國 立 交 通 大 學
運 輸 科 技 與 管 理 學 系

碩 士 論 文

解決具時間窗限制的提送貨問題

Solving a Pickup and Delivery Problem with Time Window Constraints

研 究 生：黃信翔

指導教授：王晉元

中華民國　九十五　年　六　月

解決具時間窗限制的提送貨問題

Solving a Pickup and Delivery Problem with Time Window Constraints

研 究 生：黃 信 翔　　Student：Hsin-Hsiang Huang

指導教授：王 晉 元　　Advisor：Jin-Yuan Wang

國 立 交 通 大 學

運輸科技與管理學系

碩 士 論 文

A Thesis

Submitted to Institute of Transportation Technology and Management

College of Management

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Engineering

In

Transportation Technology and Management

June 2006

Hsinchu, Taiwan, Republic of China

中華民國 九十五 年 六 月

解決具時間窗限制的提送貨問題

學生：黃信翔　　指導教授：王晉元

國立交通大學運輸科技與管理學系碩士班

摘　要

本研究將探討具時間窗限制的提送貨問題(Pickup and Delivery Problem with Time Windows Constraints, PDPTW)，目的為針對 PDPTW 問題建構一數學模式而後求解。首先將 PDPTW 問題建構為集合分割(Set Partition)的整數規劃問題，在考慮貨運需求點、優先限制(Precedence Constraints)、提送貨時間窗、貨物材積、司機人員下班時間等貨運特性下，找出一組時間成本最小的可行車輛路徑組合。

本研究提出一個以變數產生法為基礎的啟發式解法求解這個集合分割問題。於求解過程中將子問題設計為考慮多種提送貨特性的最短路徑問題，以產生可改善主問題目標式的變數(Column)，並提出一個修正後的 Dijkstra’s 演算法來求解子問題。

由測試結果可知變數產生法非常適用於求解 PDPTW 這類變數數量多(規模大)的問題，能在合理的運算時間下求出不錯的路徑組合。

關鍵字：PDPTW，具時間窗限制的提送貨問題，變數產生法

Solving a Pickup and Delivery Problem with Time Window Constraints

Student：Hsin-Hsiang Huang　　Advisor：Jin-Yung Wang

Department of Transportation Technology and Management

National Chiao Tung University

Abstract

This research focuses on the modeling and solution technique of Pickup and Delivery Problem with Time Windows Problems (PDPTW). We first formulate PDPTW as a set partitioning model. This model takes customer requests, service precedence, time windows, vehicle capacity, and working time into account. The objective is to find a set of feasible routes with minimum cost.

A column generation based heuristic method is developed to solve this model. A shortest path problem with multiple side constraints is formulated as the sub-problem in order to find columns which could improve the objective function of the master problem. We also propose a modified Dijkstra's algorithm to solve this shortest path problem.

Numerical experiments indicate that the proposed solution method is sound and promising. The testing results also demonstrate that this method is good for large scale problems.

Keywords: PDPTW; Pickup and Delivery Problem with Time Windows Constraints; Column Generation

致　謝

這篇論文能夠順利完成，首先當然要感謝我的指導教授-王晉元老師，從決定跟著王老師做研究開始，每個禮拜老師都要從百忙之中抽出半個小時指導我的論文，從教導如何搜尋期刊論文、怎麼閱讀期刊論文、討論期刊論文的內容、確定自己的研究方向、研究方法到開始執行，引導我往正確的方向邁進與修正過程中的錯誤，在繳交論文計畫書、期中報告、期末報告與論文初稿的前一兩個禮拜，老師常常須撥出好幾天審閱我的論文並與我討論，有時老師還得犧牲周末的休息時間來學校繼續討論論文，如果沒有老師這樣的辛勞與耐心指導，相信自己絕對沒有能力完成這篇論文。另外，也要感謝我的論文口試老師-中央大學顏上堯老師與中華大學張靖老師，於口試前撥冗審閱學生的碩士論文，口試期間提出多個寶貴的建議與未來可繼續研究的方向，使得本論文能夠更加完善與完整。

接下來就要感謝我親愛的雙親，從出生至念研究所，從呀呀兒語至程式語言的 24 個年頭，父母親無論在精神與物質都是我向前衝的最大後盾，使我能在完全沒有後顧之憂的環境下求學成長，讓我能夠安心的挑戰各種難題與困境，所以同樣的若沒有父母親的養育之恩，我也不可能在交通大學念研究所，當然也就不會有這篇論文的產生，而在論文進行期間常常因為撰寫程式與報告的因素，導致無法時常回家陪伴父母，在此特別感激父母於這段期間的包容與體諒，讓我能有較充裕的時間準備論文並順利完成。另外也要感謝的是遠在美國工作的哥哥，於論文進行期間一直以來的關心與鼓勵，所以雖然撰寫論文時吃了不少苦頭與辛酸，但因為家人的祝福、鼓勵與叮嚀，使得我能重拾信心與力量繼續完成論文，因此希望能與你們一起分享最後的碩果，表達我真摯的愛與謝意。

最後感謝陪我一起求學生活的「超可愛粉紅白嘟嘟」、學長姊們、同學們與學弟妹們，讓我的人生充滿更多歡笑與樂趣，這將都是我一輩子不可抹滅的美好回憶，最後僅將此篇論文獻給所有我認識的至親與好友們，再一次感謝你們。

黃信翔　謹誌

中華民國九十五年六月 于風城交大

目　錄

頁次

表 目 錄

圖 目 錄

第一章 序論

1.1 研究動機

近幾年來隨著全球化與資訊化時代的演進，人們不斷朝著高效率、高品質與低成本等目標邁進，企圖營造出快速便捷且安全可靠的生活環境，各產業也無不加緊腳步跟上時代潮流，紛紛將企業中非核心的競爭區塊外包給專業廠商處理，藉以降低企業營運成本同時可獲得完善的解決方案，使企業能專注於核心競爭區塊，提供更優質的產品或服務，其中明顯的例子之一便是貨物的流通運輸，目前企業均把貨物交由專門的貨運公司運送，不如從前還須特別編制貨車與司機，造成企業營運成本高卻又達不到高水準的運輸品質。

而在消費意識高漲與競爭壓力的促使下，企業客戶將貨物運輸外包給貨運公司後，當然希望得到準時、省時、安全的貨運服務品質，以在分秒必爭的微利時代中贏過對手，因此貨運公司如何滿足企業客戶的貨運需求，提供便捷安心的貨運服務就是他們的核心價值所在，其中最關鍵的便是貨車提貨(Pickup)與送貨(Delivery)的路徑選擇，這往往是貨運公司最關心也是令調度人員最頭痛的的課題之一，因為車輛路徑規劃的好壞會直接影響公司的收益與成本，甚至左右公司的聲譽與顧客忠誠度，良好的路徑規劃不僅可減少車輛行駛的油耗與司機人員的工作時間，同時能增加公司整體貨運的容量與提送貨時間的彈性，以滿足客戶多元化的貨運需求，因此對貨運公司的永續經營而言是相當重要的。

貨物運送指的是託運人(客戶)委託運送人(貨運公司)利用各種運輸工具(Transportation Mode)將貨物從甲地運送至乙地的過程，一般稱甲地為提貨點(Pickup Node)，乙地為送貨點(Delivery Node)，同時託運人會要求運送人提貨的時間與送貨的時間，運送人必須在考量載重材積限制與貨物特性(化學品、冷凍/藏品)於所要求的時間窗內完成提貨與送貨。因此貨運公司調度人員在規劃車輛執行提送貨任務的路徑時，需要考慮提送貨位置、允許提送貨時間窗、貨物材積等基本要素來規劃貨車繞行路徑，然而目前貨車的調度派遣多是由調度人員人工作業，依據調度人員以往的經驗與貨物資訊來指派任務與貨車行走的路徑，而在必須考慮繁多複雜的限制因素下，容易做出不公平與不合理的路徑組合，造成公司人力與資源的浪費，長期如此勢必對於公司的獲利能力造成影響。

一般而言，替一組車輛規劃巡迴路徑的問題，稱為車輛途徑問題(Vehicle Routing Problem, VRP)，在這類問題中每輛車均由場站出發，滿足所有客戶需求點，車輛最終再回到場站。VRP 問題不只適用於上述的貨運公司，諸如計程車行、宅配業者、環保局垃圾車、緊急救援單位(醫院、消防局)等也會面臨這類的問題，這類問題在國內外已有相當廣泛的相關研究。而傳統 VRP 問題不考慮提送貨時間窗與司機人員下班時間等限制下解出車輛經過客戶需求點的順序，有鑑於此，為了因應貨物運送的特性，將上述的限制納入傳統 VRP 問題後，成為具時間窗限制的提送貨問題(Pickup and Delivery Problem with Time Windows， PDPTW)。本研究將針對 PDPTW 問題建構一數學模式，並設計一演算法以求解該數學模式，替調度人員產生一組車輛路徑，同時可滿足貨運需求與貨運限制，並降低車輛巡迴的距離與油耗，提升司機人員的運送效率與工作士氣。

1.2 研究目的

本研究之目的便是在考慮貨物運送的特性下，針對貨運業者的提送貨問題建構一數學模式，並設計一套演算法加以求解該模式，在滿足所有貨運需求點、提送貨時間窗、貨物材積與司機人員下班時間等限制條件下產生一組最小成本的車輛巡迴路徑，期望藉由將車輛路徑的規劃作業自動化，不僅提供較佳的路徑組合，同時降低人工調度負擔，免除因為人工缺失所導致的派遣不公平與不必要的浪費，有效提升貨物運送的效率與降低貨物運送的成本。

1.3 研究範圍

本研究所要探討的研究範圍為針對多車輛、單一車種、單一場站、需求點已知、每個提貨點必有一個送貨點與其配對、每個提貨點與送貨點均有允許執行的時間窗限制的 PDPTW 問題，其中車輛提送貨的型態為車輛均從場站出發，最終再回到場站，並允許車輛混合提送貨，而問題的目標為求解總成本最小的車輛巡迴路徑組合。茲將研究範圍整理如表 1.1：

表 1.1 研究範圍表

項目	型態
車輛數目	多部車輛，且沒有車輛數多寡的限制
車輛種類	單一車種，車輛容量相同
場站數目	單一場站
需求型態	需求點已知且不變
提送貨型態	1. 車輛以空車狀態從場站出發，最終也以空車狀態回到場站 2. 每個提貨點必有一個送貨點與其配對 3. 允許車輛混合提送貨
時間窗型態	每個提貨點與送貨點均有允許執行的時間窗限制
模式目標	車輛巡迴路徑組合的總成本最小

1.4 研究流程

本研究的研究流程如圖 1.1，茲將流程圖中各步驟詳細說明如下：

1. 描述與界定問題

依據貨運公司的現況了解與相關書面資料的勘察，提出目前貨車提送貨之路徑問題，並根據研究動機與目的將問題作一完整的描述與界定。

2. 蒐集與回顧文獻

蒐集國內外探討提送貨問題之相關文獻，並回顧這些文獻之解決方案與演算法，包括最佳解解法(Exact Solution Method)與啟發式解法(Heuristics Algorithm)等，並衡量分析這些解法的優缺點與適用性，從中挑出一個較符合本研究範圍的提送貨問題的解法並改良修正之。

3. 建構車輛提送貨路徑模式

根據現況了解與相關文獻之搜集與回顧，建構一數學模式來描述車輛提送貨路徑的問題特性與相關的限制條件。

4. 設計求解模式之演算法

利用所設計之演算法求解車輛提送貨路徑的問題。

5. 撰寫程式

6. 進行測試範例求解

設計一測試範例來測試數學模式的建構與演算法的設計是否正確且有效率。

7. 分析與評估測試結果

分析求解結果並評估演算法是否正確且有效率。

8. 修正演算法

依據測試結果的分析，若演算法正確且能夠在可容忍的時間內求解則不需要修正，反之則重新設計或修正求解模式之演算法。

9. 結論與建議

對本研究之過程與結果提出結論與建議。

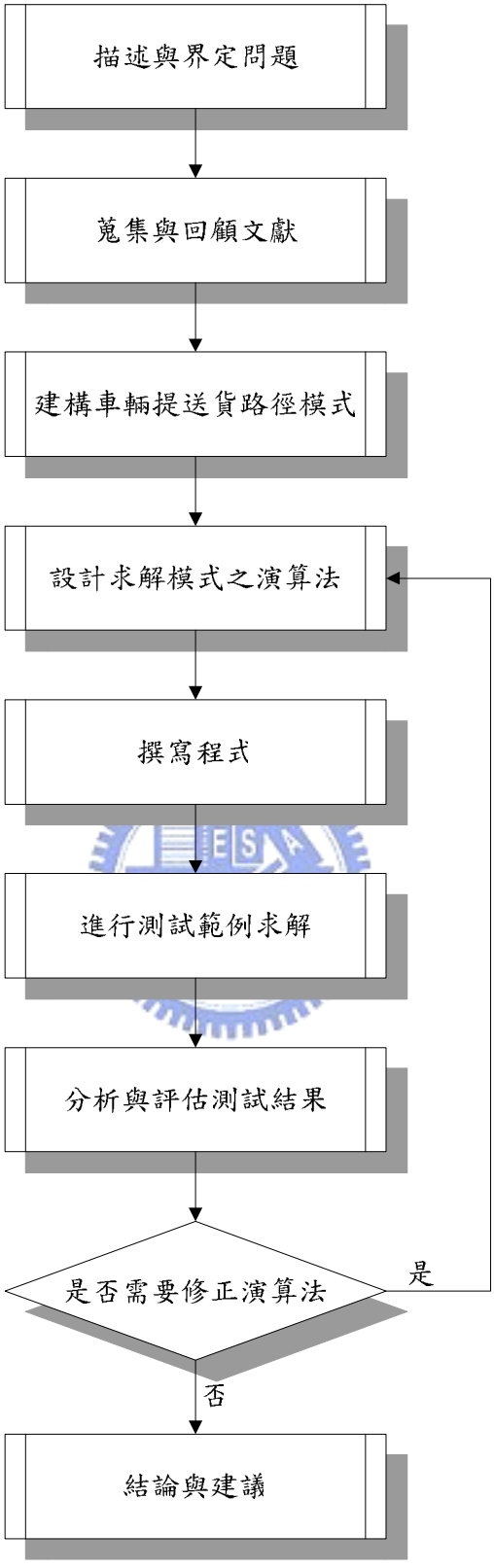
描述與界定問題
蒐集與回顧文獻
建構車輛提送貨路徑模式
設計求解模式之演算法
撰寫程式
進行測試範例求解
分析與評估測試結果
是否需要修正演算法
是
否
結論與建議

圖 1.1 研究流程圖

第二章 文獻回顧

2.1 具時間窗限制的提送貨問題簡介

具時間窗限制的提送貨問題(Pickup and Delivery Problem with Time Windows，PDPTW)[16]在於描述替一隊貨車做路徑規劃，每輛車行經一連串的提貨點(Pickup Node)與送貨點(Delivery Node)以提供貨運服務，在不違反貨車容量的限制下滿足所有託運人的貨運需求，而貨運需求為託運人要求運送人在特定的地點與指定的時間窗內提領貨品，並且期望在限定的時間窗內運送至相對應的目的地，因此每個提貨點都應該有個送貨點與其配對，故在 PDPTW 問題中通常會滿足兩種限制，一為優先限制(Precedence Constraints)，代表針對每個貨運需求而言，執行提貨任務應發生在執行送貨任務之前，另一則為聯結限制(Coupling Constraints)，代表針對每個貨運需求的提送貨任務均由同一輛車執行。

由於貨運業者必須採購或租貸車輛並僱傭司機人員來提供貨運服務，且貨車在繞行時會產生油耗成本與過路費用，以及車輛長期使用下的保養維修費用等龐大的開銷下，會迫使公司主管盡其所能的善用可利用的貨車容量，因此在能滿足所有貨運需求的前提下，若能減少執行任務的車輛數對公司越有利，或是在不能減少車輛數的情況下，若能規劃良好的繞行路徑，使貨車避免因不熟悉路線、任務安排不合理等人為因素而導致的不必要的繞行距離，也可減少公司整體的油耗與過路成本，因此針對 PDPTW 問題而言，所追求的目標通常為使用車輛數最少或是車輛繞行成本最小，其中繞行成本可以是繞行距離、時間或金錢單位等。

在 PDPTW 問題之中，會因所面臨的情況或想要探討的因素有所不同而可區分成好幾類，若依照貨運需求的已知與否可分為動態(Dynamic)與靜態(Static)的提送貨問題[14]；若依照場站數目的多寡可分為多場站(Muti-Depot)與單一場站(Single Depot)的提送貨問題；若依照車輛種類的異同可分為異種車種(Heterogeneous Vehicles)與單一車種(Homogeneous Vehicles)的提送貨問題；若依照車輛可執行的提送貨型態來分，可分為混合式提送貨與單一提/送貨問題，其中混合式提送貨為在貨車離開場站後至回到場站的路徑中，允許執行提貨或送貨任務，但同一貨運訂單的提貨任務要在送貨任務之前執行，若貨車離開場站後須直接執行提貨任務，則貨車應以尚有容量的狀態離開場站，如圖 2.1、2.2 所示，反之若貨車離開場站後須直接執行送貨任務，則應在場站裡就載貨出發，

如圖 2.3 所示；而單一提/送貨則限制車輛從場站出發至回到場站的路徑中，僅能執行提貨或送貨任務，執行提貨任務的車子不可再執行送貨任務，同樣的執行送貨任務的車子不可在執行提貨任務，故車子從場站出發時可能為尚有容量或是載貨狀態，如圖 2.4、2.5 所示。在本研究中所探討的提送貨問題則允許貨車混合式提送貨，而貨車從場站離開時均為空車狀態，而後執行一連串的提送貨配對(訂單)，最後再以空車狀態回場站，如圖 2.1、2.2 之情況。

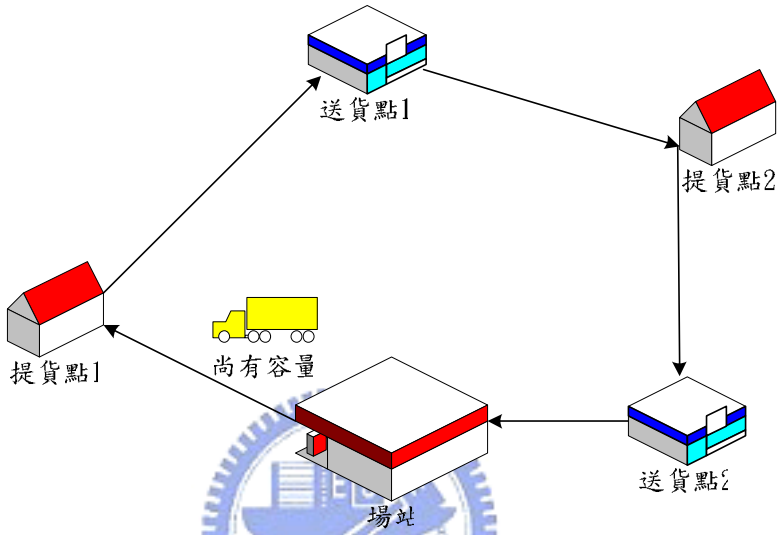

圖 2.1 混合式提送貨情況一

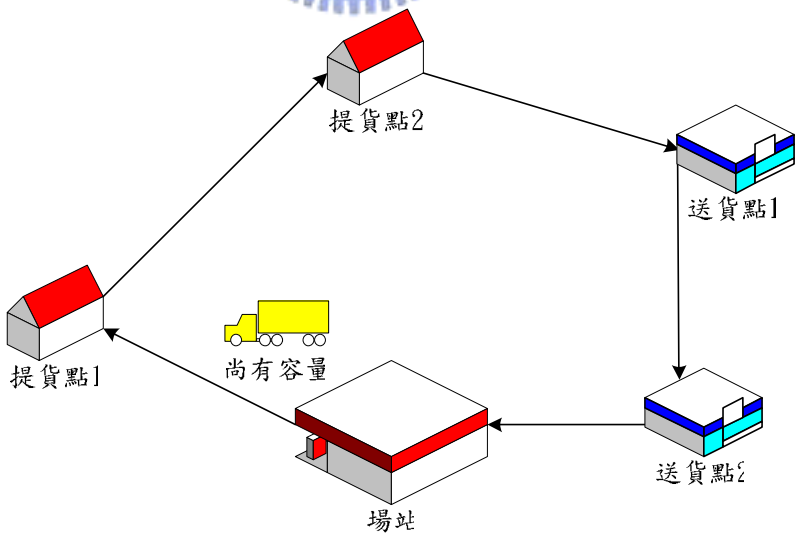

圖 2.2 混合式提送貨情況二

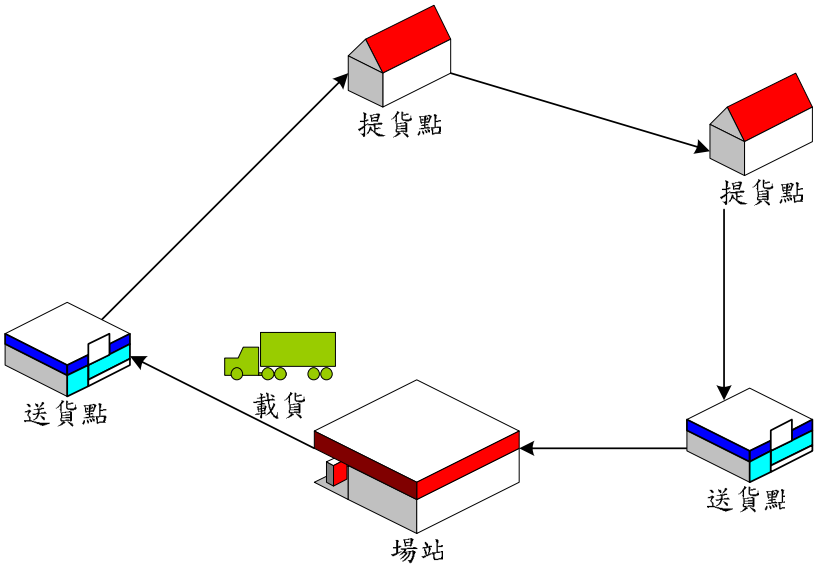

圖 2.3 混合式提送貨情況三

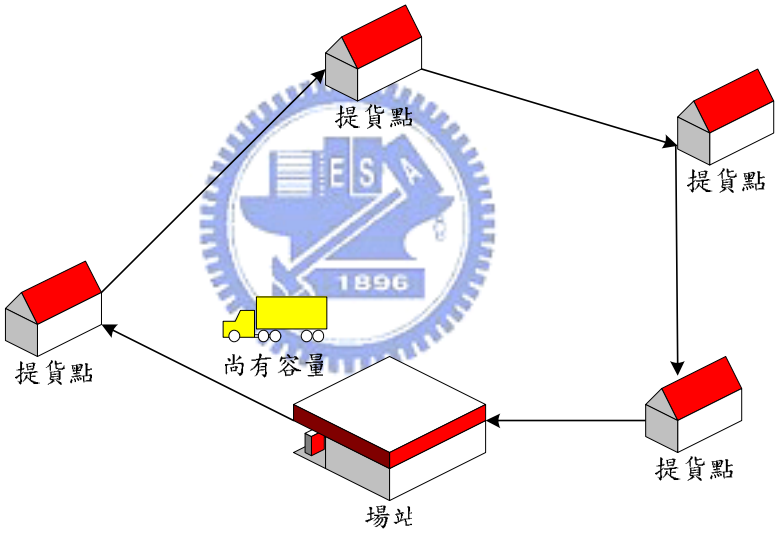

圖 2.4 單一提貨

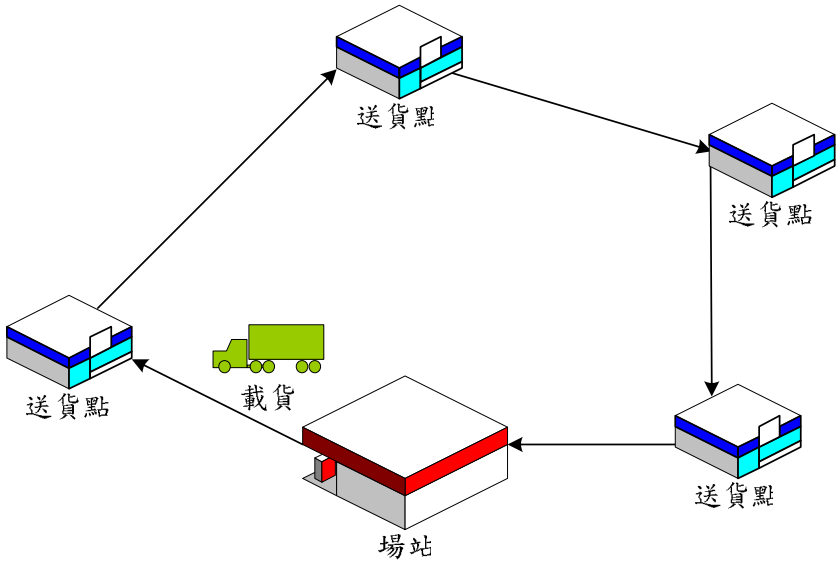

圖 2.5 單一送貨

不論所面臨的情況為何，我們均可以用以下的數學方式表示一個 PDPTW 問題。假設 N 為託運者的貨運需求集合，令 $N=\{P_1,P_2,\ldots,P_n,D_1,D_2,\ldots,D_n\}$，其中 P_k 代表第 k 筆訂單的提貨任務，D_k 代表第 k 筆訂單的送貨任務。令 $P=\{P_1,P_2,\ldots,P_n\}$ 代表提貨任務集合，$D=\{D_1,D_2,\ldots,D_n\}$ 為送貨任務集合，每個提貨任務必與一個相對應的送貨任務配對，如 (P_1, D_1) 所示，故 $P\cup D=N$，$|N|=2n$，$|P|=|D|=n$，即有 n 個任務配對。任一任務 $t\in N$ 均有個允許執行的時間窗 $[E_t,L_t]$ 與貨物載重材積 W_i，其中 E_t 代表允許執行任務 t 的最早時間，簡稱最早時間，L_t 代表允許執行任務 t 的最晚時間，簡稱最晚時間，若任務 $t\in P$ 則 $W_i>0$，反之若任務 $t\in D$ 則 $W_i<0$。而任務 i 到任務 j 的旅行時間則以 t_{ij} 表示，通常為了方便起見會將執行任務所需的時間自動加入 t_{ij} 中，不作明顯特別的考慮，並且為了防止貨車先執行某提送貨配對(訂單)的送貨任務才執行提貨任務，將 t_{ji} 設為 M，其中 $j\in D$，$i\in P$，$(i,j)=(P_k, D_k)$，M 為一個很大的數字，使得進行路徑規劃時不會選擇到這樣的路徑。最後令 V 代表可用的車輛集合，$|V|=m$，且貨車之容量為 C。

2.2 建構 PDPTW 問題模式

在求解 PDPTW 問題時，可依據問題特性與假設條件建構一個數學模式，而後再發展一適合的解法來求解，如 Xu et al. [4]、Lau et al. [5]、 Gronalt et al. [9]、Avella et al. [12] 等均使用此方式，其中 Xu、Gronalt 與 Avella 等人所提出的模式的決策變數為路徑 (route/tour)，當該路徑被選擇時則對應的決策變數則為 1，反之則為 0，此三位的模式均須滿足所有貨運訂單必須被服務且僅能一次的限定，除此之外 Xu 與 Avella 的模式還設

定路徑數須小於車輛數的限制，而 Gronalt 的模式則設定了許多時間上的限制，例如允許提送貨的時間窗限制、兩任務節點間的旅行時間限制、路徑的起始時間限制等。另外 Lau 所提出的模式的決策變數則代表路段(arc/link)，當該路段被選擇時則對應的決策變數則為 1，反之則為 0，設定的限制包含滿足所有貨運訂單必須被服務且僅能一次、貨車須從場站離開且最終須回到場站、貨車容量以及優先限制(Precedence Constraints)時間窗限制等。由以上的數學模式可知，PDPTW 問題可建構成整數規劃(Integer Programming)問題而後求解，以下以 Avella et al. [12]所提出的 PDPTW 模式為例：

$$\text{Min} \quad \sum_{j=1}^{n_c} c_j x_j$$

$$\text{S.T.} \quad \sum_{j=1}^{n_c} g_{ij} x_j = 1 \text{ ， } i = 1,...,n_o \qquad (2\text{-}1)$$

$$\sum_{j=a(k)}^{b(k)} c_j x_j \le T_k \text{ ， } k = 1,...,n_t \qquad (2\text{-}2)$$

$$x_j \in \{0,1\}$$

其中 x_j 為二元變數，當路徑 j 被選到時 x_j=1，反之 x_j=0；c_j 為路徑 j 的時間成本；n_c 為可行的路徑數目，亦即變數的(Column)數目；g_{ij} 為二元參數，當貨運訂單 i 為路徑 j 所服務時 g_{ij}=1，反之 g_{ij}=0；$n_c(k)$為第 k 輛車的可行路徑數目，令 a(k)= $n_c(k-1)+1$，b(k)= $n_c(k)$；n_o 為貨運訂單數目；n_t 為車輛數目；T_k 為車輛 k 的最大工作時間。(2-1)式為要求每個貨運需求都須被某車輛服務到，且僅有一次，(2-2)式要求每部車的工作時間不可超過最大工作時間。故欲解決 PDPTW 問題時可先將問題建構為整數規劃模式來求解，因此本研究將於下面小節中討論求解整數規劃問題的解法與解決的問題。

2.3 PDPTW 問題求解方法

由於貨車所繞行的提送貨路徑對於貨運公司的獲利能力與司機人員的運送效率有直接且重大的影響，因此相關的演算解法不僅在學術上有解決問題的貢獻度，在實務上也能有效應用來提供更優質的貨運服務，故長久以來一直是許多學者深入探討研究的課題之一[1][2][3][4][5][8][9][10][11][12][14][16]。

由上一節可知在求解 PDPTW 問題時，可將問題建構成整數規劃模式而後求解，而目前求解整數規劃的解法可分為兩大類，一為最佳解解法(Exact Solution Method)，另一則為啟發式解法(Heuristics Algorithm)，其中最佳解解法以分支限定法(Branch-and-Bound

Method)為大宗，然而經由證明 PDPTW 問題屬於 NP-Hard 問題[5]，若使用最佳解法求解大多僅能處理規模較小的問題，且求解過程須耗費大量的運算時間，通常無法在有效的時間內獲得最佳解，因此通常不會考慮以最佳解法來求解 PDPTW 問題，取而代之的就是使用啟發式解法，能夠在較短的時間內獲得品質不錯的近似最佳解，故文獻上許多數學者均利用啟發式解法來求解 PDPTW 問題[3][5][8][9][14][12][16]。本研究將於 2.4 小節回顧將問題建構為整數規劃問題而後利用最佳解基礎的演算法來求解之相關文獻，於 2.5 小節回顧 PDPTW 的啟發式解法與解決的問題。

2.4 回顧整數規劃問題與解法

由 2.2 小節可知在求解 PDPTW 問題時，可建構成整數規劃模式而後求解，故本小節先針對求解整數規劃的相關文獻與用來解決的問題作一回顧：

Gademann and Velde [11]探討的問題為將訂單分批處理，在平行通道的倉庫中提領所有項目的總旅行時間最小化問題。該研究首先提出該問題的運算時間複雜度(Computational Complexity)，並證明該問題屬於 NP-Hard 問題，但是若能夠限制每批次不包含超過兩個訂單以上則可以在多項式時間(Polynomial Time)內解決。之後，該研究發展了一套 branch-and-price 最佳化演算法來解決該問題，為此先將該問題建構為一般的集合分割(Set Partition)問題，而後使用變數產生法(Column Generation)來求解集合分割問題的線性規劃放鬆(Linear Programming Relaxation)，由實驗結果顯示最佳化演算法的績效令人注目。

Sung and Hong [1]探討一個多重傳送路徑(Multicast Routing)問題，在此問題中訊號必須從來源點(Source Node)傳送至多個目的點(Destination Node)，並且找出在通訊網路中最小的成本樹，其中所有通訊節線的延誤值為事先給予的。作者將此問題以整數規劃問題表示，利用 branch-and-price 演算法求解最佳整數解，首先利用相關的問題特性來減少解空間，再使用變數產生法來求解已經減少解空間的線性規劃放鬆，最後提出一套分支策略取得整數解，由實驗結果顯示該演算法可以在合理的時間內解決實際大小的問題。

Savelsbergh [10]探討一般化指派問題(Generalized Assignment Problem)，該問題為找出如何將多個任務指派給多個代理人，使得利潤最大化的指派方式，同時必須滿足每個任務僅被指派給一個代理人，且不得違反每個代理人的容量限制。作者提出一套新的演

算法來解決該問題，稱為 branch-and-price，首先將該問題轉為一般的集合分割問題，即為一整數規劃問題，作者討論了許多分支的策略以使得在分支定限樹中(Branch-and-Bound Tree)的每個節點可以使用變數產生法，藉由融合變數產生法與分支定限法來求得最佳整數解。

Barnhart et al. [2] 討論具有大量變數的整數規劃問題，包括一般化指派問題(Generalized Assignment Problem)與人員排班問題(Crew Scheduling Problem)，一般化指派問題已於上一篇文獻討論過，在此不再贅述，而人員排班問題又可稱為配對問題(Pairing Problem)為探討將每部機具僅分派給一個人員的問題，機具可以為飛機、公車、捷運列車、火車等。作者使用變數產生法與分支定限法來解決這類大型的整數規劃問題，在分支定限樹上的每個節點上不是產生變數就是證明線性規劃的最佳性，同時作者也討論了變數產生法與分支定限法在執行上的課題，如分支策略以及解決線性規劃放鬆的有效方式。

從上述文獻中可知變數產生法常被用來求解變數多的數學規劃問題，因此以下介紹何謂變數產生法以及解題的步驟流程。變數產生法是由 Dantzig 與 Wolfe 兩位學者於 1960 年所提出的方法，又稱為 Dantzig-Wolfe Decomposition[17][18]。主要概念為利用線性規劃中的對偶理論(Dual Theory)來產生對於改善目標值有貢獻的變數(Column)，以避免耗費龐大的時間來窮舉沒有貢獻的變數。因此，變數產生法常用於解決具有龐大變數的線性規劃問題(Linear Programming Problem)，常見的如集合涵蓋問題(Set Covering Problem)與集合分割問題(Set Partition Problem)，由於以求解的可行性與方便性來說，將欲解決的問題轉為集合涵蓋問題會較轉為集合分割來的好，故多數研究會將問題轉為集合涵蓋問題，之後再做適當的調整機制以滿足集合分割的要求。以下就以集合涵蓋問題之求解來說明變數產生法的步驟流程，在此假設一個集合涵蓋問題如以下所示：

$$\begin{aligned} \text{Min} \quad & \sum_{r\in R} c_r x_r \\ \text{S.T.} \quad & \sum_{r\in R} a_{kr} x_r \geq 1 \text{，} \ \forall k \in K \qquad \text{(2-3)} \\ & x_r \geq 0 \text{，} \ \forall r \in R \end{aligned}$$

上述的模式即為集合涵蓋問題中的主問題(Master Problem)。在建構主問題時，由於所面臨的問題特性常常很難去列出所有變數，因此多以啟發式解法或人工方式找到一組

初始可行解的變數，再以單體法(Simplex Method)求解目前主問題的最佳解。

接下來依據對偶理論，將主問題轉為對偶問題(Dual Problem)，令 π_k 為對應於(2-3)式的對偶變數，故轉換後的對偶問題如下所示：

$$\begin{aligned} \text{Max} \quad & \sum_{k\in K} \pi_k \\ \text{S.T.} \quad & \sum_{k\in K} a_{kr}\pi_k \le c_r \ ，\ \forall r \in R \qquad (2\text{-}4) \\ & \pi_k \ge 0 \ ，\ \forall k \in K \end{aligned}$$

之後則建構子問題來產生對於改善目標值有貢獻的變數(Column)，多數情況下均把子問題建構為最短路徑問題(Shortest Path Problem)，將上述的對偶問題中的對偶值傳入子問題中求解，若判斷求解出來的變數可改善目標值，則加入主問題中重新求解，重複以上步驟，直到無法再產生新的變數，便知道目前包含的變數已經達到最佳解。而判斷子問題所產生的變數是否具有貢獻度的法則是依據對偶可行性(Dual Feasibility)來確認，由(2-4)式推導可知，如果對於其他尚未考慮的變數(Column)均滿足 $c_r - \sum_{k\in K} a_{kr}\pi_k \ge 0$，則不用再考慮這些尚未加入的變數，因為他們均無法改善目標值，此時則找到最佳解；反之，可以從尚未加入的變數中尋找到一個可使 $c_r - \sum_{k\in K} a_{kr}\pi_k < 0$ 的變數加入主問題重新求解。若所得到的最佳解為整數解，即為原來集合涵蓋或集合分割的最佳解，若不為整數解，則文獻上通常再利用分支定限法(Branch-and-Bound Method)來求得整數解[17][18]。

2.5 PDPTW 問題之啟發式解法

啟發式解法的概念是會先產生一個可行初始解，然後在初始解的鄰近區域(Neighborhood)內搜尋可以改善目標值的解，這個動作稱為鄰近或區域搜尋(Local Search)，然而為了避免僅找到區域最佳解(Local Optimum)，各種啟發式解法通常都有其特殊的機制跳脫區域最佳解，到其他未搜尋的解空間搜尋，這個動作則稱為全域搜尋(Global Search)，雖然啟發式解法無法確定可以找到最佳解，但往往可以保證得到品質不錯的近似最佳解，在面臨大型問題或有時效性的問題時是較好的求解方法。以下針對使用啟發式解法來解決 PDPTW 問題的文獻作一討論說明：

Nanry and Barnes [16]提出一個適應性禁忌搜尋法(Reactive Tabu Search)來解決

PDPTW 問題，在滿足優先限制(Precedence Constraints)與聯結限制(Coupling Constraints)情況下使用三種不同的移動/交換方式獲得新的鄰近解，並且設計了一套選擇鄰近解的階級策略，使能夠動態地在三種鄰近解中選擇並且修正搜尋的方向，另外作者也提出一套方法來檢測是否掉入區域最佳解的陷阱，並提供跳出區域最佳解的方式。最後作者為了驗證適應性禁忌搜尋法的效度，以索羅門(Slomon)[15]的具時間窗限制的車輛路徑問題(Vehicle Routing Problem with Time Windows，VRPTW)之標竿例題為基礎，創造適用於 PDPTW 問題的標竿例題做演算法測試，由測試結果得知無論解的品質或求解速度都有不錯的表現。

Li and Lim [3]提出一種巨集啟發式解法來解決 PDPTW 問題，稱為禁忌嵌入式模擬退火法(Tabu-Embedded Simulated Annealing Algorithm)，追求的目標為最小化使用車輛數、總旅行成本、排程期間與司機人員等待時間的總和，上述的四個考慮因素均有相對應的權重值，以差別出因素之間的相對重要性。禁忌嵌入式模擬退火法為當目前的最佳解在經過數個預定的巡迴搜尋還沒有改善時開始啟動運作，最後將演算法套用於修改自索羅門(Slomon)的 VRPTW 標竿例題而新產生的六組測試範例，由測試結果得知該方法為解決多種車輛 PDPTW 問題的第一方法。

Lau and Liang [5]提出一個兩階法(Two Phase Method)來解決 PDPTW 問題，追求的目標為最小化使用車輛量與總旅行距離的總和。在兩階法的第一階段中，提出一種融合傳統插入法(Insertion Heuristic)與掃描法(Sweep Heuristic)的優點的新式構建演算法來產生初始解，在第二階段中則使用禁忌搜尋法(Tabu Search)來改善解的目標值，透過三種不同的鄰近區域移動方式來找尋最佳解。最後作者將該演算法套用於修改自索羅門的 VRPTW 標竿例題而新產生測試範例，由測試結果得知與標竿解相比之下該方法的解是非常好的。

Gronalt et al. [9]探討貨車滿載情況下的 PDPTW 問題，由於作者認為空車的繞行對於貨物運送上沒有任何附加價值，故將目標放在追求最小化空車的移動距離。針對該問題作者首先建構一個最佳完整的數學模式，再依據網路流量的關係放鬆最佳數學模式，藉由這個放鬆後的數學模式可計算出目標式的下限值(Lower Bound)，此外作者提出四種以節省法(Savings Algorithm)為基礎的新式啟發式解法，並使用一小型範例詳細說明該演算法之步驟流程。最後作者使用隨機產生的範例來測試新式演算法，由測試結果得知均

可以快速的找到品質非常好的解。

Snežana et al. [14]探討快遞業者所面臨的動態 PDPTW 問題，業者必滿足同天內(Same-Day)的提送貨要求，且貨運需求並非可使用隨機模式(Stochastic Model)預測的。動態 PDPTW 的標準解法是利用由 Psaraftis[13]所提出的滾動時間軸(Rolling Time Horizon)，當要指派新的任務給某輛車時，也許要同時考慮這樣的指派對於短期時間軸與長期時間軸的影響比較好，尤其在特殊情況下，良好的管制車輛閒置/等待時間對於長久而言可能會減少總路徑成本，因此作者提出一個修改自插入法(Insertion Heuristic)的雙時間軸(Double-Horizon)啟發式解法來解決動態 PDPTW 問題，其中雙時間軸即考慮長期與短期。測試結果顯示使用雙時間軸啟發式解法可提供不錯解品質。

Renaud et al. [8]探討提送貨的旅行銷貨員問題(Pickup and Delivery Traveling Salesman Problem, PDTSP)，在這個問題下要找出經過場站與數個提送貨配對的漢米爾頓迴圈(Hamiltonian Cycle)。針對這個問題作者提出三種不同擾動啟發式演算法(Perturbation Heuristics)，可視為多種幫助演算法跳脫區域最佳解的機制，如同禁忌搜尋法一般，這三種分別為 Instance Perturbation(IP)、Algorithmic Perturbation(AP)與 Solution Perturbation(SP)。由測試結果顯示 SP 解的品質優於 AP，AP 解的品質又優於 IP。

Avella et al. [12]探討一個燃料公司運送燃料的問題，該公司必須利用容量均不同的異質車隊將各種燃料從一個大型的中心燃料槽運送至分屬各地的小燃料槽 (fuel pump)，追求的目標式為使用可得的資源(車輛與司機人員)滿足所有訂單且最小化運送的總旅行距離。本研究提出啟發式與最佳解法來解決該問題，啟發式解法則取自節省法的概念，不論針對大小問題均可快速得到一可行解；而最佳解法則是先把問題以集合分割模型(Set Partition Model)表現，以作者提出的啟發式解法先求出初始可行解，再用 Branch-and-Price 演算法求得最佳解。測試範例則是由燃料公司的實際數據得來，測試結果顯示使用啟發式解法得到的解比公司目前的總旅行成本與車輛數少，若使用最佳解法求解則可以得到更好的品質解。

2.6 小結

經由回顧 PDPTW 問題之相關文獻可知，多數研究選擇使用啟發式解法來求得品質不錯的近似最佳解，以期望能夠解決較大規模的問題，或是能夠較有效率地得到一組可行解，但為了求解速度而使用啟發式解法，在廣闊的解空間中要找到最佳解幾乎是不可

能的，且容易陷入區域最佳解的陷阱中，若沒有完善的跳脫機制將造成嚴重的缺失。而本研究所探討的 PDPTW 問題並非要求在極短時間內即時快速的產生路徑供調度人員使用，反之希望的是在可接受的運算時間內求得一可行的較佳路徑組合即可，故可考慮採用最佳解基礎的演算法求解，以避免啟發式解法在搜尋解空間時本質上的缺點，故本研究將不採用啟發式解法來解決 PDPTW 問題，而是利用最佳解基礎的演算法來求解，融合變數產生法與分支定限法，並設計一套修正的演算法來求解變數產生法中的子問題。

第三章 研究方法

由於本研究所探討的 PDPTW 問題並非要在短時間內快速產生車輛的巡迴路徑，因此將採用最佳解基礎的演算法來求解，以避免使用啟發式解法陷入區域最佳解的缺失。首先將具時間窗限制的提送貨問題建構成整數規劃問題，由於提送貨問題複雜度大，屬於 NP-Hard 問題[5]，要窮舉所有的車輛路徑來找出最佳解是非常耗費時間的或甚至不可能，故在本研究中將採用變數產生法(Column Generation Method)來求解，因為變數產生法在求解的過程中一次只考慮部份的變數(Column)，利用主問題與子問題間的訊息傳遞，主問題將對偶變數值傳給子問題，子問題將新產生的變數傳回主問題，透過對偶可行性來驗證新變數的貢獻度，逐步得到最佳解，如此將可增進求解效率，並可避免因變數過多而導致無法求解。若使用變數產生法求出的最佳解並非為整數解時，再結合分支定限法(Branch and Bound)以獲得整數解。

3.1 定義名詞

本小節將定義一些在研究架構流程中常見的名詞，說明如下：

1. 任務：貨運公司接獲託運者的貨運需求(訂單)後，即產生一個提貨任務與送貨任務的配對，貨運公司必須派車前往提貨點，並在指定的時間窗內執行提貨任務，之後前往送貨點並在指定的時間窗內執行送貨任務，如此完成此筆貨運需求。在位置表示上提貨點即代表一提貨任務，送貨點即代表一送貨任務。

2. 巡迴路徑：每輛貨車從場站出發後執行一連串的任務，最終再回到場站的任務鏈稱為巡迴路徑，簡稱路徑。如圖 3.1 所示，貨車從場站出發後行走的巡迴路徑為：場站-提貨任務 1-送貨任務 1-提貨任務 2-送貨任務 2-場站。

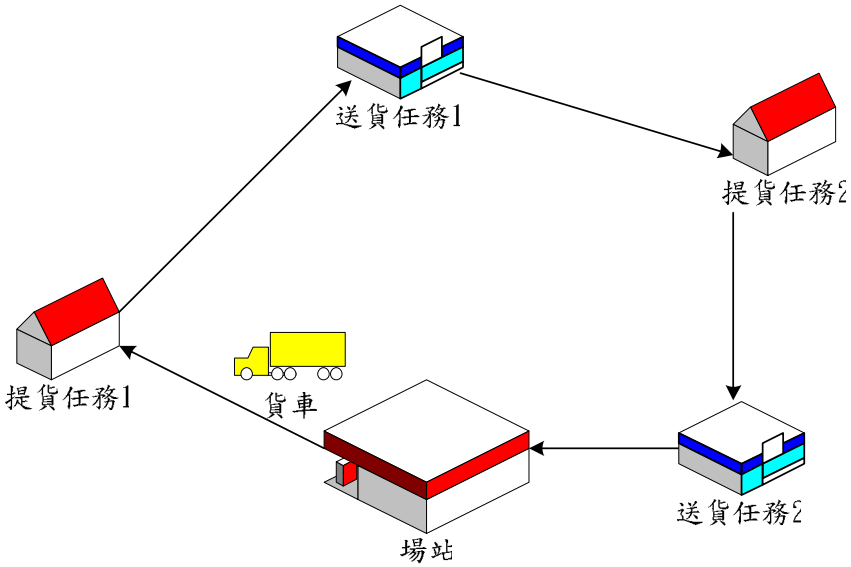

圖 3.1 巡迴路徑示意圖

3.2 本研究所探討的 PDPTW 問題

本研究將探討的 PDPTW 問題有以下幾點特性，茲將歸納如以下十點所示：

1. 多車輛但單一車種，故每輛車的可載重容量相同。
2. 司機人員有預定的下班時間，此即為貨車可執行任務的最晚時間。
3. 僅考慮一個場站。
4. 每個貨運需求即代表一組提貨與送貨的配對，故每個提貨任務必有一個送貨任務與其配對，而每個送貨任務也必有一個提貨任務與其配對。
5. 每個任務均有允許執行的時間窗，車輛僅能在此時間窗內執行任務，若於時間窗前到達時則須等待至最早允許執行任務的時間，如此將產生等待時間，反之若於時間窗內到達則可直接執行任務，將沒有等待時間。
6. 每個任務均有貨物載重材積，若為提貨任務則貨物載重材積為正值，若為送貨任務則貨物載重材積為負值。
7. 貨運需求已知不再變動，亦即提貨任務與送貨任務已知。
8. 每輛車均以場站為出發點，最終再回到場站。
9. 每輛車均為混合式提送貨，如 2.1 小節中所述。

10. 本 PDPTW 問題之成本僅考慮時間，故求解目標為得到一組花費時間最少的車輛巡迴路徑組合。

3.3 模式架構

在考量 PDPTW 問題具有路徑組合規模過大的情形下，本研究將藉由變數產生法(Column Generation Method)來求解 PDPTW 問題，該方法將欲求解的問題分成主問題與子問題來求解，一次只考慮一部分的可行變數(Column)，求解主問題後可得一組對偶變數值，將對偶變數值傳遞給子問題，再求解子問題以期望產生對於主問題有貢獻度的變數(Column)，是否具有貢獻是由對偶可行性來驗證。若驗證後為不滿足對偶可行性則代表此變數具有貢獻度，故將此變數加入主問題繼續求解，以逐步獲得最佳解，反之則表示此變數不具有貢獻度，不用再將此變數加入主問題中。如此週而復始，直到所有未產生的變數均滿足對偶可行性，以完成變數產生法的整體解題流程。

在使用變數產生法的架構下，本研究中將 PDPTW 問題建構為集合分割問題(Set Partition Problem)，並放鬆整數限制後成為一線性規劃問題，將之視為變數產生法中的主問題(Master Problem)，而後產生一組主問題的可行初始變數，利用單體法(Simplex Method)求解主問題以獲得對偶變數值，接下來將子問題設計成一種具有多項提送貨特性的最短路徑問題，將從求解主問題中得到的對偶變數值代入此最短路徑問題，之後本研究發展一個以 Dijkstra 演算法為基礎的最短路徑演算法來求解該種最短路徑問題，求解出的最短路徑即代表主問題中的一個可行變數(Column)，此時則藉由對偶可行性來判斷是否將此變數加入主問題中繼續求解，若不滿足對偶可行性則加入主問題，反之則得知已獲得主問題線性規劃放鬆的最佳解，如此則停止變數產生法的解題流程。由於主問題為一線性規劃問題，故變數產生法的最佳解可能為非整數解，此時則利用分之定限法(Branch-and- Bound)來求得整數解，若完成變數產生法後所得的最佳解即為整數解，則得知找到 PDPTW 的最佳解，如此將可結束整個模式架構。本研究之模式架構如圖 3.2 所示，茲將架構圖中各步驟詳細說明如下：

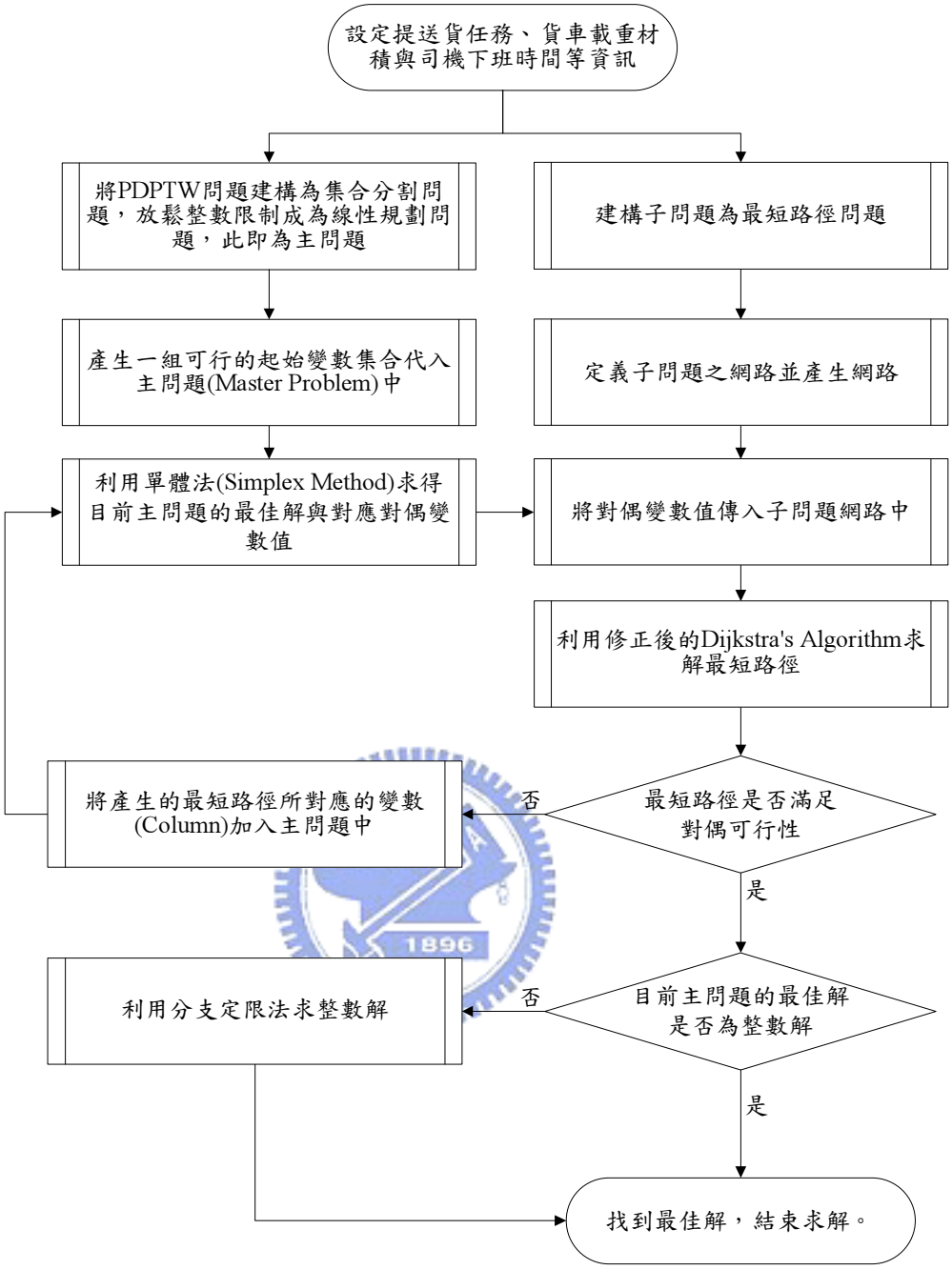
設定提送貨任務、貨車載重材積與司機下班時間等資訊
將PDPTW問題建構為集合分割問題，放鬆整數限制成為線性規劃問題，此即為主問題
建構子問題為最短路徑問題
產生一組可行的起始變數集合代入主問題(Master Problem)中
定義子問題之網路並產生網路
利用單體法(Simplex Method)求得目前主問題的最佳解與對應對偶變數值
將對偶變數值傳入子問題網路中
利用修正後的Dijkstra's Algorithm求解最短路徑
將產生的最短路徑所對應的變數(Column)加入主問題中
否
最短路徑是否滿足對偶可行性
是
利用分支定限法求整數解
否
目前主問題的最佳解是否為整數解
是
找到最佳解，結束求解。

圖 3.2 模式架構圖

1. 設定提送貨任務、貨車載重材積與司機下班時間等資訊

依據 3.2 節的問題特性描述，設定所有提送貨任務、貨車載重材積與司機下班時間等資訊，作為演算法讀取資料來源，每一筆任務所包含的資訊如表 3.1 所示：

表 3.1 任務資訊表

訂單編號	節點編號	任務類型	最早時間	最晚時間	載重材積
1	2	提貨	2	5	9
1	5	送貨	6	8	-9

其中「節點編號」為此任務在網路中所代表的節點號碼，「任務編號」為此任務所屬的訂單號碼，「任務類型」則表示此任務為提貨任務或送貨任務，「最早時間」為允許執行該任務的最早時間，「最晚時間」為允許執行該任務的最晚時間，「載重材積」為此任務的貨物載重材積，當此任務為提貨時則載重材積為正值，若為送貨則載重材積為負值，而訂單編號相同的提貨任務與送貨任務稱為一任務配對，任務配對之載重材積加總為零。

2. 將 PDPTW 問題建構為集合分割問題，放鬆整數限制成為線性規劃問題

本研究將 PDPTW 問題構建成集合分割問題(Set Partitioning)型式，其問題定式如下：

$$\text{Min} \quad \sum_{r \in R} c_r x_r$$

$$\text{S.T.} \quad \sum_{r \in R} a_{tr} x_r = 1 \text{，} \forall t \in T \qquad (3\text{-}1)$$

$$x_r \in \{0,1\} \text{，} \forall r \in R \qquad (3\text{-}2)$$

其中 x_r 代表一路徑，當路徑 r 被選到時 $x_r=1$，反之 $x_r=0$；c_r 為路徑 r 的時間成本，成本的定義將於第四章說明；R 為所有可行路徑之集合，因此目標式為追求所有可行路徑的總成本最小；在(3-1)式中 a_{tr} 為二元參數，當任務 t 為路徑 r 所服務時 $a_{tr}=1$，反之 $a_{tr}=0$；T 為所有任務之集合。故(3-1)式為要求每個任務都須被服務到，且僅有一次；而(3-2)式僅是設定 x_r 為二元變數。本研究並不考慮執行任務的車輛數，故於上述的集合分割問題中並未加入與車輛數相關的限制式。

將上述的集合分割問題放鬆整數限制式，即(3-2)式，成為線性規劃問題，即為變數產生法之主問題(Master Problem)。

3. 產生一組可行的起始變數集合代入主問題中

限定每一輛車僅服務一筆訂單，即每輛車僅經過一對提貨點與送貨點，如此產生與訂單筆數相同的車輛巡迴路徑，而每個路徑所相對應的變數即為起始變數集合，而後帶入主問題中。

4. 利用單體法(Simplex Method)求得目前主問題的最佳解與對應對偶變數值

由於主問題為集合分割問題放鬆整數限制後之線性規劃問題，故本研究使用單體法來求解線性規劃問題，將利用 CPLEX 軟體來求解目前主問題的最佳解，並獲得對應於(3-1)式之對偶變數值，即對應於任務 t 之對偶值。

5. 建構子問題為最短路徑問題

本研究將針對子問題建構成一最短路徑問題，求解出的最短路徑即代表主問題中的一個變數(Column)，之後依據對偶可行性來判斷是否將此新變數加入主問題中繼續求解。

6. 定義子問題之網路並產生網路

定義子問題網路的節點、節線與節線成本而後產生網路，本部份將於第四章作詳細的介紹。

7. 將對偶變數值傳入子問題網路中

將第四步驟利用數學軟體求解獲得的對偶變數值π_j傳入子問題網路中，π_j為所對應(3-1)式 j 任務的對偶變數值，如此將可求出節線成本，進而使用最短路徑演算法求解。

8. 利用修正後的 Dijkstra's Algorithm 求解最短路徑

由於本研究所設計的子問題網路為滿足許多提送貨特性的網路問題，欲於這類網路問題中尋找最短路徑時，並非傳統的最短路徑演算法可直接解決的，故本研究將修正傳統的 Dijkstra's 演算法來求解子問題，以在能滿足各種提送貨限制下找出總節線成本最小的路徑，詳細的演算法流程將於第四章介紹。

9. 判斷最短路徑是否滿足對偶可行性

利用對偶理論中的對偶可行性來判斷利用修正後的 Dijkstra's 演算法求得的最短路徑對於主問題是否有貢獻度，若滿足對偶可行性則得知已達最佳解；反之若則代表此路徑對於主問題目標式有貢獻，將對應此路徑的變數加入主問題，重複 4、7、8 步驟繼續求解。

10. 判斷目前主問題的最佳解是否為整數解

若目前主問題的最佳解即為整數解，則得知已找到最佳解，求解過程結束；反之則須利用分支定限法(Branch and Bound)求得最佳整數解。

第四章 建構與求解子問題

本研究於第三章中針對具時間窗提送貨問題的研究方法做一整體性的敘述，得知本研究將採用以最佳解為基礎的演算法來求解，在考量求解的時效性與提送貨問題的龐大規模下，本研究將採用變數產生法(Column Generation Method)來求解，而變數產生法主要是由主問題與子問題兩部份構成，這兩部份的設計與求解方法均因問題的性質不同而異。在本研究中子問題的部分如 3.3 節所述，將子問題設計並建構為一最短路徑問題，之後藉由修正後的 Dijkstra's Algorithm 來求解最短路徑，求解出的最短路徑即代表主問題中的一個可行變數(Column)，再依據對偶可行性的判斷來決定是否加入主問題中。

4.1 建構子問題之網路

本章主要針對變數產生法中建構子問題之網路作一詳細的介紹，故必須先定義網路中的節點、節線與節線成本，始能建構出子問題的網路，以利後續變數產生法之求解流程，詳細的定義如下說明所式：

1. 節點：

   A. 子問題網路中，每一任務代表一節點，令 $T=\{1_P,2_P,\ldots,n_P,1_D,2_D,\ldots,n_D\}$為任務集合，其中 K_P 代表第 K 筆訂單的提貨任務，K_D 代表第 K 筆訂單的送貨任務，每個提貨任務必與一個相對應的送貨任務配對，如$(1_P, 1_D)$所示。

   B. 節點資訊包括節點編號、訂單編號、任務類型(提貨或送貨)、時間窗$[ET_t,LT_t]$、載重材積，其中 ET_t 代表允許執行任務 t 的最早時間，簡稱最早時間，LT_t 代表允許執行任務 t 的最晚時間，簡稱最晚時間。

   C. 節點 O、D 代表場站，車輛皆由場站(O)出發執行任務，最終回到場站(D)，因此節點 O 可視為網路之起點，節點 D 可視為網路之迄點。

2. 節線：

   (1) 節點 O、D 間不能有節線直接相連

   (2) 節點 O 與所有的提貨節點相連，節點 D 與所有的送貨節點相連。

   (3) 任兩任務節點 i 與 j，意即 $i,j\in T$，i,j 節點間須符合下列條件即可相連：

A. $ET_i + Time(i,j) \leqq LT_j$，其中 Time(i,j)為任務節點 i 到任務節點 j 的實際旅行時間，不包含等待時間。

B. $ET_i + Time(i,j) \leqq EndT$，其中 EndT 代表司機人員的下班時間，車輛於巡迴途中均須考慮執行下一筆任務的執行時間是否超過下班時間，若超過則不可執行該任務，反之則可執行。

3. 節線成本：

由於變數產生法是依據單體法之對偶可行性(Dual Feasibility)來判斷是否找到最佳解，因此令 i,j 節點間的節線成本為 $Cost(i,j) = Time(i,j) + WaitT_j - \pi_j$，其中 $WaitT_j$ 代表執行任務節點 j 的等待時間，若車輛到達節點 j 的時間 $AT_j < ET_j$ 時，即車輛於最早時間之前到達，則車輛必須等待時間為$(ET_j - AT_j)$，意即 $WaitT_j = ET_j - AT_j$；反之，若車輛到達節點 j 的時間 $AT_j > ET_j$ 時，則車輛無須等待即可馬上執行任務 j，意即此時 $WaitT_j = 0$。而 π_j 為主問題中 j 任務所對應的對偶變數值。

接下來說明為何上述的節線成本定義可以滿足於 2.4 節中對偶可行性的驗證方式。在 2.4 節中說明了判別子問題所求出的解是否對於改善主問題之目標值有貢獻的依據，為藉由轉換主問題的對偶問題的第一個限制式來做判斷，即(2-4)式所示 $\sum_{k \in K} a_{kr}\pi_k \leq c_r$，$\forall r \in R$，或是寫成 $c_r - \sum_{k \in K} a_{kr}\pi_k \geq 0$，$\forall r \in R$，若求得子問題的解足(2-4)式的對偶可行性，則得知已找到最佳解，可停止變數產生法的解題流程，反之若求得子問題的解違反(2-4)式，則得知此解所對應的新變數可改善主問題的目標值，因此須將此新變數加入主問題後繼續執行變數產生法的解題流程。

令 $c_r^{'}$ 為路徑 r 的減少成本(reduced cost)，c_r 為路徑 r 的時間成本，為節線旅行時間與等候時間的加總，因此

$$\begin{aligned} c_r^{'} &= c_r - \sum_{k \in K} a_{kr}\pi_k = \sum_{i,j \in r, k \in K} a_{ij}^r * Time(i,j) + \sum_{k \in K} a_{kr} * WaitT_k - \sum_{k \in K} a_{kr}\pi_k \\ &= \sum_{i,j \in r, k \in K} a_{ij}^r * Time(i,j) + \sum_{k \in K} a_{kr} * (WaitT_k - \pi_k) \end{aligned} \quad (4\text{-}1)$$

故於子問題網路中須將節線成本修正成時間成本(旅行時間與等候時間的總和)-對偶成本的形式，如上述節線成本的定義。如此在子問題中所求得的最短路徑 r 的成本 Cost

為所經過的節線成本加總，$Cost = \sum_{i,j \in r} Time(i,j) + \sum_{k \in r} WaitT_k - \sum_{k \in r} \pi_k$ ，其中 $\sum_{i,j \in r} Time(i,j) + \sum_{k \in r} WaitT_k$ 等於(4-1)式中的 c_r ，$\sum_{k \in r} \pi_k$ 則對應到(4-1)式中的 $\sum_{k \in K} a_{kr}\pi_k$ ，因此於子問題中求得的最短路徑成本即代表該路徑的於主問題中的 reduced cost，可分成以下兩個情況討論：

1. 求出的 $Cost \geq 0$ ：代表此條最短路徑的 reduced cost ≥ 0，即 $c_r - \sum_{k \in K} a_{kr}\pi_k \geq 0$ ，滿足(2-4)式的對偶可性，如此則得知找到最佳解，可停止變數產生法的解題流程。

2. 求出的 $Cost < 0$ ：代表此條最短路徑的 reduced cost<0，即 $c_r - \sum_{k \in K} a_{kr}\pi_k < 0$ ，此路徑所代表的變數可改善主問題的目標值，因此須將此最短路徑所對應的新變數加入主問題後繼續執行變數產生法的解題流程。

因此藉由子問題網路中的節線成本定義，來求得的子問題最短路徑成本滿足變數產生法中主問題對偶可行性的判斷基準。

定義完節點、節線與節線成本，以及說明完節線成本定義的合理性後，假設要設計一個含有 8 個節點的提送貨網路，節點資料如表 4.1 所示，其中任務類型可為 W、P 與 D 三種類型，W 代表場站為網路中的起迄點，P 代表提貨任務，D 代表送貨任務。依據上述網路節點、節線的定義，我們可以建構出如圖 4.1 的網路拓撲，其中每個節點以 i-jK 表示，i 代表節點編號，j 代表訂單編號，K 為任務類型，而節線成本因包含等待時間與對偶變數值，需要在演算法執行時才能求得，故在此階段仍無法定義出各條節線的節線成本。

表 4.1 節點資料表

節點編號	訂單編號	任務類型	最早時間	最晚時間
1	0	W	0	0
2	1	P	0	10
3	2	P	2	9
4	3	P	1	11
5	2	D	2	11
6	1	D	1	15
7	3	D	2	12
8	0	W	0	10000

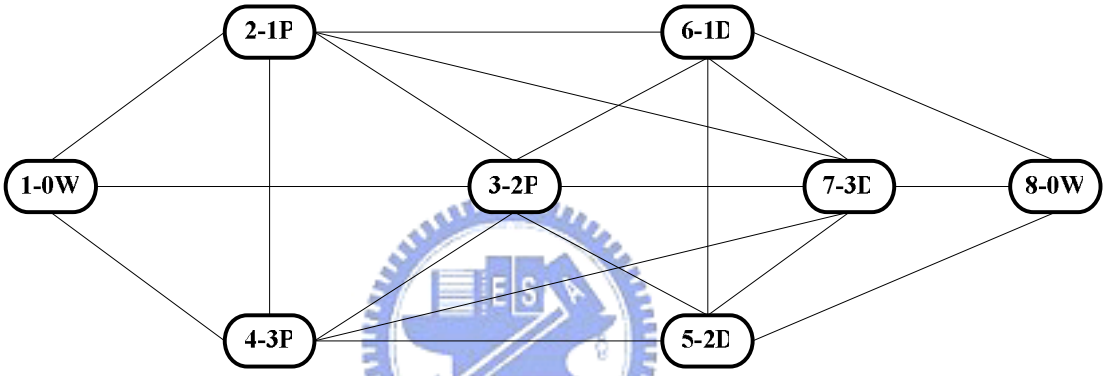

圖 4.1 子問題網路拓撲

藉由上述完成網路中的節點、節線與節線成本的定義後，即可依據各種網路特性建構出各種類型的子問題網路，如大或小規模網路、高或低密度網路等，網路建構完成後即可藉由本研究所提出的修正 Dijkstra's Algorithm 來求解最短路徑，求得的最短路徑依據對偶可行性來決定是否加入主問題中。

4.2 求解子問題

由第三章與 4.1 小節得知，本研究所設計的子問題為包含許多提送貨特性的最短路徑問題，必須考量提送貨任務的允許執行時間窗、提送貨任務配對、提送貨優先次序、司機下班時間與車輛載重容量等因素後，最後並可列出一條完整的最短路徑。通常在求解最短路徑問題所使用的 Dijkstra 演算法並沒有考慮這些提送貨因素，故針對本研究之子問題最短路徑問題而言，無法直接使用 Dijkstra’s 演算法來解決，因此本研究將修正 Dijkstra’s 演算法來求解子問題。

本研究所提出的修正後 Dijkstra's 演算法為考量多種提送貨因素下的最短路徑演算法，因此除了像 Dijkstra's 演算法在每個節點上存取時間成本標籤值外，尚須存取載重材積標籤、上游點標籤與對偶值標籤，成為多重標籤的最短路徑演算法，並在演算法中加入司機下班時間與車輛容量的檢驗機制，以在不違反下班時間與最大載重材積的限制下找出總節線成本最小的路徑。另外本演算法在找尋最短路徑時也須滿足優先限制(Precedence Constraints)與聯結限制(Coupling Constraints)，意即同一訂單的提貨任務須優先於送貨任務執行，且每筆訂單的提貨與送貨任務均由同一輛車執行，以使修正後的演算法能在滿足多種提送貨因素下找一條完整的最短路徑。

本最短路徑演算法是以 Dijkstra's 演算法為基礎作修正，整體演算法的概念流程與 Dijkstra's 演算法相同，但是在更新每個節點的標籤時，需要針對各個節點作下列五種限制的判斷，使得演算法能在滿足各種提送貨的限制因素下找出一條最短路徑：

1. 優先限制：針對同一筆訂單而言，車輛應先執行該訂單的提貨任務後才可執行該訂單的送貨任務。
2. 提送貨合理性限制：若車輛的累積載貨不等於零時，不允許回場站。
3. 時間窗限制：若車輛在某一節點的最早時間之前到達，則必須等候至最早時間才可執行任務；若車輛在某一節點的最早時間與最晚時間中到達，則無須等待即可直接執行任務；若車輛在某一節點的最晚時間後到達，則不允許執行該節點任務。
4. 司機下班時間限制：若車輛於某一節點的執行時間超過司機下班時間，則不允許執行該節點任務。
5. 車輛容量限制：若車輛於某一節點的累積載重超過車輛容量，則不允許執行該節點任務。

將上述五種提送貨限制納入 Dijkstra's 演算法後，演算法流程如圖 4.2 所示，之後本研究稱此部分的演算法為 D1 演算法。茲將流程圖中標示為 1 與 2 的步驟作以下說明：

1. 資料初始化：此步驟為初始化各節點資料，設定司機下班時間與車輛容量兩參數，並建立一空的 S 節點集合。首先設定所有節點的最早時間、最晚時間與載重值，接下來設定起點的等待時間、累積載貨、成本標籤值、執行時間、上游點與對偶值為零，再設定其他任務節點與迄點的等待時間、累積載貨、上游點與對偶值為零，成

本標籤值與執行時間為無窮大，實際執行時則是給定一個很大的數。以下針對這些資料作定義說明：

A. 司機下班時間：以 EndT 代表司機人員的下班時間，定義如 4.1 節所述。

B. 車輛容量：以 C 代表車輛的容量，車輛於巡迴途中均須考慮執行下一筆任務的累積載重值是否超過車輛容量，若超過則不可執行該任務，反之則可執行。

C. S：為網路中已標記的節點集合，網路中的任何節點 i 僅能分成屬於 S 集合或不屬於 S 集合兩類，分別以 $i \in S$ 與 $i \notin S$ 表示之，若 $i \in S$ 代表節點 i 曾為成本標籤值最小之節點並標記之，使成為 S 集合之元素之一。

D. 最早時間：以 ET_i 代表允許執行節點 i 任務的最早時間，定義如 4.1 節所述。

E. 最晚時間：以 LT_i 代表允許執行節點 i 任務的最晚時間，定義如 4.1 節所述。

F. 等待時間：以 $WaitT_i$ 代表於車輛於節點 i 的等待時間，定義如 4.1 節所述。

G. 載重值：以 W_i 代表節點 i 的載重值，若節點 i 為提貨點時 $W_i>0$，反之若節點 i 為送貨點時 $W_i<0$。

H. 累積載貨：以 CW_i 代表節點 i 的累積載重值，假設節點 i 的上游點為節點 u，則節點 i 的累積載重值可由下式求得：$CW_i = CW_u + W_i$。

I. 成本標籤值：以 L(i)代表節點 i 的成本標籤值。本研究所提出的最短路徑演算法即在每個 iteration 中藉由比較各節點的成本標籤值來決定行走的路徑，將成本標籤值最小的節點放入 S 集合中。

J. 上游點：以 Γ-1{i}代表節點 i 的上游點，故若節點 i 的上游點為節點 u，可用 Γ-1{i}=u 表示之。

K. 執行時間：以 $ImpT_i$ 代表節點 i 的可執行時間。在本研究中假設可執行時間即為離開時間，不考慮車輛於節點的服務時間，故假設 $\Gamma^{-1}\{i\}=u$，則節點 i 的可執行時間可由下式求得：

$ImpT_i = ImpT_u + Time(u,i) + WaitT_i$。

L. 對偶值：以 π_i 代表節點 i 的對偶值，此值須藉由單體法求解主問題所得。

2. 更新節點 v 資料：更新節點 v 的等待時間、累積載貨、上游點、成本標籤值與執行時間等資料。

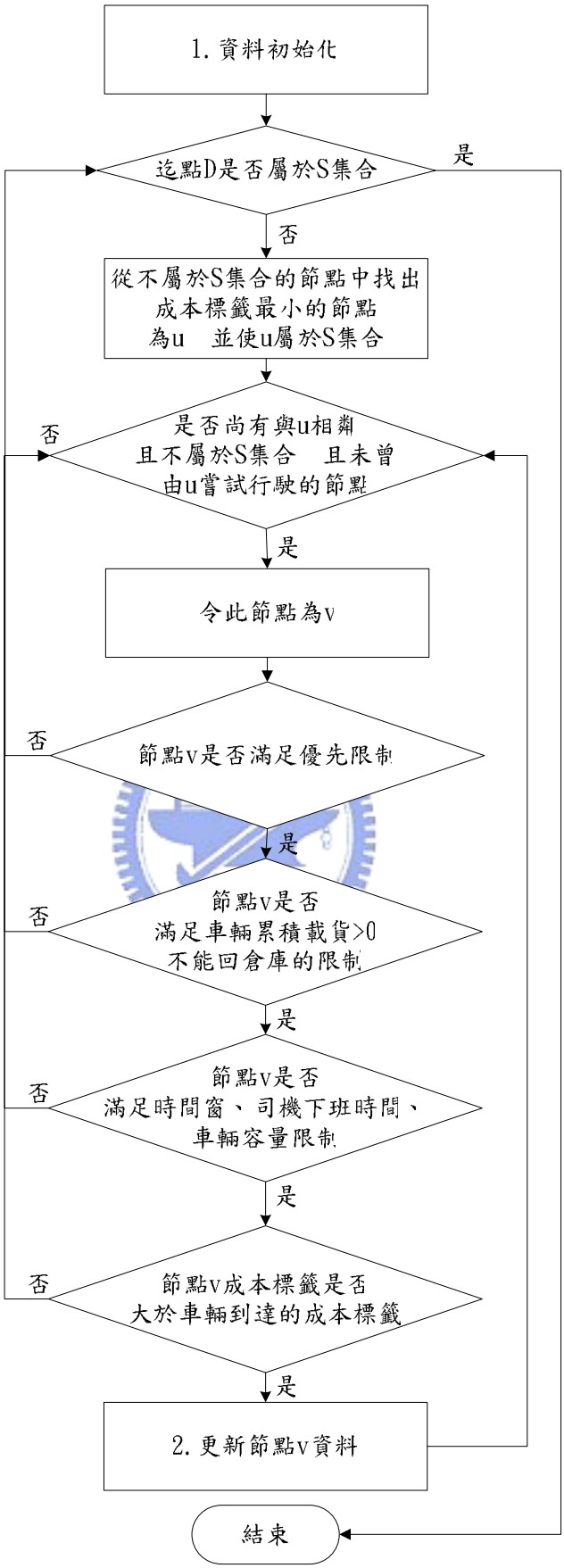

圖 4.2 D1 演算法流程圖

依據上述五種提送貨限制判斷來找尋最短路徑的演算法流程中，其解題步驟與精神主軸仍承襲了傳統 Dijkstra's 演算法，在每個 iteration 中於未標籤過的節點集合中找尋成本標籤值最小的節點，而後標籤(Label Setting)該節點，因此最終即使迄點沒有上游點且成本標籤值很大，但在其他節點均已經標籤過的情況下，仍是會將迄點標籤住而後結束，如此將無法產生一條從起點至迄點的可行路徑。至於導致迄點沒有上游點的因素多是因為有了提送貨合理性的限制，意即當車輛的累積載貨不等於零時，不允許回迄點(場站)。也就是在最後產生的路徑可能會有提送貨節點無法配對的情況發生，意即車輛巡迴路徑中執行了某筆訂單的提貨任務，但沒有執行該筆訂單的送貨任務。

以下以圖 4.3 的網路來說明迄點可能沒有上游點的情形，圖的表示方法與圖 4.1 的網路拓撲定義相同，其中節點(1-0w)為起點，節點(8-0w)為迄點。假設目前演算法執行到一半所找到的路徑為(1-0w)→(2-1P)→(4-3P)→(3-2P)，意即該貨車已經執行了第一筆、第二筆與第三筆訂單的提貨任務，因此此時節點(1-0w)、(2-1P)、(4-3P)與(3-2P)均已被標籤住，意即(1-0w)、(2-1P)、(4-3P)與(3-2P) ∈S，如圖 4.4 所示，其中灰色的節點代表已被標籤住，箭頭代表貨車的行駛路徑。

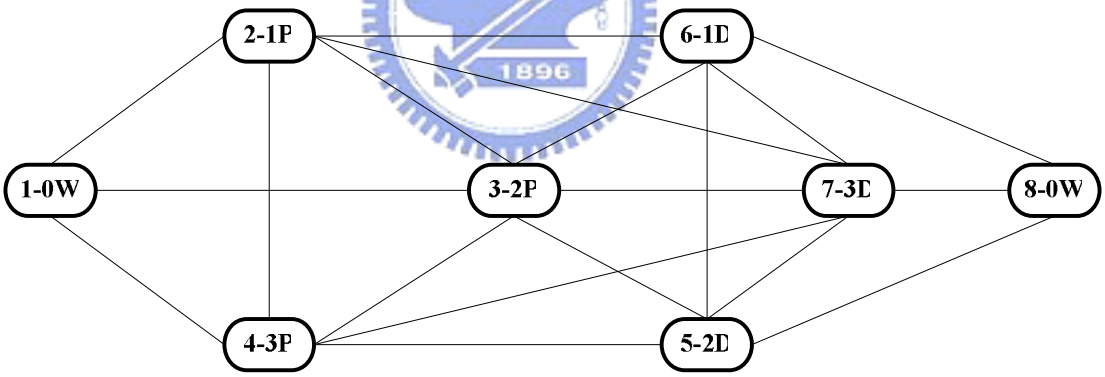

圖 4.3 迄點沒有上游點之範例網路拓撲

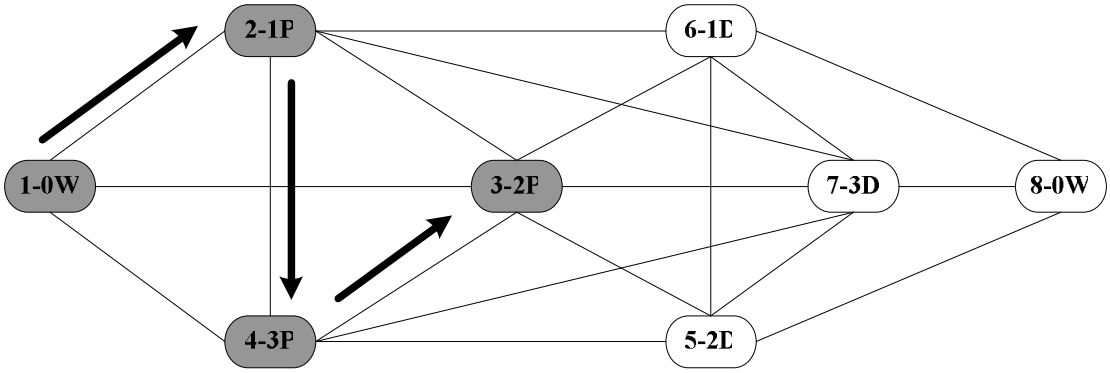

圖 4.4 迄點沒有上游點之範例網路說明圖一

接下來如圖 4.5 所示，假設走節點(6-1D)為最短，因此標籤住節點(6-1D)，意即(6-1D)∈S，此時與節點(6-1D)相鄰且且不屬於 S 集合的節點為(5-2D)、(7-3D)與迄點(8-0w)，但貨車卻不允許直接從(6-1D)回到迄點(8-0w)，因為此時貨車僅執行完第一筆訂單的提貨與送貨任務，車上仍有第二筆與第三筆訂單的貨品，意即車輛此時的累積載貨不為零，在尚未執行完第二筆與第三筆訂單的送貨任務之前，貨車並不允許回到迄點。又假設此時貨車從節點(6-1D)行駛至(5-2D)與(7-3D)時，均不滿足更新(5-2D)與(7-3D)的成本標籤限制，故在這次 iteration 中僅標籤住節點(6-1D)，並更新節點(6-1D)的資料與設定上游點為(3-2P)。

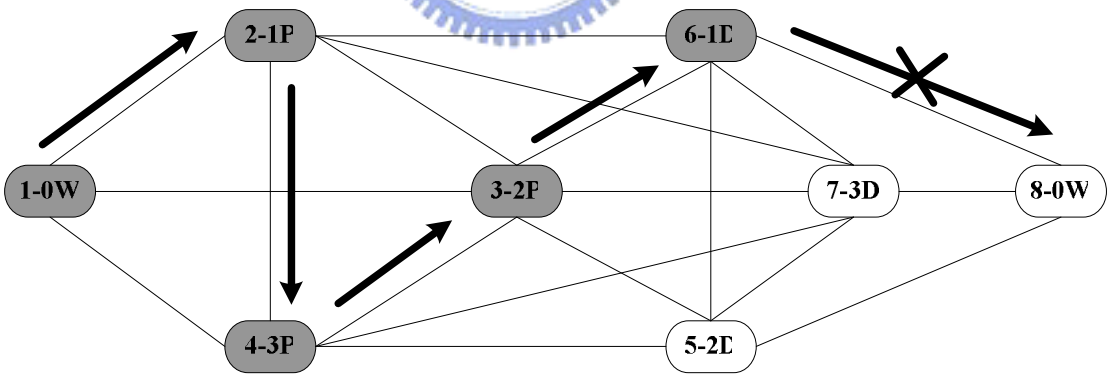

圖 4.5 迄點沒有上游點之範例網路說明圖二

接下來如圖 4.6 所示，假設走節點(7-3D)為最短，因此標籤住節點(7-3D)，意即(7-3D)∈S，此時與節點(7-3D)相鄰且不屬於 S 集合的節點為(5-2D)與迄點(8-0w)，但貨車仍不允許直接從(7-3D)回到迄點(8-0w)，因為此時貨車僅執行完第三筆訂單的提貨與送貨任務，車上仍有第一筆與第二筆訂單的貨品。又假設此時貨車從節點(7-3D)行駛至(5-2D)時，不滿足更新(5-2D)的成本標籤限制，故在這次 iteration 中僅標籤住節點(7-3D)，

並更新節點(7-3D)的資料與設定上游點為(3-2P)。

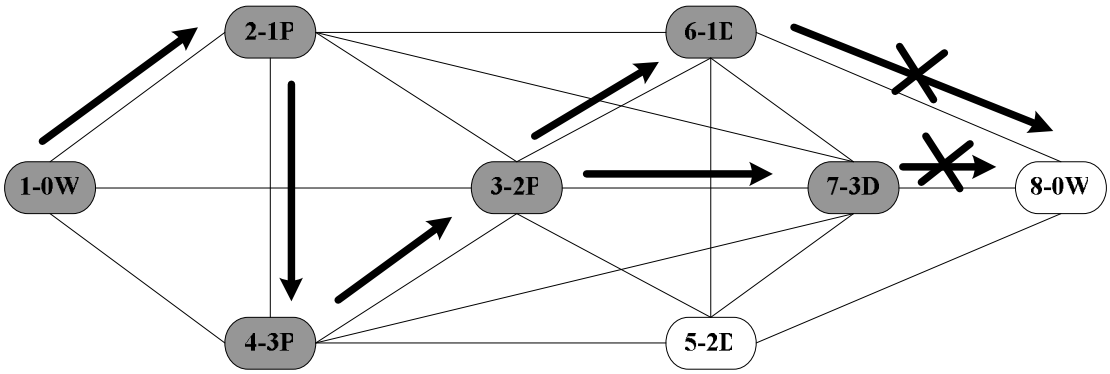

圖 4.6 迄點沒有上游點之範例網路說明圖三

接下來如圖 4.7 所示，假設接下來走節點(5-2D)為最短，因此標籤住節點(5-2D)，意即(5-2D)∈S，此時與節點(5-2D)相鄰且不屬於 S 集合的節點只有迄點(8-0w)，但貨車仍不能直接從(5-2D)回到迄點(8-0w)，因為此時貨車僅執行完第二筆訂單的提貨與送貨任務，車上仍有第一筆與第三筆訂單的貨品，意即車輛此時的累積載貨不為零，在尚未執行完第一筆與第三筆訂單的送貨任務之前，貨車並不允許回到迄點。故在這次 iteration 中僅標籤住節點(5-2D)，並更新節點(5-2D)的資料與設定上游點為(3-2P)。

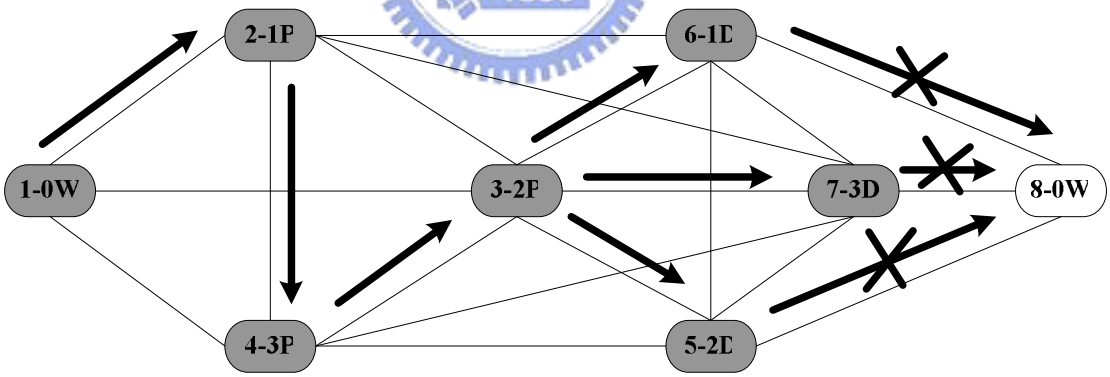

圖 4.7 迄點沒有上游點之範例網路說明圖四

最後如圖 4.8 所示，整個網路上除了迄點(8-0w)以外其他節點都已經被標籤住的情況下，在最後一個 iteration 中還是會把迄點給標籤住，而後結束 D1 演算法，但此時迄點(8-0w)卻沒有上游點，故無法得到一條從起點(1-0w)至迄點(8-0w)的完整路徑。

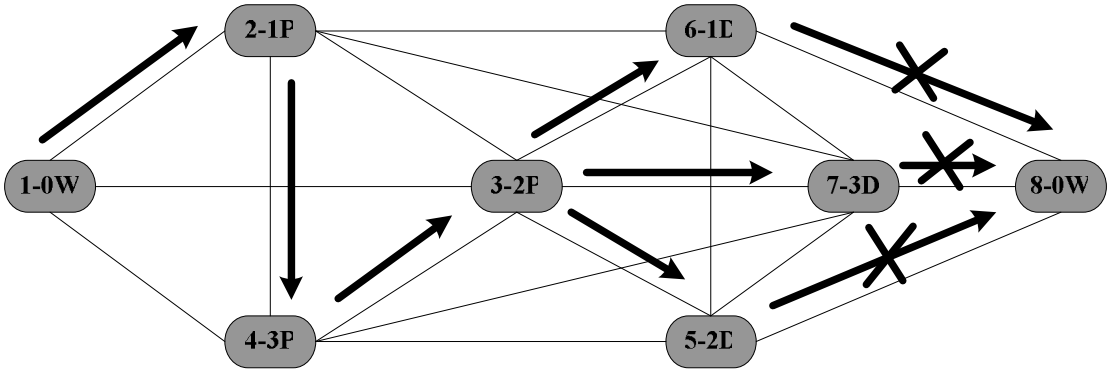

圖 4.8 迄點沒有上游點之範例網路說明圖五

為了解決上述問題，本研究另外設計以下四個步驟的演算法，在迄點已被標籤住但卻沒有上游點的情況下使用，企圖在滿足上述的各類提送貨特性下建構出一條合乎提送貨配對且路徑總成本小於零的路徑：

1. 依據目前不完整的路徑來找出路徑中應該執行但沒有執行的任務節點，舉例來說若目前路徑有經過第一筆訂單的提貨節點，但卻沒有經過第一筆訂單的送貨節點，此時就要將第一筆訂單的送貨節點強制插入路徑中，以完成完整的提送貨路徑。假設共有 m 個任務節點應執行而未執行，而後窮舉這些任務節點的排列次序可能，如此總共會有 m!種可能，令 n=m!。

2. 依照第一步驟的次序將這些任務節點強迫插入路徑中，插入的法則為在滿足優先限制(Precedence Constraints)下採用最小成本插入法，意即在滿足先提貨後送貨的原則下，將這些應執行而為執行的節點插入至使成本增加最少的位置，插入後將可得到一條從起點至迄點的完整路徑，若此整路徑的路徑成本小於零，則將此路徑所對應的變數加入主問題中繼續求解；反之若插入節點後的完整路徑的成本大於或等於零，則開始執行第三步驟的修補演算法。

3. 將所有的訂單(提送貨配對)依據對偶變數值由小到大排序，按照排序結果移除第二步驟獲得的路徑中的提送貨配對。由移除一個提送貨配對開始，若移除提送貨配對後的路徑成本小於零，則將此路徑所對應的變數加入主問題中繼續求解；反之則繼續嘗試移除下一個提送貨配對。若嘗試所有移除一個提送貨配對的可能後均沒有找到成本小於零的路徑時，則開始移除兩個提送貨配對，規則步驟與移除一個提送貨配對相同，也就是若移除兩個提送貨配對後的路徑成本小於零，則將此路徑所對應

的變數加入主問題中繼續求解；反之則繼續嘗試下一種移除兩個提送貨配對的可能，如此週而復始的循環直到移除 m 個提送貨配對。若嘗試所有移除 m 個提送貨配對的可能後均沒有找到成本小於零的路徑時，則開始執行第四步驟的修補演算法。

4. 依照第三步驟的對偶值排序結果，由小到大標籤住第二步驟獲得的路徑中「已強迫插入」任務節點的提送貨配對，而後重新執行 D1 演算法。由標籤住一個提送貨配對開始，若標籤提送貨配對後執行 D1 演算法所求得的路徑成本小於零，則將此路徑所對應的變數加入主問題中繼續求解；反之則繼續嘗試標籤住下一個提送貨配對。若嘗試所有標籤住一個提送貨配對的可能後均沒有找到成本小於零的路徑時，則開始標籤兩個提送貨配對，規則步驟與標籤一個提送貨配對相同，也是若標籤兩個提送貨配對後執行 D1 演算法所求得的路徑成本小於零，則將此路徑所對應的變數加入主問題中繼續求解；反之則繼續嘗試下一種標籤兩個提送貨配對的可能，如此週而復始的循環直到標籤 m 個提送貨配對。若嘗試所有標籤 m 個提送貨配對的可能後均沒有找到成本小於零的路徑時，則於第一步驟獲得的 n 種插入節點排列次序可能中選擇下一種插入節點的次序，繼續依序執行步驟二、三、四，直到嘗試完所有插入任務節點的次序時則結束整個修補方案。

本研究以圖 4.3 迄點沒有上游點的情況為例，執行上述的四個步驟演算法：

1. 假設目前不完整的路徑為(1-0w)→(2-1P)→(4-3P)→(3-2P)→(5-2D)，因此應執行但沒有執行的任務節點共有 2 個，分別為(6-1D)與(7-3D)，如此總共會有 2 種排列次序可能，即(6-1D)→(7-3D)與(7-3D)→(6-1D)，此時 m=2，n=m!=2!=2。

2. 首先嘗試以第一種排列次序(6-1D)→(7-3D)強迫插入路徑中，假設插入後得到的路徑為：(1-0w)→(2-1P)→(4-3P)→(3-2P)→(5-2D)→(6-1D)→(7-3D)→(8-0w)，如圖 4.9 所示，若此完整路徑的路徑成本小於零，則將此路徑所對應的變數加入主問題中繼續求解；反之若此完整路徑的成本大於或等於零，則開始執行第三步驟的修補演算法。

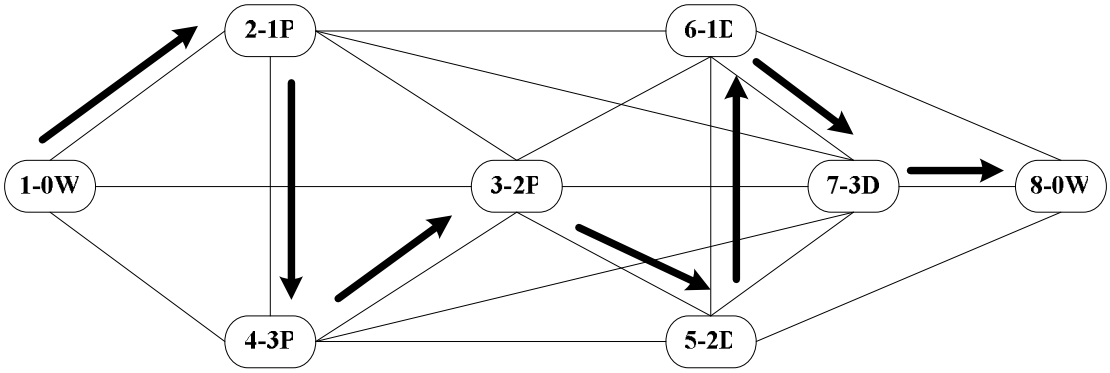

圖 4.9 解決迄點沒有上游點之範例網路說明圖一

3. 將所有的訂單(提送貨配對)依據對偶變數值由小到大排序，若排序的結果為1<2<3，意即第一筆訂單的對偶值小於第二筆訂單的對偶值，第二筆訂單的對偶值小於第三筆訂單的對偶值，按照排序結果移除第二步驟獲得的路徑中的提送貨配對。

首先由一次移除一筆提送貨配對開始。因為第一筆訂單的對偶值最小，因此先移除第一筆訂單的提送貨配對，意即移除節點(2-1P)與節點(6-1D)，使路徑成為：(1-0w)→(4-3P)→(3-2P)→(5-2D)→(7-3D)→(8-0w)，如圖 4.10 所示，若此路徑成本小於零，則將此路徑所對應的變數加入主問題中繼續求解；反之則繼續嘗試移除第二筆訂單的提送貨配對，意即移除節點(3-2P)與節點(5-2D)，使路徑為：(1-0w)→(2-1P)→(4-3P)→(6-1D)→(7-3D)→(8-0w) ，然而在網路中節點(4-3P)與節點(6-1D)並無節線相連，因此將此種移除方式並不合理，故將不納入考慮。接下來再嘗試移除節點(4-3P)與節點(7-3D)，若嘗試所有移除一筆提送貨配對的可能後均沒有找到成本小於零的路徑時，則開始移除兩筆提送貨配對。

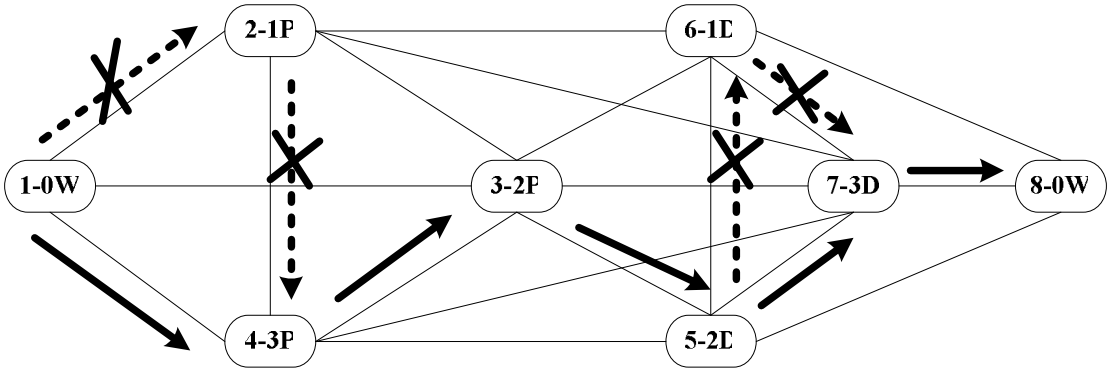

圖 4.10 解決迄點沒有上游點之範例網路說明圖二

以目前有 3 筆訂單的網路為例，依據對偶變數值由小到大排序一次移除兩筆訂單的組合有(1,2)、(1,3)與(2,3)共三種可能，因此首先移除第一筆與第二筆訂單的提送貨配對，使路徑成為：(1-0w)→(4-3P)→(7-3D)→(8-0w)，若此路徑成本小於零，則將此路徑所對應的變數加入主問題中繼續求解；反之則繼續嘗試下一種移除兩個提送貨配對的可能，因為在自此例中「已強迫插入」的任務節點只有兩個，故 m=2，因此若嘗試所有移除兩個提送貨配對的可能後均沒有找到成本小於零的路徑時，則開始執行第四步驟的修補演算法。

4. 依照第三步驟的對偶值排序結果，即 1<2<3，由小到大標籤住第二步驟獲得的路徑中「已強迫插入」任務節點的提送貨配對，在此例強迫插入的提送貨配對為第一筆與第三筆訂單的提送貨配對，意即節點(2-1P)、(6-1D)、(4-3P)與(7-3D)，而後重新執行 D1 演算法。

首先由標籤住一個提送貨配對開始，因為第一筆訂單的對偶值小於與第三筆訂單的提送貨配對，因此先標籤住第一筆訂單的提送貨配對，意即標籤住節點(2-1P)與(6-1D)，如圖 4.11 所示節點(2-1P)與(6-1D)均以灰色表示，而後執行 D1 演算法，如此找出的路徑將不會包含節點(2-1P)與(6-1D)，意即從網路中先踢除節點(2-1P)與(6-1D)後再重新找最短路徑，若所求得的路徑成本小於零，則將此路徑所對應的變數加入主問題中繼續求解；反之則開始嘗試標籤住第三筆訂單的提送貨配對，意即標籤住節點(4-3P)與(7-3D)，而後執行 D1 演算法，若所求得的路徑成本小於零，則將此路徑所對應的變數加入主問題中繼續求解；若嘗試所有標籤一筆提送貨配對的可能後均沒有找到成本小於零的路徑時，則開始標籤兩筆提送貨配對。

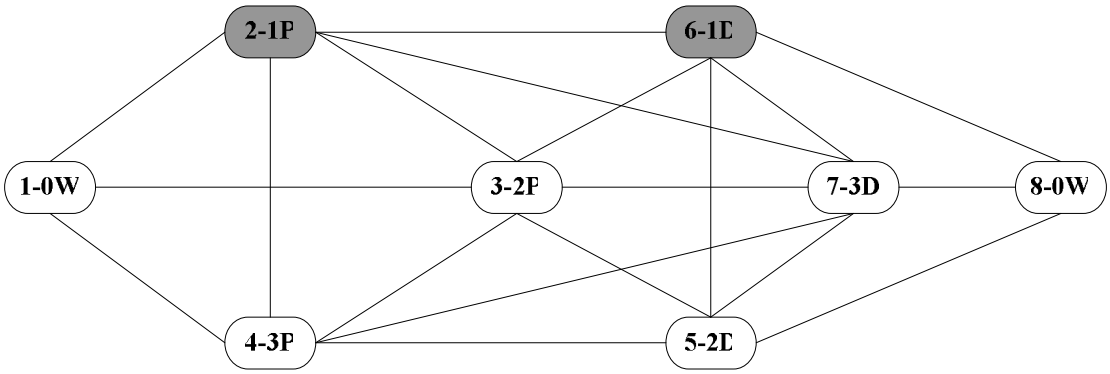

圖 4.11 解決迄點沒有上游點之範例網路說明圖三

在此例中「已強迫插入」任務節點的提送貨配對為第一筆與第三筆訂單的提送貨配對，因此一次標籤住兩筆提送貨配對就是標籤住第一筆與第三筆訂單的提送貨配對，意即標籤住節點(2-1P)、(6-1D)、(4-3P)與(7-3D)，而後執行 D1 演算法，若所求得的路徑成本小於零，則將此路徑所對應的變數加入主問題中繼續求解；反之若成本大於或等於零，則因為在自此例中「已強迫插入」的任務節點只有兩個，故 m=2，因此若嘗試所有標籤兩個提送貨配對的可能後均沒有找到成本小於零的路徑時，則回到第一步驟中選擇(7-3D)→(6-1D)的插入次序可能，繼續依序執行步驟二、三、四。

因此將上述五種提送貨限制與為了解決迄點沒有上游點的四步驟演算法，一併加入傳統的 Dijkstra's 演算法中，以完成整個修正後的 Dijkstra's 演算法，藉以求解子問題，其概念流程如圖 4.12 所示：

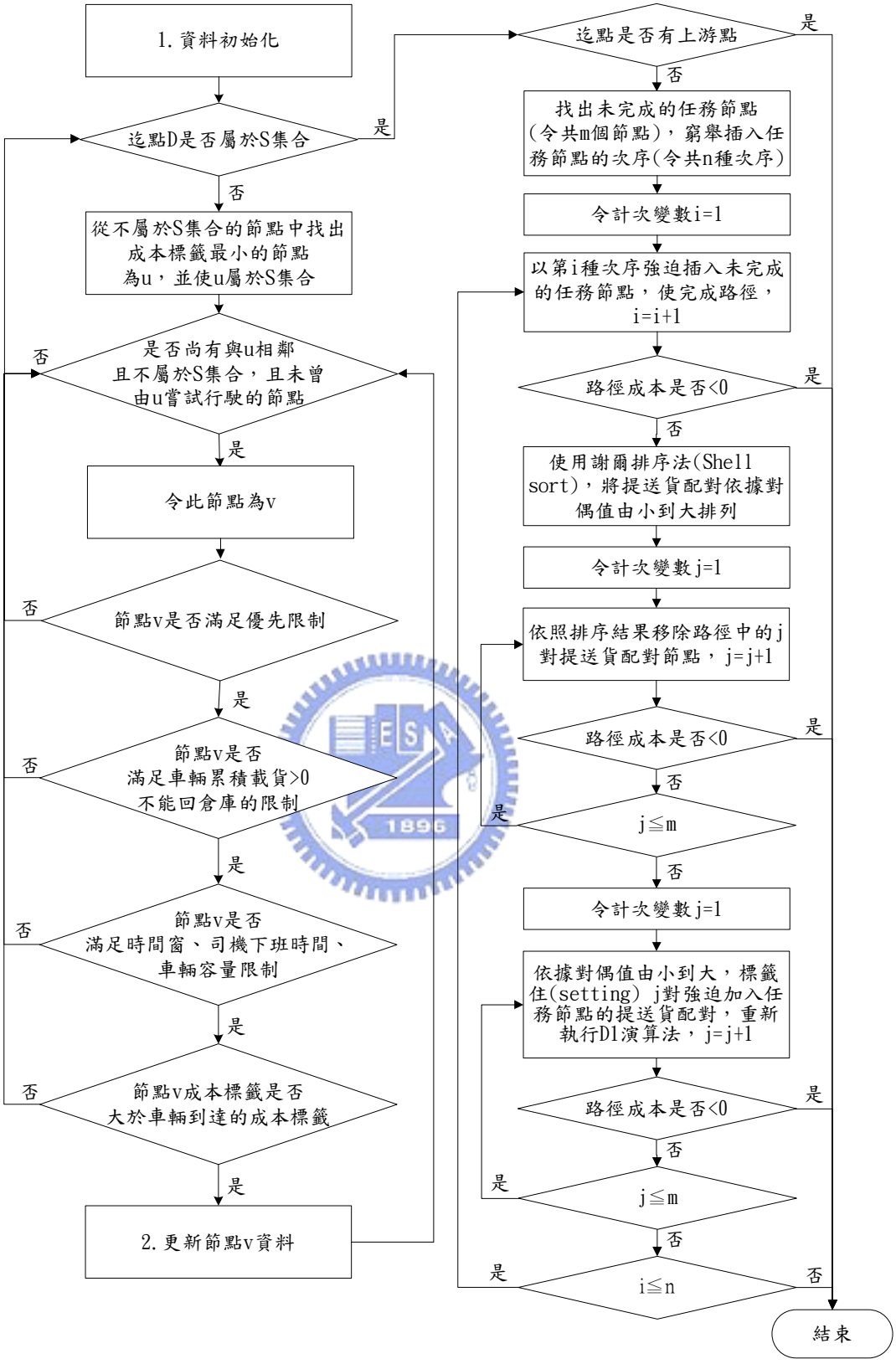

圖 4.12 修正後的 Dijkstra's 演算法概念流程圖

4.3 範例

本研究設計一含有 8 個節點的小型提送貨網路，並藉由上述的最短路徑演算法來求解，網路拓撲如圖 4.13 所示，其中節點的表示方法與圖 4.1、4.3 的定義相同，另外在每條節線上的數字代表改節線的旅行時間，而各節點的初始資料如表 4.2 所示，其中各欄位定義如 4.1 與 4.2 節所述：

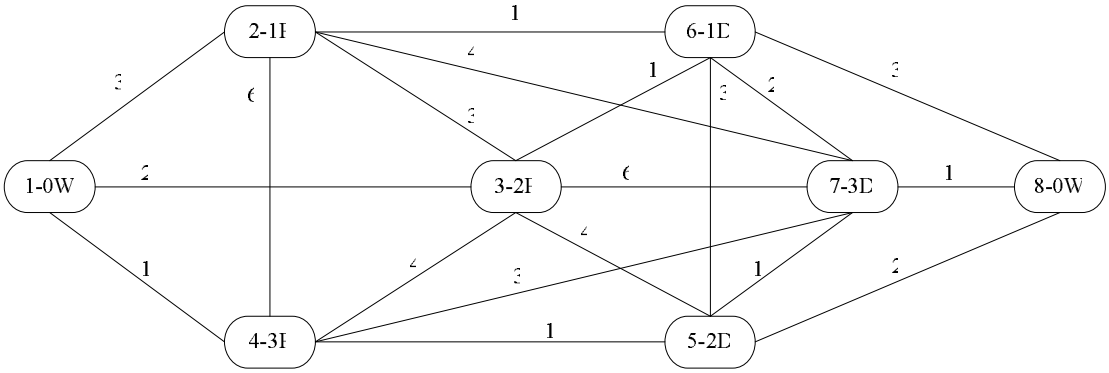

圖 4.13 網路拓撲

表 4.2 節點初始資料表

訂單編號	節點編號	任務類型	最早時間	最晚時間	成本標籤值	執行時間	載重	對偶值	屬於 S 集合
0	1	W	0	0	0	0	0	0	0
1	2	P	0	10	10000	10000	5	7	0
2	3	P	2	9	10000	10000	3	0	0
3	4	P	1	11	10000	10000	2	5	0
2	5	D	2	11	10000	10000	-3	8	0
1	6	D	1	15	10000	10000	-5	0	0
3	7	D	2	12	10000	10000	-2	0	0
0	8	W	0	10000	10000	10000	0	0	0

接下來設定司機下班時間為 15，車輛容量為 10，便可開始執行修正後的 Dijkstra's 最短路徑演算法。以下將示範本演算法的第一個 iteration 計算過程，其中各符號的定義如 4.1 與 4.2 節所述。計算過程如下所示：

1. 判斷 D 不屬於 S

2. 於所有不屬於 S 集合的節點中找尋成本標籤值最小的節點，並令該節點為 u 且屬於 S 集合，亦即：

   $\min\{L(v), \text{where } v \notin S\}=\min\{L(1),L(2),L(3),L(4),L(5),L(6),L(7),L(8)\}=0$，令 $u=1$，$u \in S$。

3. 判斷是否有與 1 相鄰的節點：找到節點 1 有相鄰節點 2。

4. 判斷節點 2 是否滿足優先限制：因節點 2 為第一筆訂單的提貨節點，故滿足優先限制。

5. 判斷節點 2 的提送貨合理限制：因節點 2 為提貨節點，故滿足提送貨合理性限制。

6. 計算節點 2 的等待時間：因 $ImpT_1 + Time(1,2)=0+3=3>ET_2=0$，故 $WaitT_2 =0$。

7. 判斷節點 2 是否滿足時間窗、司機下班時間、車輛容量限制：

   A. 因 $ImpT_1 + Time(1,2)=0+3=3 \leqq LT_2=10$，故節點 2 滿足時間窗限制

   B. 因 $ImpT_1+Time(1,2)+WaitT_2=0+3+0=0 \leqq EndT=15$，故節點 2 滿足司機下班時間限制

   C. 因 $CW_1+ W_2=0+5=5 \leqq C=10$，故節點 2 滿足車輛容量限制。

8. 判斷節點 2 的成本標籤是否可更新：

   因 $L(1)+Time(1,2)+WaitT_2-\pi_2=0+3+0-7= -4<L(1)=1000$，故可更新節點 2 資料。

9. 更新節點 2 的資料：

   A. 更新節點 2 的成本標籤：$L(1)= L(0)+Time(1,2)+ WaitT_2-\pi_2=0+3+0-7= -4$

   B. 更新節點 2 的執行時間：$ImpT_2=ImpT_1+Time(1,2)+ WaitT_2=0+3+0=3$

   C. 更新節點 2 的累積載重：$CW_2= CW_1+ W_2=0+5=5$

D. 更新節點 2 的上游點編號：$\Gamma^{-1}\{2\}=1$

10. 判斷是否有與 1 相鄰的節點：找到節點 1 有相鄰節點 3 後，執行 4~9 步驟。

11. 判斷是否有與 1 相鄰的節點：找到節點 1 的相鄰節點 4 後，執行 4~9 步驟。

12. 判斷是否有與 1 相鄰的節點：節點 1 沒有其他相鄰節點，結束修正後 Dijkstra's 最短路徑演算法的第一個 iteration。

將修正後 Dijkstra's 最短路徑演算法的第一個 iteration 的計算結果整理成表 4.3 節點資料，如下所示：

表 4.3 1 個 iteration 後的節點資料

節點編號	上游點編號	等待時間	最早時間	最晚時間	成本標籤值	執行時間	累積載重	屬於 S 集合
1	0	0	0	0	0	0	0	1
2	1	0	0	10	-4	3	5	0
3	1	0	2	9	2	2	3	0
4	1	0	1	11	-4	1	2	0
5	0	0	2	11	10000	10000	0	0
6	0	0	1	15	10000	10000	0	0
7	0	0	2	12	10000	10000	0	0
8	0	0	0	10000	10000	10000	0	0

完成整個修正後 Dijkstra's 演算法的流程後，可獲得一條最短路徑：1->2->6->3->5->8，路徑成本為-4，節點資料如表 4.4 所示。因為目前求出的路徑成本小於零，由 2.4 節的對偶可行性的驗證方式可知此路徑對於主問題目標式有貢獻，故應將對應此路徑的變數(Column)加入主問題中繼續整個變數產生法的解題流程。

表 4.4 完成修正後 Dijkstra's 演算法的節點資料

節點編號	上游點編號	等待時間	最早時間	最晚時間	成本標籤值	執行時間	累積載重	屬於 S 集合
1	0	0	0	0	0	0	0	1
2	1	0	0	10	-4	3	5	1
3	6	0	2	9	2	5	3	1
4	1	0	1	11	-4	1	2	1
5	3	0	2	11	-6	9	0	1
6	2	0	1	15	-3	4	0	1
7	4	0	2	12	-1	4	0	1
8	5	0	0	10000	-4	11	0	1

透過修正後的 Dijkstra's 最短路徑演算法，即可在建構好的網路中求得最短路徑，這條路徑在主問題中即代表一變數(Column)，之後利用對偶可行性來判斷是否將此變數加入主問題中繼續求解，若最短路徑於迄點的成本標籤大於或等於零，則得知已達最佳解；反之若小於零，則代表此路徑對於主問題目標式有貢獻，將對應此路徑的變數加入主問題。如此將可完成變數產生法之整體求解流程。

第五章 範例測試結果

由於具時間窗的提送貨問題所要考慮的可行變數數量過於龐大，因此本研究嘗試以變數產生法結合分支定限法來求解具時間窗的提送貨問題，其中將變數產生法的主問題建構為放鬆整數限制的集合分割問題，子問題則建構成一最短路徑問題，並設計修正後的 Dijkstra's 演算法來求解子問題。為了測試藉由變數產生法求解具時間窗的提送貨問題的效率，避免耗費大量時間窮舉路徑以解決問題的缺失，本研究將自行設計四種包含不同節點數的網路進行測試，詳細的測試過程於 5.2 節說明，最後於 5.3 節展示測試的結果。

5.1 產生測試範例

本研究依據 4.1 節針對網路節點、節線與節線成本的定義，個別建構四個具有不同節點數量的網路，分別為包含 8、14、50 與 100 個節點的網路，每個網路均包含一個起點與一個迄點，其餘的節點均為任務節點，其中一半為提貨節點，另一半則為送貨節點。因此在包含 8 個節點的網路中，是由 3 筆訂單所產生 3 個提貨節點與 3 個送貨節點，再加上起點與迄點所組成，包含 14 個節點的網路則是由 6 筆訂單產生 6 個提貨節點與 6 個送貨節點，再加上起點與迄點所組成，而包含 50 與 100 個節點的網路也是依此方式所組成。最後設定每個節點的允許執行時間窗，意即最早時間與最晚時間，此部份由本研究針對各個節點設定一對合理的時間窗。

5.2 測試過程

本研究的測試過程是藉由 AMD Athlon(tm) XP 3200+, 1.00GB RAM 的個人電腦來運作，作業系統為 Microsoft Windows XP Professional SP2，使用 Microsoft Visual C++ 6.0 編輯器撰寫修正後的 Dijkstra’s 演算法，藉由 CPLEX 7.0 [6][7] 中的 Concert Technology 數學工具來求解變數產生法中主問題線性規劃與整數規劃問題。

於 5.1 節建構完成網路拓墣後，則開始建構變數產生法中的主問題，首先產生初始變數，變數的範圍此時設為大於或等於零，故為一線性規劃問題。在本研究中初始變數

為代表一條僅服務一筆訂單的巡迴路徑，意即路徑為起點→某筆訂單的提貨點→某筆訂單的送貨點→迄點，均僅經過四個節點。針對 8 個節點之網路而言，因包含 3 筆訂單故建立分別對應於這 3 筆訂單的 3 個初始變數，而後納入主問題中，並依序列出這 3 筆訂單所產生的 6 個任務節點的對應限制式，而後利用 Concert Technology 數學工具，以單體法(Simplex Method)來求得對應於限制式的對偶變數值 π_k=，k=1~6，再將這些對偶變數值傳遞到網路中所相對應的任務節點上。

接下來使用 C++程式語言撰寫修正後的 Dijkstra’s 演算法來求解子問題，若求得的路徑成本<0，則新增一個對應於該路徑的變數，並將此新變數加入主問題中同時修正目標式與限制式，同樣的設定這個新變數的範圍為大於或等於零，之後再使用 Concert Technology 數學工具重新求解主問題，以獲得新的對偶變數值，再將求得的對偶變數值傳入子問題中所相對應的任務節點上，重新使用修正後的 Dijkstra’s 演算法來求解，直到修正後的 Dijkstra’s 演算法求解出的路徑成本>0 時終止整個變數產生法的解題流程。

之後利用 Concert Technology 數學工具將目前主問題中所有變數的形態轉為二元整數型態，使的主問題成為一整數規劃問題，再重新求解主問題以獲得整數解。對於 14、50 與 100 個節點的網路也是依據上述的過程作範例測試。

5.3 測試結果

將四個分別包含 8、14、50 與 100 個節點的網路依照 5.2 節所說明的測試過程作測試，得到求解結果如表 5.1 所示。在是否使用「解決迄點沒上游點」之演算法方面，由表可知範例一為僅包含 8 個節點的小型網路，執行 D1 演算法後，並未發生迄點沒有上游點的情形，因此不需要執行「解決迄點沒上游點」的四步驟演算法；然而當網路規模逐漸擴大時，如範例二、三與四，執行 D1 演算法後，會發生迄點沒有上游點的情形，因此需要執行「解決迄點沒上游點」的四步驟演算法。

就求解效率而言，利用變數產生法來求解具時間窗的提送貨問題可以快速的求解，而且對於規模越大的網路快速求解的效果越明顯。以範例二為例，範例二包含 14 個節點，是由 6 筆訂單產生 6 個提貨節點與 6 個送貨節點，再加上起點與迄點所組成，若使

用窮舉法將要列舉 $C_1^6 * 2! + C_2^6 * 4! + C_3^6 * 6! + C_4^6 * 8! + C_5^6 * 10! + C_6^6 * 12! = 501{,}393{,}972$ 條路徑才可求解，而本研究利用變數產生法求解，除了原先的初始變數外，僅須再找到 20 條成本為負的路徑，將其所對應的變數(Column)加入主問題中，即可求解出總成本為 24 的可行路徑組合；範例三除了原先的初始變數外，僅須再找到 149 條成本為負的路徑，將其所對應的變數(Column)加入主問題中，即可求解出總成本為 89 的可行路徑組合；範例四除了原先的初始變數外，僅須再找到 401 條成本為負的路徑，將其所對應的變數(Column)加入主問題中，即可求解出總成本為 177 的可行路徑組合。

表 5.1 測試結果

	範例一	範例二	範例三	範例四
節點數	8	14	50	100
包含訂單筆數	3	6	24	49
是否使用「解決迄點沒上游點」之演算法	否	是	是	是
本演算法所找到的變數(Column)數量	1	20	149	401
本演算法之求解目標值	16	24	89	177

第六章 結論與建議

6.1 結論

本研究設計一套以最佳解為基礎的演算法來求解具時間窗的提送貨問題，整個演算法是由變數產生法與分支定限法所組成。在變數產生法中將主問題建構為放鬆整數限制的集合分割問題，故視為一線性規劃問題，而子問題則建構為一最短路徑問題，由於此最短路徑問題之網路具有許多提送貨的特性，如允許執行時間窗、提送貨任務配對、提送貨優先次序、司機下班時間與車輛載重容量等特性，且最終須能夠列出一條完整的路徑，故無法直接使用傳統的 Dijkstra's 最短路徑演算法來求解本研究的子問題，因此本研究以傳統的 Dijkstra's 演算法做修正，成為一滿足上述的提送貨特性的修正後 Dijkstra's 演算法來求解子問題。

由測試結果可知變數產生法非常適用於求解變數數量多(規模大)的問題，而本研究所探討的具時間窗提送貨問題就是屬於這類問題，且問題的規模(變數的數量)會隨著網路擴大而呈爆炸性成長，若使用土法煉鋼的窮舉方式，光是 14 個節點的網路就需要列舉五億多條路徑方可求解，如此將容易失去求解問題的時效性，不符貨運物流產業中分秒必爭的特性，因此使用本研究所提出的演算法來求解具時間窗提送貨問題，能在合理的運算時間下求出不錯的路徑組合，有效提升貨車調度人員的派遣作業與降低貨物運送的成本。

6.2 建議

1. 本研究所設計的修正後 Dijkstra's 演算法，雖然是以傳統的 Dijkstra's 演算法為基礎作修正的，然而因為修正後的演算法同時考慮多種提送貨特性，導致求解子問題所產生的路徑可能會有迄點沒有上游點的問題，如此將無法列出一條完整的路徑，故於 4.2 節另外設計四個步驟來解決迄點沒有上游點的問題，如此將導致求解子問題的演算法分成兩階段來執行，建議未來可將修正後的 Dijkstra's 演算法設計成一階段的演算法，能在避免迄點沒有上游點的問題前提下找出完整的最短路徑。

2. 本研究所設計的修正後 Dijkstra's 演算法，因為考慮多種提送貨特性，導致失去了傳統 Dijkstra's 演算法所具有的求解最佳性，目前修正後 Dijkstra's 演算法無法於子問題中找出最短路徑，建議持續修正演算法以提升修正後 Dijkstra's 演算法的求解能力。

3. 由於本研究將子問題設計成一考慮多種提送貨特性的最短路徑問題，而後嘗試修改 Dijkstra's 演算法，以期望使用修正後的 Dijkstra's 演算法來求解子問題，然而在找尋最短路徑時需要滿足提送貨問題的優先限制(Precedence Constraints)，意即針對每筆訂單而言，要先執行提貨任務才能執行送貨任務，以及限定當貨車的累積載重為零時不允許回場站，這些限制導致修正後的 Dijkstra's 演算法無法找到最短路徑，因此建議後續的研究不以 Dijkstra's 演算法為基礎來發展考慮提送貨特性的最短路徑問題的求解方法。

4. 本研究的修正後 Dijkstra's 演算法是以傳統 Dijkstra's 演算法為基礎作修正的，因此解題的精神仍承襲了傳統的 Dijkstra's 演算法，意即在傳統的 Dijkstra's 演算法中標籤住目前成本標籤值最小的節點時，均為永久標籤，意即當該節點一但被標籤住後，就不能再重新更新該節點的資訊，使得節點的上游點、成本標籤值、累積載貨等資料變成永久固定的，在永久標籤與優先限制(Precedence Constraints)等限制下，導致修正後 Dijkstra's 演算法在求解子問題網路時，通常無法找到最短路徑，因此建議後續的研究可以採用 Label Correcting 演算法，利用其同時具有暫時與永久標籤的解題特性，來求解具有提送貨特性的最短路徑問題。

參考文獻

1. CS Sung and JM Hong, “Branch-and-price algorithm for a multicast routing problem,” *Journal of the Operational Research Society* (1999) 50, 1168-1175.
2. Cynthia Barnhart, Ellis L. Johnson, G.eorge L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance, "Branch-and-Price :Column Generation for Sloving Huge Integer Problem," *Operations Research* Vol. 46. No. 3, May-June 1998.
3. Haibing Li and Andrew Lim, “A Metaheuristic for the Pickup and Delivery Problem with Time Windows,”
4. Hang Xu, Zhi-Long Chen, and Srinivas Rajagopal, and Sundar Arunapuram, “Solving a Practical Pickup andDelivery Problem,” *Transportation Science* © 2003 INFORMS Vol. 37, No. 3, August 2003, pp. 347–364.
5. Hoong Chuin LAU, Zhe LIANG, “Pickup and Delivery with Time Windows :Algorithms and Test Case Generation,”
6. ILOG, ILOG CPLEX 7.0 Getting Started, August 2000.
7. ILOG, ILOG CPLEX 7.0 User’s Manual, August 2000.
8. Jacques Renaud, Fayez F. Boctor, and Gilbert Laporte, “Perturbation heuristics for the pickup and delivery traveling salesman problem,” *Computers & Operations Research* ,29 (2002) 1129-1141.
9. Manfred Gronalt, Richard F. Hartl, and Marc Reimann, “New savings based algorithms for time constrained pickup and delivery of full truckloads,” *European Journal of Operational Research* ,151 (2003) 520–535.
10. Martin Savelsbergh, “A Branch-and-Price Algorithm for the Generalized Assignment Problem,” *Operations Research* Vol. 45. No. 6, November-December 1997.
11. Noud Gademann and Steef Van De Velde, ”Order batching to minimize total travel time in a parallel-aisle warehouse,” *IIE Transactions* (2005) 37, 63–75.
12. Pasquale Avella, Maurizio Boccia, and Antonio Sforza, “Solving a fuel delivery problem by heuristic and exact approaches,” *European Journal of Operational Research* ,152 (2004) 170–179.
13. Psaraftis, H. N., “Dynamic vehicle routing: status and prospects,” *Annals of Operations research* 61, 143-164.
14. Snežana Mitrovič-Minič , Ramesh Krishnamurti, and Gilbert Laporte, “Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows,” *Transportation Research Part B* ,38 (2004) 669–685.
15. Solomon, M., “Algorithms for the vehicle routing and scheduling problems with time window constraints.” *Operations Research* 35, 254-265.
16. William P. Nanry , J. Wesley Barnes, “Solving the pickup and delivery problem with time windows using reactive tabu search,” *Transportation Research Part B*, 34 (2000)

107-121.
17. 盧宗成，「捷運司機員排班問題之研究-以台北捷運公司為例」，國立交通大學運輸工程與管理學系碩士論文，民國八十九年六月。
18. 游雅惠，「捷運列車排班問題之研究—以台北捷運淡水-新店線為例」，國立交通大學運輸工程與管理學系碩士論文，民國八十九年六月。

簡　歷

姓　　名：黃信翔

籍　　貫：台中市

出 生 日 期：民國 71 年 8 月 11 日

聯 絡 地 址：高雄市三民區九如一路 95 巷 3 號 6 樓之 1

E-mail　　：shiang0811.tem93g@nctu.edu.tw

學　　歷：

民國 95 年 6 月　國立交通大學運輸科技與管理學系碩士班畢業

民國 93 年 6 月　國立交通大學運輸科技與管理學系畢業

民國 89 年 6 月　國立鳳山高級中學畢業

民國 86 年 6 月　高雄市立小港國民中學畢業

民國 83 年 6 月　高雄市立小港國民小學畢業