿足問題可行解之目標，亦即限制規劃中的求解演算法為一種智慧型的空間搜尋演算法(Intelligent Tree Search Algorithm)。以下即分別介紹目前限制規劃中最基本的一致性檢驗技術與目前限制規劃中最常使用之智慧型空間搜尋演算法。

（一）一致性檢驗技術

限制規劃理論中最基本的一致性檢驗技術包括：節點一致性(Node Consistency)、節線一致性(Arc Consistency)、界限一致性(Bounds Consistency)、一般化之一致性(Generalized Consistency)四種。每種一致性檢驗技術的複雜度皆不同，越簡單的一致性檢驗技術意味著其所需花費之運算時間越短；反之，越複雜的一致性檢驗技術意味著其所需要花費之運算時間越長，使得每一種一致性檢驗技術皆有其優點與適用時機。

因每一種一致性檢驗技術，皆有其最適合使用之限制式類型。如當限制式為一元限制式時，適用節點一致性檢驗技術；當限制式為二元限制式時，適用節線一致性檢驗技術；當 n(n≧3)元限制式時，適用界限一致性檢驗技術』。舉例而言，某一限制式為 X2 = 1－Y2 ∧ X≠0 ∧ Y≠0，其決策變數所對應之值域為 D(X)=D(Y)={-1,0,1}，首先分析此一限制式中，最多為二元限制式，最少為一元限制式，所以最適合此一限制式之一致性檢驗技術同時包含節點與節線一致性檢驗技術，因此，若不先經過限制式之特性分析而一味地運用界限一致性檢驗技術來縮減變數值域時，由於此限制式和與其決策變數的值域為 bounds consistent，所以決策變數之值域完全不會被縮減，但若同時運用節點與節線一致性檢驗技術來縮減變數值域，由於此限制式和它的值域不為 arc consistent，所以此限制式之決策變數值域將被縮減至空集合，亦即 D(X) = D(Y) = φ。

對於實務上的限制滿足問題，由於其包括的限制式種類可能同時出現一元、二元及多元限制式，所以，設計一個整合性的一般化一致性檢驗技術來有效地處理各種限制式，實為目前限制規劃領域中的一個重要課題。

（二）空間搜尋方法

為了讓限制規劃中的空間搜尋方法能更有系統、更有效率，電腦科學家經過評估目前文獻上所提的 Generate-and-Test (GT)及 Backtracking (BT)二種系統化搜尋演算法與結合一致性檢驗技術之可行性，最後決定採行 Backtracking 演算法作為限制規劃之空間搜尋演算法。

(1) 系統化的空間搜尋演算法

系統化的空間搜尋演算法乃指「系統化地(Systematically)、完整地(Completely)搜尋問題的可行解空間」。

GT 與 BT，其間之差異性可從兩個角度來分析，從變數指派之角度來看，GT 是獨立且同時地一次指派給每個決策變數一個可能值(generating value to each variable independently and simultaneously)，而 BT 是獨立但依序地指派給每個決策變數一個可能值(generating value to each variable independently but orderly)；從限制式之角度來看，GT 的限制式檢驗時機為當所有變數皆被指派一個可能值時，才進行限制式之檢驗，而 BT 的限制式檢驗時機為每當一個變數被指派一個可能值，即進行限制式檢驗，所以就空間搜尋之搜尋效率而言，BT 演算法之執行績效較 GT 演算法較佳。

a. Generate and Test (GT) ：

GT 演算法為系統化搜尋演算法中最原始也最容易懂的演算法，其演算法中包含兩個重要之元件，一個是 Generator(解產生器)和 Tester(解檢查器)，其運作方式為「由解產生器產生所有的可能解，然後將其丟入解檢查器中檢查看看是否滿足問題的所有限制式」。其中，GT 演算法中的解產生器完全沒有依據變數間的相關性(即限制式)來產生可能解，使得 GT 的產生器搜尋較無效率，因為 GT 是盲目地產生可能解而無法事先避免未來可能發生之衝突。

b. Backtracking (BT) ：

換枝檢驗法(BT)為系統化搜尋演算法中最基本、最常使用之演算法，其演算法之運作概念為「以 Tree Search 為核心搜尋架構、以深度搜尋(Depth First Search, DFS)為搜尋策略、以初始化(initiate)決策變數(從決策變數所對應之值域中選擇一個值指派給此決策變數)作為分枝(Branching)之目的、以決策變數值域和限制式間之一致性檢驗與紀錄每個決策變數之目前值域狀態為每個分枝節點之必要工作、以遇到死巷節點(deadend node)為定限(Bounding)及換枝檢驗之準則」，其中由於每個分枝節點需負責記錄每個決策變數之目前值域狀態，所以分枝節點又可稱為一個狀態節點(State Node)，而在 BT 演算法中，所謂的決策變數值域和限制式間之一致性檢驗乃指「新初始化變數與已初始化變數間之一致性檢驗」；所謂的死巷節點乃指「在進行決策變數值域和限制式間之一致性檢驗時，若發生上述之不一致現象，即表示此時的分枝節點為一個死巷節點，而不能再進行分枝；反之，則為一個分枝節點，能再進行分枝」；所謂的換枝檢驗乃指「因遇到死巷節點而回溯至

死巷節點之前一個分枝節點，並從此狀態節點選擇另一個分枝進行搜尋」。

在 BT 演算法之運作過程中，可依據「初始化」因素，將決策變數之類型分為已初始化變數(already initiated variables)、新初始化變數(newly initiated variable)/正初始化變數(currently initiated variable)、未初始化變數(not-yet initiated variables)三類，而所謂的已初始化變數乃指「在目前狀態節點之前，已被初始化之決策變數」，由於已初始化變數的值與問題的所有限制式皆具有一致性(Consistency)，所以由已初始化變數所形成的解集合可稱為「部分解(Partial Solution)」；所謂的新初始化變數乃指「在目前狀態節點上，所選擇要進行之分枝/初始化變數」；所謂的未初始化變數乃指「尚未進行任何初始化之變數」。

BT 演算法之運作流程為「根據 DFS 準則，依序地擴充(extend)一個部分解(partial solution)，直至找到一個可行解為止，若在擴充部分解的過程中遇到死巷節點，則換枝檢驗的動作立即發生並回溯至死巷節點之前一個分枝節點，以進行另一個分枝搜尋」，其中，所謂的擴充乃指「從未初始化變數中，選擇一個變數作為新初始化變數，並從此一新初始化變數的值域中，選擇一個值(Value)作為此變數的初始化值(Initiate Value)。

就演算法之執行效率而言，BT 演算法確實較 GT 演算法有效率，因為從 BT 演算法之運作流程中，可明顯地發現，當換枝檢驗發生時，即表示 BT 演算法刪除了一個不必要的搜尋空間，而此搜尋空間之大小即為所有未初始化變數值域的卡式積(Cartesian Product)。以運用 BT 演算法於求解 4-queens 問題為例，便可更清楚地瞭解之 BT 演算法之實際運作流程，其實際運作流程即如圖 2.2 所示：

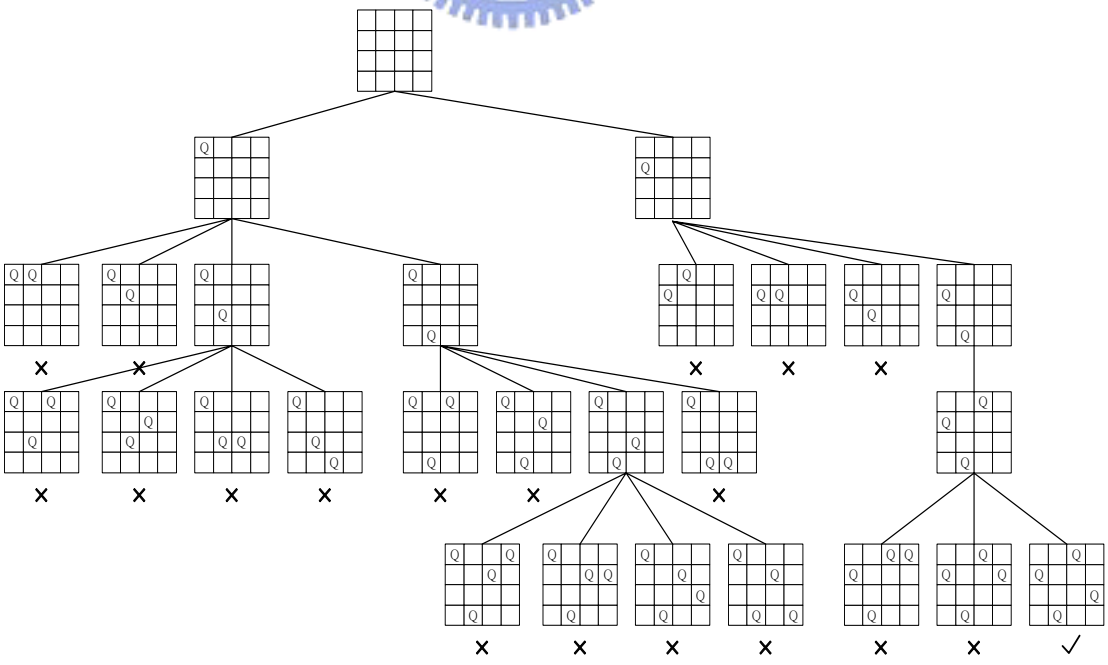

圖 2.2　BT 搜尋法-以 4-queens 為例

資料來源：[15]

綜觀上述，可明確地發現，雖然 BT 演算法較 GT 演算法有效率，但由於 BT 演算法之運作方式只能透過死巷節點的發生而減少部分的搜尋空間，仍然無法事先避免未來會發生的衝突，尤其當我們使用 BT 演算法於求解複雜度高且問題規模大的組合最佳化或限制滿足問題時，便能更明顯地凸顯 BT 演算法的無效率性。為了改善 BT 演算法之缺點，電腦科學家則透過引進一致性檢驗技術於 BT 演算法中，構成一種新的智慧型空間搜尋演算法。

(2) 系統化的智慧型空間搜尋演算法 BT 搜尋法-以 4-queens 為例

系統化的智慧型空間搜尋演算法乃指能系統化地、完整地、有效率地搜尋問題的可行解空間，最有名的有 Forward Checking (FC)與 Look-Ahead Checking(LC)兩種。理論上，BT、FC、LC 三種演算法之差異能以「演算法中，使用一致性檢驗技術於哪些變數類型之配對間」此原則來加以區分。

在系統化的智慧型空間搜尋演算法中，由於其最基本地一定有使用一致性檢驗技術於「新初始化變數與未初始化變數間」，使得在搜尋過程中之每個狀態節點中，未初始化變數之值域一定與已初始化變數之值域保持一致性(Consistency)，因此，每當從某個狀態節點進行分枝時，無須像 BT 演算法一樣要進行新初始化變數與已初始化變數間之一致性檢驗，此一優點除了可大幅降低智慧型空間搜尋演算法花費在一致性檢驗之執行時間而且還能使慧型空間搜尋演算法透過事先避免未來會發生的衝突，來減少換枝檢驗次數、增加空間搜尋之效率，進而提高整體演算法之執行效率。

a. Forward Checking (FC) ：

由 Haralick 與 Elliott 兩位學者於 1980 年所發展出來的 FC 演算法為智慧型空間搜尋演算法中最基本、最常使用之演算法，其運作概念與 BT 大致相同，差異性在於 FC 演算法是運用一致性檢驗技術於新初始化變數與未初始化變數間而不是運用在新初始化變數與已初始化變數間，藉此差異使得 FC 演算法能透過事先地刪除未初始化變數值域中與目前部分解具有不一致性的值域值(Domain Values)以避免目前部分解將面臨的未來衝突，進而大大縮減搜尋樹空間與搜尋過程中可能產生之換枝檢驗次數，大幅地提升傳統 BT 演算法之執行效率，成為一種智慧型空間搜尋演算法。同樣地，以運用 FC 演算法於求解 4-queens 問題為例，便可更清楚地瞭解之 FC 演算法之實際運作流程，其實際運作流程即如圖 2.3 所示：

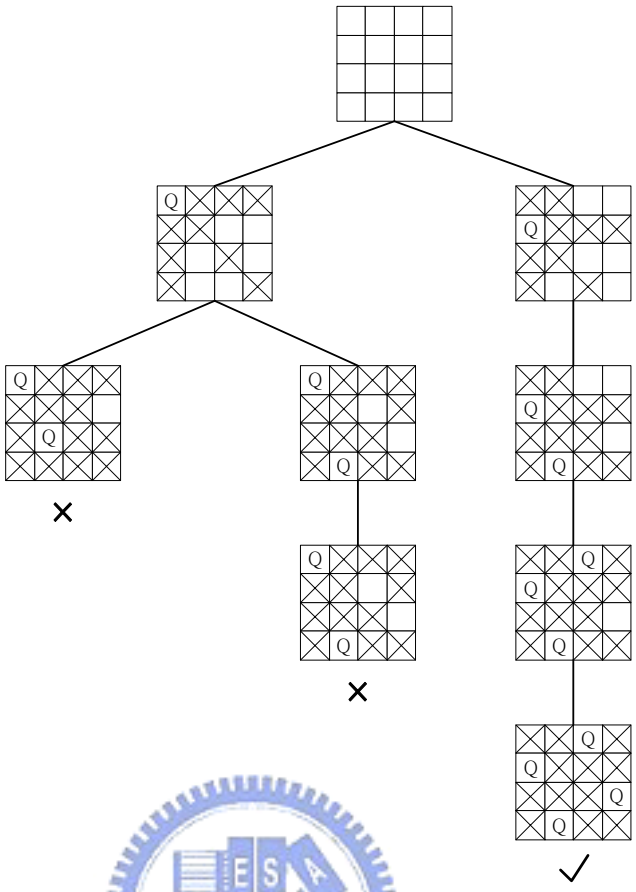

圖 2.3　FC 搜尋法-以 4-queens 為例
資料來源：[15]

詳細比較 FC 與 BT 演算法之搜尋樹空間，可明顯地發現，FC 之分枝節點數(Branch Node)較 BT 少很多，且 FC 的分枝深度(Depth of Branch)亦較 BT 短，從此可得知，FC 演算法之執行效率較 BT 演算法好很多。

b. Look-Ahead Checking (LC) ：

LC 演算法是一種比 FC 更智慧型(Intelligent)的演算法，其運作方式與 FC 皆相同，不同的是，LC 進一步地將一致性檢驗技術擴充運用至「未初始化變數與未初始化變數間」，使得 LC 較 FC 能更大幅度地縮減不必要的搜尋空間，然而此一額外之一致性檢驗使得 FC 在搜尋中的每個分枝節點上必須比 FC 花費更多的時間來執行一致性檢驗。

由 Sabin 與 Freuder 兩位學者於 1994 年所發展出來 MAC (Maintaining Arc Consistency)一致性檢驗技術為目前文獻上最常使用在未初始化變數與未初始化變數間之一致性檢驗技術，其運作方式乃「以節線一致性檢驗技術為基礎、以未初始化變數與其對應之值域所構成的子限制滿足問題(sub-CSP)為一致性檢驗之對象」，當 MAC 偵測到某個未初始化變數之所有值域與目前之 sub-CSP 呈現不一致(non-consistent)之現象時，即表示目前之 sub-CSP 不存在任何可行解，所以無須

再從目前之分枝節點進行分枝。

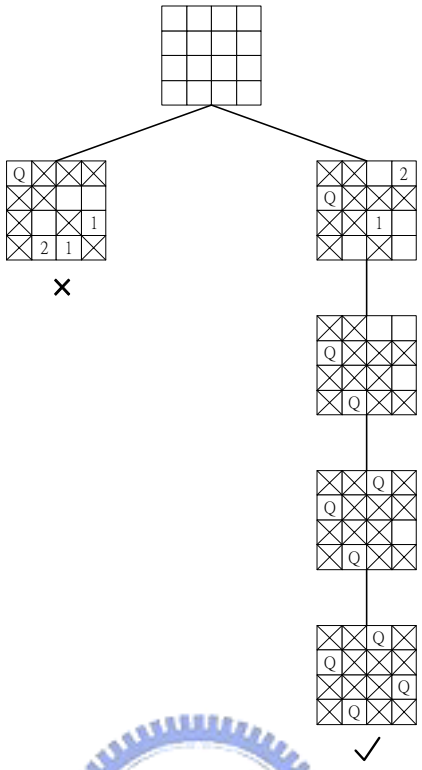

圖 2.4　MAC 搜尋法-以 4-queens 為例
資料來源：[15]

以 4-queens 為例，運用 MAC 求解此問題之運作方式如圖 2.4 所示，其中有數字的格子表示「未初始化變數中與目前 sub-CSP 呈現不一致之值域值」，而數字的標示順序乃表示「當標示數字為 i 之未初始化變數值域值因與目前之 sub-CSP 不一致而被刪除時，便會進一步促使標示數字為 i+1 之未初始化變數值域值與目前 sub-CSP 產生不一致之現象，其中 i≧1」，由此可知，標示數字為 1 的個格子表示，最開始與 sub-CSP 產生不一致之未初始化變數值域值，而標示數字為 2 的格子表示，當標示數字為 1 的值域值從其相對應之未初始化變數值域中移除之後，標示數字為 2 的未初始化變數值域值也必須因而從其對應之值域中移除，而其他的標示數字則依此類推。從圖 2.4 之左分枝來分析，由於經過 MAC 之一致性檢驗之後，發現最後一列之目前 queen 變數值域值完全與目前 sub-CSP 不一致，使得 LC 演算法能提早判定此一分枝完全不可能存在任何可行解而提止分枝和進行換枝檢驗，藉此減少空間搜尋之分枝數與提高整體空間搜尋演算法之執行效率。

2.4.4　模式化語言 OPL 之簡介

組合最佳化問題在現實世界的問題中到處存在，舉凡規劃(Planning)、排班排程(Scheduling)、排序(Sequencing)、資源規劃(Resource Allocation)等都是這類型的問題，而目前一些較強而穩健的求解系統雖然都能對大規模的線性規劃與各類型整數規劃問題提供效率不錯的求解演算法，但由於實際的實務問題規模非常大且複雜，往往使得

OR 人員在求解實務問題時，花費很多時間在模式化問題(Modeling Problem)並且轉換成這種求解系統可以讀的檔案格式與求解演算法，對於 OR 人員而言，若能擁有一套緊密連接「模式構建」與「求解演算法」的模式化語言，將使 OR 人員更快速地解決所面臨之實務問題，並大幅地提高 OR 人員的生產力與工作效率。

OPL(Optimization Programming Language)為一種專門求解組合最佳化問題的模式化語言(Modeling Language)，不同於一般命令式語言(Imperative Language)(如：C 語言、C++、VB 等)，OPL 為一種宣告式語言(Declarative Language)，由於繼承宣告式語言的特性，使得限制式在模式中的順序並不會影響模式之求解結果，所以使用者可以隨時地加入、修改、刪除與管理問題之限制式。模式化語言最大的貢獻在於，它拉近了「電腦程式語法」和「數學表示法」之距離，讓使用者能簡便地描述並解決問題。

OPL 與傳統模式化語言不同的功能可歸納成下列三項：

1. 支援限制規劃模式：支援限制規劃(Constraint Programming, CP)模式是 OPL 與傳統模式化語言最大的不同。

2. 使用者可以自訂搜尋演算法(Search Procedure)：OPL 允許使用者針對其面臨的組合最佳化問題(如：線性規劃問題、整數規劃問題、限制滿足問題)特性，自行訂定搜尋演算法；而傳統的模式化語言並不提供這個功能。

提供更多種的限制式表達方式：針對我們實務問題最常見的排班排程問題(Scheduling Problem)與資源配置問題(Resource Allocation)，OPL 提供專門的語法來表示活動(Activity)與資源(Resource)之間的關係，使得使用者欲模式化這類問題時，能更容易上手；此外，OPL 提供比傳統模式化語言更多的限制式表達方式，如：Higher-Order Constraint、Logical Constraint 等。