中華民國運輸學會 110 年
學 術 論 文 研 討 會
中華民國 110 年 12 月

復康巴士多點路線排程平行化演算法設計[1]

Parallel Processing Algorithm for Vehicle Routing Optimization Problem of Rehabilitation Bus

孫士勝[2]

蘇凰儀[3]

蔡明志[4]

郭建志[5]

摘要

現階段多數復康巴士業者仍採用人工模式進行運能估算、訂單排程以及路線規劃，平均一位專業的排班工作員需要花 1-2 天來排班，不僅曠日費時、消耗人力，而且效率不佳的排程結果往往會影響復康巴士的載運量。隨著老年化的社會來臨、醫療需求的增長以及交通路線的迥異，一旦訂單增加，人工排程的績效將更加難以提升，最後導致整體載客率低落。為了解決上述的問題，本研究規劃對復康巴士排程問題正規化，其目標可視情況設定為最小化棄單數量、最小化平均每輛車分配到的訂單數之差異程度、或最小化總空車時間（減少空時浪費）。除了常見的傳統車輛路徑規劃的條件以外，本研究規劃路線排程時評估相當嚴峻的條件包括：駕駛工時、隨著時段改變的旅行時間、共乘服務等。本研究透過混合整數線性規劃工具以現有的最佳解解法器在小型問題下找到最佳解、並估計現行人工計算之解答能夠改善的空間。透過貪婪演算法、局部搜尋、基因演算法等等概念來設計啟發式演算法以尋求可行解，並採用演算法平行處理的架構、加速程式的執行效率，可解決本研究中提出的復康巴士多點路線排程最佳化問題之變形，也得到較佳的排程結果。經實驗結果，20-22 輛復康巴士搭配 200-220 筆訂單透過本研究設計之平行化基因演算法加上局部搜尋約 1 分鐘內可取得最佳解，其計算時間比 Gurobi 的約 10 分鐘縮短許多；隨著車輛數與訂單數增加，以 Gurobi 計算 30 輛復康巴士搭配 300 張訂單花費超過 90 分仍無法求解，透過本研究開發之平行化演算法處理 100 輛復康巴士搭配 1000 筆訂單，利用基因演算法加上局部搜尋僅約 10 分鐘即可得解，可大幅度提高求解效率。

關鍵詞：復康巴士、多點路線排程、平行化演算法、VRP

Abstract

Currently, most rehabilitation bus companies still use manual calculation methods for capacity estimation, order scheduling, and route planning. On average, a professional scheduling worker takes 1-2 days to schedule, which is time-consuming and labor-intensive. Moreover, inefficient scheduling results often

[1] 本研究係財團法人中華顧問工程司小客車照護運輸服務多點路線排程最佳化演算法建置計畫之部分成果。

[2] 財團法人中華顧問工程司智慧運輸中心研究員兼 AIoT 研究室組長（聯絡地址：10637 台北市大安區辛亥路二段 185 號 28 樓，電話：(02)8732-5567 ext. 1313，E-mail: sssun@ceci.org.tw）。

[3] 財團法人中華顧問工程司偏鄉智行中心副研究員。

[4] 財團法人中華顧問工程司智慧運輸中心主任。

[5] 國立中正大學資訊工程學系暨研究所助理教授。

affect the capacity of rehabilitation bus. Once the orders increase, it will be more difficult to improve the performance of manual scheduling, which will eventually lead to a low performance. To solve the above-mentioned problems, this research normalizes the vehicle routing and scheduling problem of rehabilitation buses. The goal can be set as minimizing the number of abandoned orders, minimizing the variance of the average number of orders allocated to each vehicle, or minimizing the total empty time, etc. In addition to the common conditions of traditional vehicle routing problems, this research evaluates some severe conditions, including driving hours, travel time that varies with time of a day, and ride sharing services, etc. In this research, the mixed integer linear programming tools are used to find the optimal solutions under small size problem sets, and act as the estimator for improving the solutions of the current manual calculation methods. We design heuristic algorithms to find feasible solutions through the concepts such as greedy algorithm, local search, and genetic algorithm. Besides, we develop a parallel processing algorithm architecture to accelerate the execution efficiency. According to the experimental results, the optimal solution of 20-22 rehabilitation buses with 200-220 orders can be obtained in about 1 minute through our designed parallelized genetic plus local search algorithm, and the computing time is shorter than Gurobi's 10 minutes. As the number of vehicles and orders increases, Gurobi could not solve the condition for 30 rehabilitation buses and 300 orders within 90 minutes. Our designed parallel algorithm with genetic algorithm and local search can solve the problem set with 100 rehabilitation buses and 1,000 orders in only about 10 minutes, which greatly improves the efficiency.

Keywords: rehabilitation bus, Vehicle Routing Problems, VRP, parallel processing algorithm

一、前言

復康巴士為身障者往返醫院的社會福利設施，但多數縣市巴士的數量與身心障礙者之需求有極大的差距，以 2016 年的台北市為例，復康巴士 328 輛，服務趟次 66 萬 8716 次，以台北市 12 萬身心障礙市民來說，平均每人每年只能被服務不到 6 次，可謂供不應求。政府規定的現行服務之運作機制下，復康巴士業者必須按照復康巴士車輛數、工時限制等等因素來預先估計各個時段之運能，再將運能轉換成每個時段能提供的載運趟次；而後，使用者事先於相關網站、手機 App 或以電話預約叫車服務並註明時間與地點。為了能優先服務較高身心障礙等級的用戶，目前的機制是根據其身心障礙等級區別，以台北市為例：特 A 等級可於 5 天前開始預約，A1 等級則是 4 天半前，A2 等級是 4 天前，B 等級和非台北市民則是 3 天前，但每家業者略有差異；在整個系統結束用戶下訂後，大約 1-2 天內復康巴士將收到的訂單進行車輛排程，且需滿足所有已接受之訂單。目前多數醫院、診所皆提供預估就診時間，許多用戶使用復康巴士之服務進行例行性就診，排程後的結果需在預估就診的時間窗內將用戶送達目的地。因此，如何估計每個時段之運能以及如何在排程中滿足所有訂單(儘量不掉單)將是相當重要的問題。

表1 台北市復康巴士服務對象與身心障礙類別對照表。

障礙等級	開始預約日	身心障礙等級類別
特 A 等級	5 天前	植物人。重度以上下肢體障礙者。(障礙等級註記重度以上者)
A1 等級	4 天半前	重度以上視覺障礙者。(障礙等級註記重度以上者)
A2 等級	4 天前	不屬特 A、A1 等級之類別障礙。(障礙等級註記重度者)
B 等級	3 天前	所有類別障礙等級註記中、輕度障礙者。

現階段多數復康巴士業者仍採用人工模式進行運能估算、訂單排程以及路線規劃，平均一位專業的排班工作員需要花 1-2 天來安排一日的班表，不僅曠日費時、消耗人力，而且不同排班工作員的排程結果往往會影響巴士的載運量。隨著老年化的社會來臨、醫療需求的增長以及交通路線的迥異，一旦訂單增加，人工排程的績效將更加難以提升，最後導致整體載客率低落。

為了解決上述問題，本研究規劃對復康巴士排程問題正規化，目標可視情況設定為最大化

載運量的排程、最小化總移動距離（節省能源消耗）或最小化派遣車輛數等等。除了傳統車輛路徑規劃的條件以外，復康巴士排程問題在規劃路線排程時需要評估相當嚴峻的條件如下：

A. 駕駛工時：根據我國《汽車運輸業管理規則》第 19 條之 2 規定（交通部(2020)），駕駛人連續駕車 4 小時，至少休息 30 分鐘，採分次休息，每次應不得少於 15 分，駕駛的行車時間與休息時間需於排程中納入考量。

B. 隨著時段改變的旅行時間：交通深受不同時段的車流量影響，例如上下班時間的尖峰時段行車時間常大幅增加，然而傳統的車輛路徑規劃相關問題往往忽略這個因素。

C. 乘客共乘服務：不同乘客若能搭乘同一班車輛則可大幅減少不必要的空間與時間浪費，提高整體載客率以及巴士利用率，例如可安排視障乘客及輪椅乘客的順路共乘，因此本研究考慮共乘路線排程來提高滿足所有訂單的機率。

隨著台灣邁入高齡化社會，政府機關更加重視無障礙運輸服務的提供，其中運輸載客之效率與成本花費亦是主管機關須納入考量之議題，基於自動化排程設計具備效益極大化、滿足載客需求之特性，將可為無障礙運輸服務提供穩定、減少人力參與的路線規劃系統，也可更進一步建立更為廣泛之服務範圍。

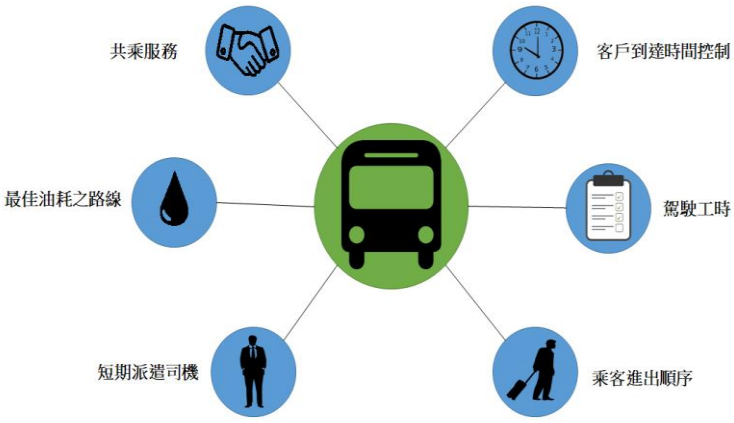

圖1 復康巴士路線排程最佳化問題考慮因素

二、文獻回顧

車輛路徑問題(Vehicle Routing Problems，以下簡稱為 VRP)最早是由 Dantzig 與 John(1959)提出，此車輛路徑規劃與排程的最佳化問題是指：客戶各自有不同的貨物需求，配送中心向客戶提供貨物，由一個車隊負責分送貨物，排成適當的行車路線，目標是滿足客戶的需求，並能在一定的限制下，達到諸如路程最短、成本最小、耗費時間最少等目的。由於 VRP 問題持續發展，考慮需求點對於車輛到達的時間有所要求之下，在 VRP 之中加入時窗限制，形成具時窗限制的車輛路徑問題（Vehicle Routing Problems with Time Windows，簡稱為 VRPTW），為複數車輛與複數起迄點可行路線規劃的組合最佳化問題。車輛除了要滿足 VRP 問題的限制之外，還必須要滿足需求點的時間窗限制，VRPTW 具有許多的變形，例如考量油耗、車輛數量等等問題於交通領域中，也有考慮接送位置的車輛路徑問題（Pickup and Delivery VRPTW，簡稱 PDVRPTW），透過巧妙設計的演算法不但能降低車輛行駛時間並且能減少油耗，也因考量到道路的擁塞程度而更進一步的分流車流量，達到減緩交通的擁塞現象。本研究目標的復康巴士排程問題為典型 VRP 問題之變型，以下說明相關的文獻。

2.1 相關文獻分類簡述

依據過往文獻之題目條件與應用情境，本研究將文獻分類為以下四類：

A. 復康巴士應用相關（似）車輛路徑問題之調查與研究，其中張朝能等人(2018)、陳其華等人(2019)、潘珮琪(2018)、白峻安(2015)、陳惠國等人(2013)、辛孟鑫(2007)、Haibing 與 Andrew(2003)、Sophie 等人(2010)、Jean-F.與 Gilbert(2003)、郭佳林(2004)、以及游珮暄(2013)皆可歸類於此類。此類文獻之題目條件式與復康巴士較為類似，考慮了硬時間窗、工時及車輛容量限制，但大多缺少共乘與休息時間的考量。其中陳其華等人(2019)、白峻安(2015)、陳惠國等人(2013)、辛孟鑫(2007)、為以各縣市復康巴士之應用情境為基準，擬定題目的條件式與目標式進行解題。而除了陳其華等人(2019)的實驗規模達 1087 單，其餘文獻均在 100 張訂單以下，且文獻中之目標式各異，包含最小化車輛空駛時間、旅行時間、路徑成本與使用車輛數，皆與本研究擬定之目標（最小化棄單數）不同。

B. 具硬時間窗限制之車輛路線規劃問題，其中 Dominik 與 Roberto(2013)、Jih 與 Hsu(2004)、黃漢瑄(2004)、李泰琳與張靖(2010)、Guo 與 Liu(2014)、Emanuel 等人(2007)、Brahim 等人(2006)、Zhihao 等人(2018)皆可歸類於此類。此類文獻情境較為多樣化且目標式也不盡相同，包含最小化行車路徑成本、總花費時間與使用車輛數。而其條件式大多缺少共乘、休息時間、工時或車輛容量限制，實驗規模則介於 20 至 400 點不等。

C. 不具時間窗限制或軟時間窗限制之車輛路線規劃問題，其中 Mohammadreza 等人(2018)、王春鎰(2007)、賴敬棠(2010)、Osaba 等人(2015)、Michel 等人(2015)、Arun 等人(2019)、許正良與謝益智(2012)、Liao 等人(2014)、Frank 等人(1981)、Nigel 等人(1976)皆可歸類於此類。此類文獻之題目多為探討解決傳統車輛路徑規劃問題（VRP），其題目條件考慮均較為侷限，不僅缺少了時間窗，也少了工時、共乘與休息時間等限制，題目之目標式則多為最小化行車路徑或時間成本。此類文獻的題目與本研究擬定之題目模型有較大的落差。

D. 車輛路徑規劃問題整理相關之文獻，此類文獻針對車輛路徑問題相關的研究進行整理，其中李美儀(2015)對於各種研究題目進行分類，Nasser(2010)則回顧了多種解題技巧與啟發式演算法。

2.2 相關文獻比較

本節整理並將分類內的文獻進行比較：

A. 復康巴士應用相關（似）車輛路徑問題之調查與研究：

張朝能等人(2018)針對台北的復康巴士進行調查，文獻中提到台北市的復康巴士通常是 4 個一般位和 1 個輪椅位，並統計 2016 年時，全台北市大約有 328 輛復康巴士。陳其華等人(2019)針對台南、嘉義的復康巴士排程問題進行探討，以電腦排程儘量減少空車里程和車輛空駛時間，並符合工時控管（勞基法），希望能夠縮短排程所需的作業時間、計算成本及產出營運預報，在總共有 146 車和 1087 張訂單的情況下，計算的結果接受 1052 張訂單，漏掉 35 張訂單。潘珮琪(2018)調查了台北市民搭乘復康巴士

服務成效，文中說明持有身心障礙證明之台北市市民可按照類別區分為特 A、A1、A2、B 等級，在 5 至 2 日前申請復康巴士之預約，而復康巴士服務的時間分佈情形呈現三個尖峰，分別是 8-9 時、11-13 時、15-16 時。本研究也按照了這個調查來產生模擬測資，用來驗證本研究所開發演算法的效果。

白峻安(2015)探討了新北市復康巴士的排程問題，為單一場站、硬時間窗、所有車的容量皆相同、限制每日最長工時，但無考慮中間休息時間，求解目標為最小化總空駛距離，方法為貪婪演算法（Greedy Algorithm）加上局部搜尋（Local Search），隨機產生粒子（例如：10 個），如果迭代成果優於前者則進行迭代（進行 10 次），更新答案的方式為先計算出作用力（局部最佳解移動方向），之後根據作用力移動來更新局部最佳解，測試的題目大小最高到 3 台車和 50 筆訂單。陳惠國等人(2013)以復康巴士為例來探討撥召運輸問題，提出了改良後的蜂群最佳化演算法（Artificial Bee Colony Algorithm）來求解，其目標為最小化總旅行時間和總等待時間，題目限制為單一場站、給定上下班時間、訂單時間為硬性時窗，車輛容量相同、車輛數已知，考慮共乘；提出的方法先利用貪婪演算法產生初始解，之後透過蜂群演算法做局部搜尋修改答案，測試題目之大小為 48 張訂單、3 車，平均每單 22 分鐘（包含空車時間），與台北市平均每單 45 分鐘不一樣。

辛孟鑫(2007)以台北市復康巴士為例，所探討的題目考慮相同車輛，有容量和時間窗限制，目標是減少車輛使用數和乘客旅行時間，提出一般啟發式演算法（Heuristic）求解，概念是觀察時間相近的訂單做媒合，儘量讓共乘可行，測試題目之大小為 49 張訂單及單一場站。Haibing 與 Andrew(2003)主要的目標為最小化車輛使用和旅行時間，也考慮了客戶等待時間，題目限制為車子容量、硬時窗、單一場站，解題技巧為貪婪演算法決定初始解，之後透過禁忌搜尋演算法（Tabu Search）來進行局部搜尋，其實驗之規模為 100 張訂單。Sophie 等人(2010)研究的最佳化問題中，條件限制為車輛數已知、具車輛容量限制、硬時間窗、單一場站和工時限制，目標是最小化總路徑成本，演算法是先以出發時間排序再使用貪婪演算法產生一組初始解，接著透過局部搜尋和鄰近點交換訂單，實驗的規模是 24-144 筆訂單，車輛數為 3-13，執行時間約為 22 分鐘。

Jean-F.與 Gilbert(2003)探討了車輛最佳化問題中，考量了車子容量限制、時間窗、工時限制與車子的數量，目標是最小化總路徑成本，演算法設計為禁忌搜尋法，實驗最多支援到 295 張訂單、20 輛車，執行時間約為 34 分鐘。郭佳林(2004)探討車輛旅途問題之延伸－具時間窗之多趟次車輛旅途問題，題目的限制是車輛容量、硬時窗、工時限制，解題目標為降低使用車輛，同時維持較低的路徑成本，演算法的設計為貪婪演算法搭配禁忌搜尋，塞單時若違反條件則放棄，以避免違反條件，可能將原有的訂單拔除，把特定的訂單分配給特定車輛。模擬測試約 100 個需求點，平均需要 300 秒才能取得初始解。游珮暄(2013)從復康巴士場站位置切入，探討如何選擇較為良好的場站設置位置，其研究發現，在台北市總需求人數為每小時 92 人情形下，所需車輛數則為 37 輛，場站需建置 5 個，分別為北投區、士林區、內湖區、萬華區及文山區。

上述與復康巴士應用相關問題之研究文獻中，其解題條件與本研究考慮共乘、休息時間、及工時限制各有所異，且目標式均與本研究的最小化棄單數不同。

B. 具硬時間窗限制之車輛路線規劃問題

Dominik 與 Roberto(2013)允許答案違反題目限制，目標是減少違反所造成的懲

罰，題目的限制是硬時窗、車輛數已知且每輛車相同容量，單一場站及設定工時限制，演算法修改了禁忌搜尋法而來，透過局部搜尋修改答案，實驗的規模為 24-144 張訂單。Jih 與 Hsu(2004)探討的題目考慮到時間窗、單一車輛、單一場站，然而沒考慮工時限制，目標是最小化總花費時間和車輛等待之時間，其演算法的精神是族群競爭式基因演算法（Family Competition Genetic Algorithm），與傳統的差別在於會有多個後代的家庭，每個隨機選擇的配偶都會有一個後代，家庭競爭中維持人口的恆定，只有勝出的家庭才能倖免。其實驗的規模在 50 筆訂單時和最佳解相同，規模最高到 100 筆訂單。

黃漢瑄(2004)比較了多個常見的演算法，例如：基因演算法（Genetic Algorithm）、族群競爭式基因演算法、蟻群演算法（Ant Colony Optimization）在車輛路由問題上面的效能，題目的目標是最小化總路線長度，限制為車子數量已知、車輛容量、車輛運行時間、具時間窗、用戶乘車時間和等待時間，解題過程中允許違反條件，但違反條件會有額外懲罰。在比較了基因和蟻群演算法以後，發現基因演算法適應性較為佳，其測試規模為 11 輛車及 120 筆訂單。李泰琳與張靖(2010)探討了具時間窗之收卸貨問題，題目中每個地點必須服務一次，而且要在服務時窗限制內完成，同一地點卸貨會在收貨時間前發生，車輛具有容量上限，車子離開和回到場站必須為空車，過程中必須保持流量守恆，使用並延伸蟻群演算法來解題，其測試規模為 100 個節點，求解平均時間約為 26 分鐘。

Guo 與 Liu(2014)探討撥召車輛路徑規劃問題，其考慮單一場站、給定車子數量、時間窗限制、工時限制，但是沒有考慮容量限制，目標是最小化使用車輛，演算法的設計為貪婪演算法。測試的是 400 張訂單。Emanuel 等人(2007)探討的車輛路由問題是接送所有用戶從出發點到達目的地，條件是多場站、多車種、容量限制，但使用軟時窗，可以允許違反條件，題目的目標是最小化路徑成本。因為使用既有的解法器來解題，實驗規模極小，僅僅測試 2-5 張訂單。Brahim 等人(2006)研究了照護運輸的排程，並使用基因演算法來解題，其考慮的題目限制為多場站、軟時窗、車輛容量具有限制，目標為最小化使用的車輛以及違反時間窗造成的懲罰。由於基因演算法在 crossover 之後可能產生不符合時窗的排程，此篇方法有額外使用 Best Fit 的方式，嘗試處理不符合時間窗的訂單，其實驗的規模為 16 台車和 164 筆訂單。Zhihao 等人(2018)探討在給定多台車之下的路由排程問題，考慮了多場站、時間窗、車輛容量限制，但沒有工時限制和休息時間，目標是最小化行車距離，方法採用粒子群演算法（Particle Swarm Optimization），實驗規模是 1-13 輛車。

此類文獻題目條件除了缺少共乘、休息時間外，Jih 與 Hsu(2004)、李泰琳與張靖(2010)、Emanuel 等人(2007)、Brahim 等人(2006)、Zhihao 等人(2018)等文獻亦沒有工時限制，Guo 與 Liu(2014)則無考量車輛容量。而 Dominik 與 Roberto(2013)、黃漢瑄(2004)允許解題違反題目條件，但於復康巴士排程問題中，是不允許答案違反題目限制條件的，違反條件限制的答案無法應用於實務情況中。

C. 不具時間窗或軟時間窗限制之車輛路線規劃問題

Mohammadreza 等人(2018)採用強化式學習（Reinforcement Learning）來解決傳統車輛路由問題（VRP），目標是讓強化式學習模型模仿最佳解的選擇，訓練過程儘量減少與最佳解的差距（用損失函數來計算），實驗和 Google OR-Tools（Google (2014)）、啟發式演算法比較，結果是 10-20 個點時，非常接近最佳解，但是當 50、100 個點時，仍是啟發式演算法效果較好。

王春鎰(2007)探討車子固定容量、單一場站，目標是最小化使用車輛。演算法的設計是先產生 6 種初始解，進行鄰近搜尋修改答案，透過門檻接受法（Threshold Accepting）、大洪水法（Great Deluge Algorithm）搜尋答案，其實驗的最大規模為 32 點。賴敬棠(2010)採用螞蟻記憶系統（Ant Memory System）來解車輛旅途問題，題目的目標是最小化運輸成本，條件為單一場站、載重限制且車輛皆相同，解題的概念是將每隻螞蟻根據建構準備產生一組解，之後進行費洛蒙的更新，運用區域搜尋改善每一隻螞蟻所建構的解答，最後選擇最好的解答進行全域的費洛蒙更新。

Osaba 等人(2015)在車輛路由裡面考慮了不對稱的旅行時間，也就是 A 點到 B 點的旅行時間不等於 B 點到 A 點的旅行時間，其他條件為單一場站、車子數量已知以及車輛容量限制，但是該問題不考慮時窗限制，目標則為最小化旅行時間，解題方法是隨機產生初始解答，之後透過基因演算法進行局部搜尋修改答案，測試的規模為 50 張訂單。Michel 等人(2015)探討了一台車的收送貨問題，類似於傳統 TSP 問題，其限制是車輛容量，使用了 Branch and cut solver 解題，測試的題目規模最大為 199 點。

Arun 等人(2019)利用強化式學習來解決傳統的車輛路由問題，題目之限制為單一場站和車輛容量，但沒有考慮時間窗，目標是最小化使用車子數量和成本，也就是距離或時間之成本，實驗的規模最高到達 1000 點，最後的結論是當使用者是平均分佈的時候，傳統啟發式演算法的效果較佳，而當使用者分佈具有聚集性的時候，強化式學習的效果會較好。許正良與謝益智(2012)探討了自動販賣機貨物配送問題，單一場站出發，必須經過所有要送貨的點，但是沒有考慮時間窗，目標是最小化總路徑長度，使用的方法為免疫演算法（Immune Algorithm），實驗規模為 2 輛車、共 212 個節點須拜訪。Liao 等人(2014)探討了最小化延遲的貨物收送問題，其題目限制為只有一輛車並且車子具有容量限制，但不考慮時間窗，目標是最小化延遲時間，也就是需求者等待供應者的運輸時間，因為題目不考慮時窗，所以非常適合使用基因演算法且不容易產生非法解。Frank 等人(1981)是第一篇提出 PDP（Pickup and Delivery Problem），Nigel 等人(1976)則是第一篇提出 DARP（Dial-a-Ride Problem）。

此類文獻之題目條件較少，於傳統的車輛路徑問題中，並沒有考慮時間窗、休息時間、共乘、工時限制等條件，且其目標多為最小化總路徑和時間成本，因此已與本研究擬定之題目模型有較大的差異。

D. 車輛路徑規劃問題整理相關之文獻

李美儀(2015)整理了近幾年來國內外之研究，大致上將車輛路線規劃問題分類為三大方向，1）顧客需求特性之車輛路線相關問題、2）車輛特性之車輛路線相關問題及路線規劃之變形、以及 3）多趟次車輛路線問題，該調查型論文整理了許多相關文獻，適合進行相關研究之研究人員參閱。

Nasser(2010)調查了多篇有關於車輛路由問題之解題技巧，裡面回顧許多數學規劃（Mathematical Programming）的解題技巧和啟發式演算法等等，傳統上採用數學規劃的方式可以求得最佳解，但是解題時間會隨著題目規模而呈現指數成長，因此有不少使用啟發式演算法透過局部交換兩台車的訂單來產生新的答案，也有不少類似的基因演算法來交換兩台車的訂單。

2.3 相關最佳化問題文獻對照表

本節將相關的文獻列表比較，並簡列相關文獻之題目與條件、應用場域、解題技巧、及實驗規模與效能。

表2 相關最佳化問題文獻對照表

編號	年份	題目與條件	應用場域	解題技巧	實驗規模與效能
本研究	2021	最小化棄單數 車輛數已知 硬時窗、工時、休息時間 每張訂單自訂是否共乘 彈性休息時間、車子容量 可自訂一般與輪椅位數 可考慮訂單公平性、空駛	台北 復康	貪婪初始 基因 局部搜尋 平行處理	1000-1600 單 100 車 全部不共乘 不漏單 貪婪執行 20-30 秒 若用基因+局部搜尋調整接單公平性和降低空駛時間需花 10 分鐘
陳其華等人(2019)	2019	最小化空車時間、空駛 硬時窗、工時、休息時間 沒有共乘	台南 嘉義 復康	不明	1087 單 146 車 漏掉 35 單 執行 30 分
白峻安(2015)	2015	最小化空駛 單一場站、硬時窗、工時 沒有休息時間	新北 復康	貪婪 局部搜尋 粒子	3 車 50 單 執行時間不明
陳惠國等人(2013)	2013	最小化總旅行與等待時間 單一場站、工作時間給定 硬時窗、車子容量 車輛數已知、共乘	復康	貪婪 蜂群	48 單 3 車 執行時間不明
辛孟鑫(2007)	2007	最小化使用車輛和旅行時間 單一場站、車子容量 時窗、共乘	台北 復康	一般啟發式演算法	49 單 執行時間不明
Haibing 與 Andrew(2003)	2003	最小化使用車輛 車子容量、硬時窗 單一場站	無	貪婪 禁忌搜尋	100 單 執行 6500 秒
Mohammadreza 等人(2018)	2018	傳統車輛路由問題（VRP）	無	強化式學習	最多 100 點
Dominik 與 Roberto(2013)	2013	減少違反條件的懲罰 硬時窗、車輛數已知 車子容量相同	無	禁忌搜尋	24-144 單 3-13 車 執行 51.81 秒
王春鎰(2007)	2006	最小化使用車輛 固定車子容量 沒有時間窗	新竹 科學 園區	局部搜尋 門檻接受 大洪水法	最大 32 點 執行時間不明
Jih 與 Hsu(2004)	2004	最小化總花費時間 單一場站、單一車輛	無	競爭式基因演算法	最大 100 訂單 執行時間不明

編號	年份	題目與條件	應用場域	解題技巧	實驗規模與效能
		時間窗，沒有工時限制			
Sophie 等人 (2010)	2010	最小化路徑成本 車子容量、硬時窗 單一場站、工時限制	無	貪婪 局部搜尋	24-144 單 3-13 車 執行 21.54 分
Jean-F.與 Gilbert(2003)	2003	最小化路徑成本 車子容量、時間窗 工時限制	無	禁忌搜尋	295 單 20 車 執行 33.88 分
郭佳林(2004)	2004	多趟次 VRP 最小化使用車輛 車子容量、硬時窗 工時限制	無	貪婪 禁忌搜尋	100 點 計算初始解答需要 300 秒
賴敬棠(2010)	2010	車輛路由問題 VRP 多車輛，容量皆相同	無	蟻群	5-18 車 50-199 車 執行 706 秒
黃漢瑄(2004)	2004	最小化路線 車輛數已知、車子容量 軟時窗（有懲罰） 用戶乘車和等待時間限制	無	基因 族群競爭 基因 蟻群	11 車 120 單 執行 6973 秒
李泰琳與張靖 (2010)	2010	具時窗之收卸貨問題 車子容量、場站	無	蟻群	100 點 執行 25.95 分
Osaba 等人 (2015)	2015	最小化旅行時間 不對稱旅行時間 單一場站、車輛數已知 車子容量，沒有時窗	無	隨機初始 基因	50 單 執行 92.73 秒
Guo 與 Liu(2014)	2014	撥召車輛路徑規劃 最小化使用車輛 單一場站、時窗限制 工時限制，沒有容量限制	無	貪婪	400 單 執行 93 秒
Michel 等人 (2015)	2015	TSP 考慮車子容量	無	Branch and cut solver	5-24 pickup 5-24 delivery 執行 1 秒
Arun 等人 (2019)	2019	傳統車輛路由問題 VRP	無	啟發式 強化式	1000 點 執行 5-8 小時
許正良與謝益智(2012)	2012	自動販賣機貨物配送 最小化總路徑長 單一場站，必須經過所有點 沒有時間窗	無	免疫算法	2 車 212 點 執行 433.5 秒
Liao 等人(2014)	2014	貨物收送 最小化延遲時間 一輛車、容量限制 沒有時間窗	無	基因	300 點 執行時間不明
Emanuel 等人	2007	最小化路徑成本	無	現有工具	2-5 單

編號	年份	題目與條件	應用場域	解題技巧	實驗規模與效能
(2007)		軟時窗、多場站、多車種 容量限制			執行 0.78 秒
Brahim 等人 (2006)	2006	最小化使用車輛 多場站、軟時窗、車子容量	無	基因	164 單 16 車 執行 20 分
Zhihao 等人 (2018)	2018	多車 VRP，最小化行車距離 車輛數已知、車子容量 時間窗、多場站 沒有工時限制	無	粒子	20-100 點 1-13 車 執行 189 秒

三、問題定義與解決方案

本研究開發復康巴士路線排程演算法，將提升復康巴士排程智慧化程度，減輕排班人員負擔，透過電腦執行複雜的排程問題，相較於人工排程，本研究將大大提升排班人員之效率，並在訂單數量增加時也能夠保持高速計算，使得相關單位之人力使用更加有效率。

3.1 問題與目標定義

為了提升復康巴士整體載客率，本研究透過 VRPTW 及 PDVRPTW 相關問題之延伸，在給定訂單的情況下，設計演算法儘量地接下所有訂單並計算較佳的排程，而排班人員可透過實際能接下的訂單數來估計每小時運能，以電腦快速運算來安排複雜的路線規劃取代人工排程。本研究著重於台北市復康巴士的路線排程與規劃，若能有效改善排程效率問題，也對其他縣市產生良性外部效果，以利未來可延伸至其他縣市。

考量到目前復康巴士營運為先接單後排班，為了滿足所有接到的訂單可被提供服務，本研究把**目標設定為在有限的車輛數和嚴峻的條件（駕駛工時、隨著時段改變的旅行時間、共乘服務等，以上限制式可根據實務需求調整）下，目標式可包括儘量滿足所有訂單（即最小化棄單數量）、最小化平均每輛車分配到的訂單數之差異程度（司機間接單量的公平性）、以及最小化總空車時間（減少空時浪費）**。如此可透過高效演算法解決復康巴士的排程問題。此外，本排程演算法也可被應用在運能估算，只要計算出不同日期給定的訂單，將不同日期、不同時段下能夠服務的訂單數量進行平均，便可估算每個時段可接受的訂單數，以提供使用者下訂。

3.2 解決方案

本研究為使演算法設計能在合理時間內取得效果較佳的復康巴士路線排程，規劃之解決方案如下說明：

A. 統一參數輸入輸出格式：建立一套排程最佳化程式之輸入輸出檔案的規則。為求統一的輸入與輸出格式，程式僅接受特定資料格式，且程式輸出格式亦有嚴格規範，本研究使用 JSON 格式作為輸出輸入。

B. 釐清目標和條件限制：本研究首先簡化了復康巴士路線排程問題，並嘗試正規化該問題類似 VRPTW（或 PDVRPTW）的問題，而後採取「混合整數規劃」(Mixed Integer Programming，MIP）之組合最佳化工具來對應相關數學式，其中**目標式為最佳化之目標（例如：最小化放棄的載客數量，即最小化棄單數）**，而**限制式根據實務需求調整設定為必須達成或不能違反的條件（包括：每位乘客必須在預約時間前抵達目標地點、駕駛工作時間限制等等）**，此數學系統的建立有利於系統性地分析問題。除了使用最佳化工具來了解最佳解的性質以外，本研究綜合多種演算法之設計及運算，從中挑選有能力取得較佳結果的方法，而後繼續修改和延伸，以期設計出計算時間較短、答案品質優良的演算法。

C. 估計現行人工計算之解答與最佳解的差距：本研究過程利用混合整數線性規劃工具與最佳解解法器（例如：Google OR-Tools （Google (2014)）、AMPL （AMPL (1985)）、和 Gurobi （Gurobi (2008)）等），在可容忍的計算時間內找到簡化後的題目之最佳解並利用此最佳解估計我們提出的演算法解答之品質，最終使用 Gurobi 作為與本研究開發之演算法作為比對之 benchmark。

D. 演算法與程式設計：本研究利用了以下數學及演算法技巧，包括：貪婪演算法（Greedy Algorithm）、局部搜尋（Local Search）、基因演算法（Genetic Algorithm）等概念，進一步設計啟發式演算法（Heuristic），同時透過演算法之平行化設計，可快速尋求可行解。

E. 測資產生器開發：本研究也開發了測資產生器以自動產生訂單，包括訂單要求的接送、抵達的時間與地點等資訊，供模擬和驗證程式效能使用。

F. 實驗模擬：程式設計完成後即利用上述測資產生器產生測試資料，用來驗證程式可行性以及運算速度，透過不同的實驗來展示程式效能與特性，並將結果圖表化。若未來可透過復康巴士業者取得真實營運資料，也能透過真實資料來對程式的效能進行調校。

3.3 達成目標之限制

VRPTW 和 PDVRPTW 是相當困難的 NP-hard 題目，許多的題目變形就連在題目僅給定一輛車時，搜尋一組可行解都是 NP-complete 的難度，換句話說，判斷是否存在可行解已經是不能在保證多項式時間內完成。

因此，當達成目標之限制式增加或是變數數量增加，VRPTW 和 PDVRPTW 即無法在合理時間內取得最佳解，例如：當車輛與乘客數量上升，導致路線的組合數成指數上升，造成解題難度的提高，所需之求解演算時間亦隨問題變數數量呈指數關係成長，使得該問題無法在合理的時間下求得最佳解，甚至連尋找可行解都有困難。此外，根據復康巴士業者不同的需求而存在許多變形的條件限制式，例如：共乘問題、駕駛工時及休息時間等等，因此復康巴士的排程問題又比 VRPTW 和 PDVRPTW 問題更加困難。

當 VRPTW 和 PDVRPTW 問題規模擴大時，為了確保可在接受的時間內得到合理或效果較好的可行解，發展解題上具效率性的啟發式解法也成為復康巴士排程問題相關重點－需要使用有效率的局部搜尋等類似的啟發式演算法，以確保可在合理時間內求出答案。

為了達成上述之目標，本研究**透過大量復康巴士相關研究文獻回顧，根據文獻中相關的貪**

婪演算法（Greedy Algorithm）、基因演算法（Genetic Algorithm）、局部搜尋（Local Search）等概念，重新設計適合復康巴士排程問題的啟發式演算法，並提供可修改的函式和參數，方便使用者日後維護和調校效能。

四、演算法設計與程式規劃

本研究回顧了大量復康巴士相關排程問題之文獻進行整理及比較，並訪談了台北市復康巴士業者，根據客戶需求建立題目模型，確定題目條件限制，接著與過往文獻之研究題目比較差異，了解這些文獻的解題技巧與成果，進而進行演算法的設計及改良。

4.1 問題條件與建立問題模型

本節說明本研究欲解決之問題條件及問題模型之建立，說明如下：

A. 路網資訊：
   i. 多個場站
   ii. 所有乘客訂單起迄點
   iii. 所有起迄點、場站之間的旅行時間（分鐘）

B. 時間與成本：
   i. 起點或迄點時間
   ii. 時窗限制為硬時窗，是指每個時段的時間區間
   iii. 上下車服務所需時間
   iv. 駕駛工作時間限制，是指上下班的時間差的上限
   v. 駕駛最長連續工時限制及休息時間限制，是指連續工作時間達到最長連續工時限制後，必須休息的時間長度
   vi. 彈性休息開始時間：休息開始時間可以前後挪動一個可容忍的時間長度
   vii. 共乘時可允許之最大繞路時間，是指原本起迄點之間的旅行時間最多可延長多少

C. 需求狀態：
   i. 乘客的起迄點
   ii. 乘客所需之座位類別及需求量
   iii. 乘客是否願意共乘

D. 車輛資訊：
   i. 各車的座位類別及容量
   ii. 車輛數
   iii. 多個車輛出發及結束之場站

E. 最佳化目標：
   i. 最小化棄單數
   ii. 最小化平均每輛車分配到的訂單數之差異程度
   iii. 最小化總空車時間
   iv. 除了上述所列，本研究開發的程式允許修改目標式，可以在未來使用時進行調

整；以上三項最佳化目標可透過標準化並調整權重，可滿足業者對於不同需求目標下的調派需求

其中，於題目模型之最佳化目標，本研究的設計考量如下：

A. 最小化棄單數：復康巴士採放趟預約制，於既有訂單下必須將所有訂單接完，即棄單數必須為 0，但若將棄單數等於 0 當作題目之限制式，則有可能因訂單數目過多產生無解之情形。所以本研究將此限制條件轉換為最小化棄單數之目標式，即盡可能將所有單子接完，確保題目有可行解。

B. 最小化平均每輛車分配到的訂單數之差異程度：根據復康巴士業者提供的實際情況，復康巴士駕駛之間會比較工作中接單數量之多寡，而有心理不平衡的情形發生。為此考量到駕駛間互相比較之公平性，本研究將此設為目標式，即盡可能平均每位駕駛的接單數量，以降低駕駛間因互相比較所產生之心理影響。

C. 最小化總空車時間：加入此目標式之目的為最大化車輛於工作時間中的使用率，即在相同時間下讓車輛盡可能接到更多訂單，以利接單數量之提升。

4.2 演算法設計規劃

統整本研究所回顧的復康巴士排程問題相關文獻，大多數文獻與本研究存在一定落差，除了題目條件缺少休息時間、共乘與駕駛工時等限制，其求解目標也與本研究不同，使得文獻的解題方法無法直接使用於本研究中。因此，本研究參考了文獻中的演算法進行設計與修改，透過結合了貪婪演算法、基因演算法和局部搜尋的精神，利用貪婪演算法產生初始解，再配合基因演算法，做局部搜尋修改答案以產生更好的解答。本研究設計演算法的考量與細節如下：

A. 貪婪演算法：本研究利用貪婪演算法產生初始解，因為貪婪演算法通常能給予較好的答案，而白峻安(2015)、陳惠國等人(2013)、Haibing 與 Andrew(2003)、Sophie 等人(2010)、郭佳林(2004)、Guo 與 Liu(2014)等文獻所使用的經典啟發式演算法與解題技巧，如：蟻群演算法、禁忌搜尋、局部搜尋等，也是利用貪婪演算法產生初始解。

B. 基因演算法：過往的文獻中，Jih 與 Hsu(2004)、黃漢瑄(2004)、Osaba 等人(2015)、Liao 等人(2014)、Brahim 等人(2006)使用了基因演算法進行解題，其中黃漢瑄(2004)比較了多個常見的啟發式演算法，如：基因演算法、競爭式基因演算法、蟻群演算法等，發現基因演算法的表現較為良好，本研究亦採用基因演算法作為演算法的核心設計。而於經典的旅行推銷員問題（Travelling Salesman Problem，TSP）中，基因演算法的過程為挑選兩個父代路徑，抽取片段路徑作為基因於父代路徑中進行交換，再將重複拜訪的地點做取代，以此產生子代路徑，但是於有時間窗限制的排程問題中，若直接將訂單當作基因在車輛的班表中進行交換，則很容易因為不符合訂單的時間窗限制或共乘等條件而產生非法解。因此本研究的基因演算法利用機率當作基因，挑選兩個訂單最多與三個訂單最少的班表進行重排，每筆訂單依循貪婪演算法的規則（能最早完成車輛目前接到的所有訂單）排序五輛車接該筆訂單的優劣程度，同時訂單選擇優劣車輛會有五個機率（例如：基因為 40%、30%、15%、10%、5%，代表選取最優車輛之機率為 40%，次優為 30%，依此類推），訂單再根據機率選取車輛排入。待重排完成後，將答案較好的機率留下，選取兩組機率做平均產生子代機率，該作法能避免非法解的

出現，同時保有基因演算法留存菁英進行繁衍的特性。

C. 局部搜尋：白峻安(2015)、Haibing 與 Andrew(2003)、Dominik 與 Roberto(2013)、王春鎰(2007)、Sophie 等人(2010)、Jean-F.與 Gilbert(2003)、郭佳林(2004)中，皆利用局部搜尋的技巧局部調整答案，可以避免貪婪演算法產生不好的解答，用以提高答案的品質。

為此，本研究開發之程式框架（Framework），採用貪婪演算法產生初始解，搭配局部搜尋和基因演算法來調整解答，演算法運算至給定的回合數限制即停止。若需要改變限制條件與目標式、新增參數，皆可透過修改「產生初始解函式」、「判斷解答品質函式」和「局部修改解答函式」來達成所希望的結果，本研究設計的平行計算框架如圖 2所示，三個函式介紹如下：

A. 產生初始解函式：
   i. 設計思維：因為本研究設計的平行程式框架透過不斷修改當下的解答來得到更好的答案，因此必須設計一個產生一組初始解的方法。
   ii. 函式步驟：本研究利用貪婪演算法產生初始解，首先按照訂單之起點時間對所有訂單做排序，若訂單無起點時間，則用迄點時間減去旅行時間，接著依序將訂單分配到目前已經有接訂單且接下此訂單後「能最早完成目前收到的所有訂單」的車輛，若目前不存在已經有接訂單的車輛可以接送此乘客，則使用尚未接過訂單的車輛來接此訂單，一直重複這個動作，直到無法再接收新的訂單，函式結束。
   iii. 留意事項：這個初始解不一定是可行解（Feasible Solution），可能存在一部份的棄單，之後將透過局部修改得到更好的解答。初始解的品質會影響最後答案的結果，未來維護的時候也可以透過修改此函式提昇效能。

B. 局部修改解答函式：
   i. 設計思維：這個函式主要的作用在於局部修改，在當前解答附近找尋另一組局部更好的解答。
   ii. 函式步驟：第一次進入此函式時，會先隨機產生 n 組機率當作基因，針對每一組基因，以初始解為基底，進行多次局部修改產生 1 組解答，用以取得 n 組解答（子代，子代的數量即為平行計算的執行緒線程數）。更具體地說，在進行局部修改時，每次將挑出 3 輛 OD 旅行時間總和最少的車輛與 2 輛最多的車輛，將其所有訂單移到漏接訂單中並進行重排，每張訂單將依照貪婪演算法（能夠最早完成手上訂單的時間）來排序這 5 輛車，愈接近貪婪演算法選擇之車輛，則有較高的機率被選到，接著隨機抽取出車子並將訂單排入該車輛，例如：基因為（40%、30%、15%、10%、5%）表示放到貪婪演算法最想選擇的車輛之機率為 40%。如此一來，能夠儘量避免產生和初始解一樣的 n 個解當作子代，幫助提高求解範圍的多樣性，能夠有效避免困在單一局部最佳解，同時透過平行計算來縮短完成時間，產生品質較佳的初始解。第二次進到這個函式以後，每次先留下上一代中結果較佳的 n/2 組機率（基因），再進行 crossover 得到另外 n/2 組基因，並以當前最佳解搭配這 n 組基因進行局部修改，取得 n 組新的答案，若產生比當前最佳解更好的答案，則更新成當前最佳解。最多進行 t 次產生子代就結束程式，並回傳當前最佳解。
   iii. 留意事項：修改一組答案的過程會經過 m 次的局部修改答案，若是產生出來的答案比當前最佳解來得好，則紀錄此組新產生的解，並根據此答案繼續產生新的答案；若該答案比當前最佳解差，則捨棄該答案。

C. 判斷解答品質函式：

i. 設計思維：這個函式影響答案的走向，也可視其為目標式，在有限的車輛下設定的目標包括：減少棄單的個數、訂單平衡、減少空車時間。

ii. 函式步驟：計算棄單的個數、車輛平均單數差總和以及車輛總空車時間，將這些元素乘上不同權重後得值（權重可根據使用者狀況調整），以此當作評斷當前解答品質的依據。

iii. 留意事項：判斷方式也可以根據使用者的目標來調整該函式的計算方式，可透過修改程式達成這個目標。

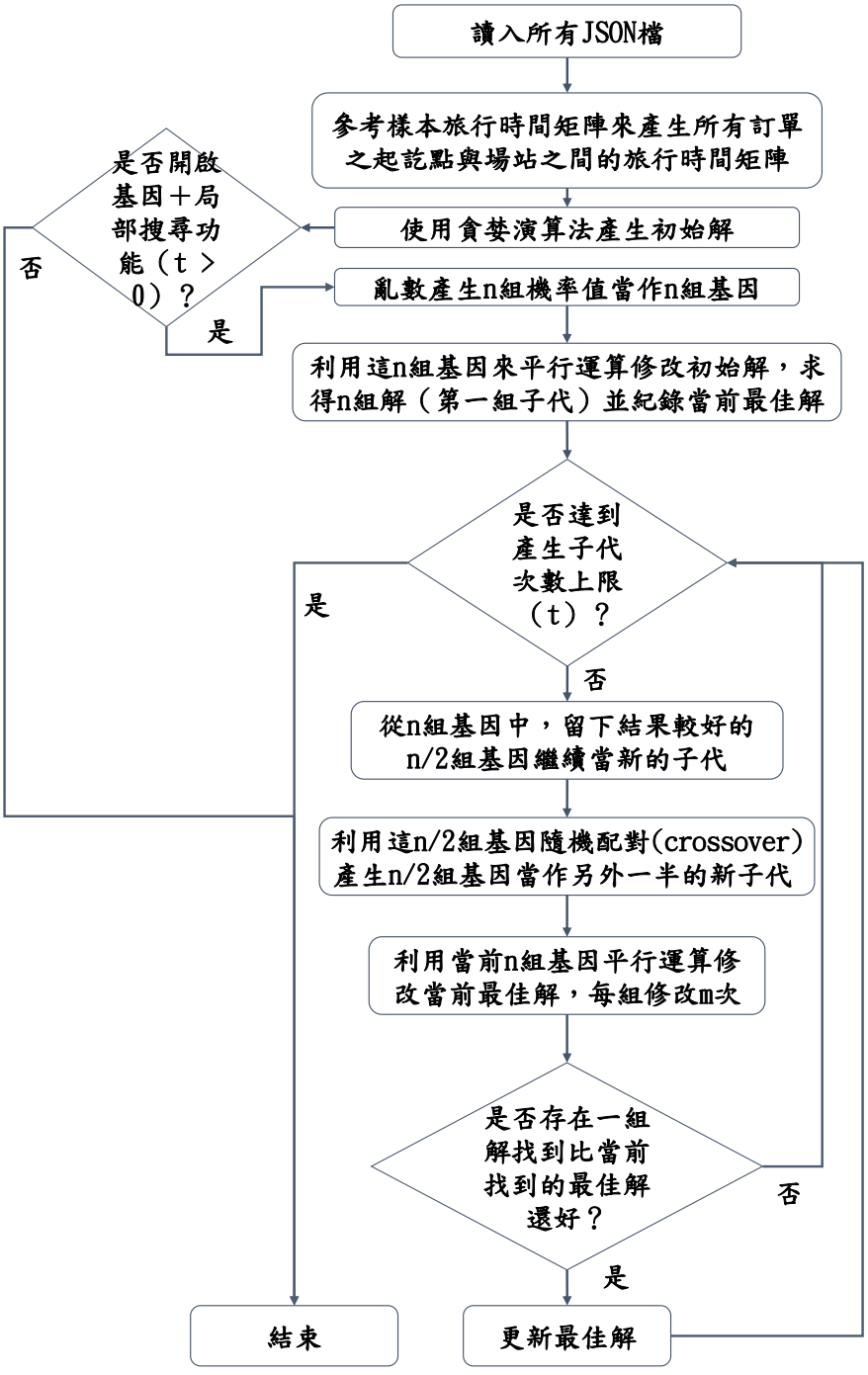

圖2 本研究平行化演算法架構

五、模擬建置與實驗結果

本研究使用平行化演算法框架，以下透過設定求解目標為「最小化棄單數量」、儘量把所有訂單接完作為本研究之驗證，模擬建置設定及實驗成果分述如下：

5.1 模擬之限制式、參數設定及模擬環境設定

本章欲求解之題目條件限制式如下：

A. 搭乘者需求條件：
   i. 搭乘需求時間（上車時間或最晚到達時間）
   ii. 起迄地點（OD）
   iii. 輪椅位及（或）一般座位之數量

B. 車隊相關需求條件：
   i. 車輛（駕駛）每日最長工時限制
   ii. 車輛（駕駛）最長連續行車時間
   iii. 駕駛最短休息時間
   iv. 每車到各點位服務時間
   v. 車輛起迄點
   vi. 彈性休息或下班的開始時間

模擬排程實驗之相關設定如下：

A. 訂單規模：
   i. 200-220 張訂單（小型測資，與 Gurobi 比較）
   ii. 1000-1600 張訂單（標準測資）
B. 3 個停車場站、車輛與訂單等資料
C. 車輛數量：
   i. 20-22 輛車（小型測資，與 Gurobi 比較）
   ii. 100 輛車（標準測資）
D. 總工作時段：9 小時，工作 4 小時需休息 30 分鐘
E. 共乘時可允許之最大繞路時間：30 分鐘
F. 相同 OD 下，不同時段的旅行時間會因路況而不同
G. 允許共乘比率：0%, 20%, 40%, 60%, 80%, 100%
H. 每回合子代數 n=50（每一次子代的數量，也就是平行處理的執行緒線程數），迭代回合數 t=10，局部搜尋次數 m=300

排程模擬實驗之開發及運算環境設定如下：

A. 硬體配置：
   i. CPU：2*Intel(R) Xeon(R) Gold 5220S (with total 36 cores and 72 threads)
   ii. RAM：768 GB (with per process memory limit=128GB)
B. 作業系統：CentOS Linux 8.3

C. 程式開發語言：C++/C，Python 3.8.5

5.2 測資產生器－產生接近真實情況的測資

本研究為了建立接近真實情況的測資，許多重要的因素必需列入考量，包括：各時間區段的訂單分佈、訂單的去回程比例和時間、訂單的地點分佈、訂單的搭乘乘客人數、共乘的訂單數量、復康巴士車輛資訊等等。以下依序介紹這些關鍵因素。

A. 各時間區段的訂單分佈：本研究參考潘珮琪(2018)之台北市復康巴士訂單時間區段分佈圖（如圖 3），該圖從早上 6:00 統計到晚上 22:00。其中可以注意到的是：總共有三個相對尖峰時段，分別出現在 8-9 時、11-13 時和 15-16 時。按照各時間區段比例乘上生成訂單總數，求得該訂單總數之下每個時間區段的訂單數，使得產生的測資具有與台北市復康巴士報告中相似的訂單時間區段分布。

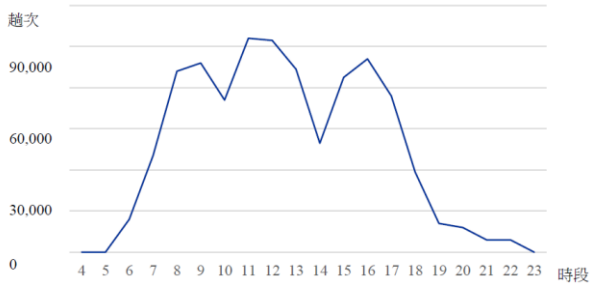

圖3 台北市復康巴士訂單時間區段分佈圖。潘珮琪(2018)

B. 訂單的去回程比例和時間（如圖 4）：本研究初期因不確定去程訂單與回程訂單的時間分佈，因此在產生測資時以隨機亂數的方式生成去程訂單的時間區段。透過本研究團隊與台北市復康巴士業者討論後，了解到使用者的訂單若有去程通常會有回程，因此，若能取得用戶的去程與回程的時間間隔，就能夠產生更真實的訂單時間分佈。雖然在陳其華等人(2019)的研究中，台北市復康巴士資料中無法判斷出同一位用戶的訂單情形，但透過台南市復康巴士的營運數據分析，發現台南市的整體訂單時間分佈與台北市類似，而且台南市復康巴士的營運數據分析中明確指出同一位乘客的訂單平均去回程時間相差約 182 分鐘（通常為客戶就醫所需花費之時間）、標準差約 89 分鐘。因此本研究透過先行產生去程訂單，再以去程訂單時間加上三小時的時間區段為中心，搜尋該時間附近的時間段分佈，若還有剩餘的時間區段，便將回程訂單的時間放在搜尋到的時段。

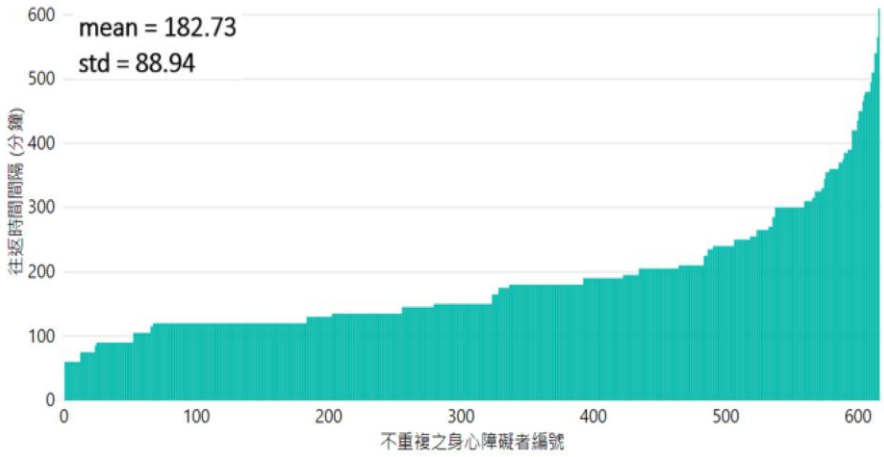

圖4 訂單去程與回程之間的時間間隔分佈圖。陳其華等人(2019)

C. 訂單的地點分佈：本團隊最初建立以台北市區內的現有醫院資料，框出各醫院的乘客範圍，矩形的邊長約莫為 4-10 公里不等。在生成訂單起迄點時，以各級醫院抽樣比例來生成所有訂單往返之醫院的分佈情形，之後在該醫院的乘客範圍內生成乘客的位置，而乘客與醫院的直線距離限制在 3.5 公里以上，將每筆訂單 OD 旅行的平均所需時間設定在 20 分鐘左右，以符合現實情況。

D. 訂單的搭乘乘客人數：本研究參考潘珮琪(2018)對台北市的復康巴士之使用情形的調查結果，特 A 等級使用率大約為 26%（如表 3），因此在產生測資時以該使用率來產生有輪椅位需求的訂單數，並且每筆訂單的乘客隨機生成規則為：

   i. 當隨機產生之訂單含有極重度身心障礙乘客，則輪椅位需求設定為 1，一般位需求設定為 0 或 1（家屬陪同）。

   ii. 當隨機產生之訂單不含極重度身心障礙乘客，則輪椅位需求設定為 0，一般位需求設定為 1 或 2（身心障礙乘客+家屬陪同）。

表3 台北市復康巴士使用人數統計

障礙等級	使用率
特 A 等級	26.52%
A1 等級	9.15%
A2 等級	29.24%
B 等級	35.10%

E. 共乘的訂單數量：以隨機亂數的方式生成要共乘的訂單，而測資生成程式允許使用者調整參數來生成對應共乘比率的測資。

F. 復康巴士車輛測資：本研究參考張朝能等人(2018)，將各車輛的輪椅位數設為 1，一般位數設為 4。

本研究開發之測資產生器可透過修改檔案的方式產生不同參數下的測資，包含乘客範圍、訂單時間分佈、訂單數量和共乘比例，以此為基礎建立各種參數下的測資進行實驗比較。

5.3 產製旅行時間矩陣

為了獲得 OD 所需旅行時間，本研究需產製所需的旅行時間矩陣（Time Matrix），最簡單的作法是對所有訂單的起迄點及場站建立點到點的旅行時間矩陣，然而，本研究之模擬所需的排程訂單量約 1000 筆，每筆訂單都需要對來回時間的距離進行計算，共需要計算約四百萬次，如果使用 Google Maps API 發送請求所耗費的時間以及金錢成本是非常可觀的，就算使用開源的 OSRM Engine（Open Source Routing Machine Engine，OSRM(2011)）也需要四至六小時才能產生所需要的時間矩陣，嚴重影響排程的執行效率。為了解決大量耗費時間及資源的問題，本研究使用了採樣的概念，在台北市範圍內的區域標示許多的樣本點（如圖 5），記錄所有樣本點之間的旅行時間，排程再針對訂單的起終點位置選取直線距離最近的樣本點，透過查表的方式即可查找最相近的時間距離，再根據與樣本點之間的距離加上少量時間差即可。透過上述方法，只需在初始時建立均勻樣本的旅行時間矩陣，未來使用上就不需每一次排程都對 OSRM Engine 發送大量請求。

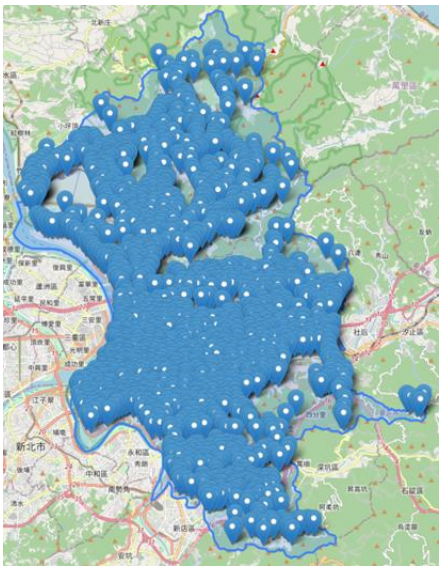

圖5 台北市均勻採樣點

本研究使用了台北市密度相當高的公車站牌以及 YouBike 站點當作樣本點，總共約 4 千個點，從這 4 千點中將兩點直線距離小於 55 公尺的其中一點刪除，使均勻撒點的個數不會過多，經過整理後剩下約 2000 點。

在計算兩點之間時間距離之前，先在 Linux 下使用 Docker 建立 OSRM Engine 後端。OSRM Engine 是款開放原始碼的地圖引擎，可安裝在伺服器上以便對其發送請求，以下為請求範例：

http://127.0.0.1:5000/route/v1/driving/13.388860,52.517037;13.385983,52.496891?steps=false

透過填入兩點的經緯度座標（如圖 6），即可取得 OSRM Engine 回傳的旅行時間，把均勻撒點的所有座標點當作參數來發送請求，即可產生出所有點到點的旅行時間矩陣。然而 OSRM 不支援即時旅行時間查詢，無法根據訂單的出發時間取得準確的旅行時間，為了要解決這個問題，本研究另外建立一個陣列來儲存不同時段的旅行時間倍率，針對同一組出發時間（例如：10 時到 11 時），透過大量採樣 Google Map 與 OSRM Engine 的旅行時間，取得大量相同路徑的時間倍率之後，再透過平均可獲得此陣列。

{"code":"Ok","routes":[{"geometry":"gaysC{xcpU??","legs":[{"steps":
[],"distance":0,"duration":0,"summary":"","weight":0}],"distance":0,"duration":0,"weight_name":"routability","weight":0}],"waypoints":[{"hint":"FoUDgB-
FA4ANAAAAAAAAAF8BAAAAAAAAqg5lQAAAAAA3vsJCAAAAAA0AAAAAAAAAXwEAAAAAAADcAgCAJ-AKB2oJdAE8TMwArVghAwoA3xIrKNO-","distance":9666967.198573,"name":"","location":[118.153255,24.381802]},
{"hint":"FoUDgB-FA4ANAAAAAAAAAF8BAAAAAAAAqg5lQAAAAAA3vsJCAAAAAA0AAAAAAAAAXwEAAAAAADcAgCAJ-AKB2oJdAH_QMwA-wkhAwoA3xIrKNO-","distance":9666482.75325,"name":"","location":
[118.153255,24.381802]}]}

圖6 OSRM Response 範例

5.4 實驗結果

本節說明本研究平行化演算法的實驗結果，由於產生測資的程式和排程程式皆有許多參數可以調整，可藉由改變各種不同參數並展示其計算結果來進行實驗，其中改變的參數包括：程式執行的執行緒、初始解的修改次數限制、訂單總數與允許共乘比例。

首先，在小型測資（(車輛, 訂單)=(20 車, 200 單)及(22 車, 220 單)）的情況下，本研究設計的排程程式皆可找到最佳解，而且求得排程結果之耗費時間遠低於 Gurobi，詳細的結果如圖7所示，當(車輛, 訂單)=(20 車, 200 單)及(22 車, 220 單)，允許共乘比例為 0%，n=50，t=10，m=300，本研究所開發的排程程式能夠接完全部訂單，並有較低的執行時間。此外，當 Gurobi 於車輛數 30 車及訂單 300 單時，無法在 90 分內求解。

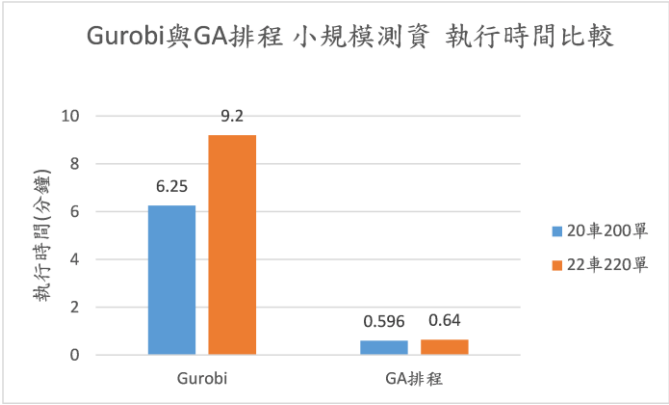

圖7 本研究平行化演算法在小型測資下的效能

當 n=50（每一次子代的數量，也就是平行處理的執行緒線程數），做 t=10 回合（共產生 10 次子代），搭配 m=300 個局部搜尋修改次數，當車輛數為 100 及訂單數為 1000 時，大約僅需要 9 分鐘，而當訂單數到達 1600 的時候，計算時間大約是 22 分鐘。此外，增加 n、t 和 m 可以幫助找到更好的答案，但是計算所需要的時間也越久，因此要謹慎設定執行緒個數、子代數和每組基因修改答案次數之限制。

本研究也觀察了訂單總數和最後接單數的關係（如圖 8），我們模擬了車輛數為 100，訂單數為 1000-1600，允許共乘比例為 100%，n=50，t=10，m=300。可以看到 1000-1200 張訂單的情況下，仍可全部接完訂單，當訂單數越多，則有機會接到更多的訂單，但也因為車輛數有限制，同時增加了棄單的風險。此外，本研究所開發的程式之收斂速度很快，如圖 9所示，當車輛數和訂單數為 100 輛與 1600 單之下，允許共乘比例為 100%，n=50，t=10，m=300，平均 3 回合即可收斂，在 1600 張訂單下，可以接完 1523 張訂單。

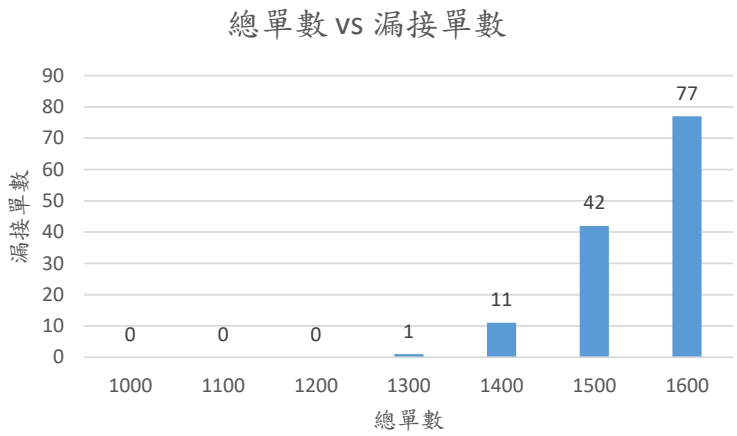

圖8 總單數對漏接單數之影響

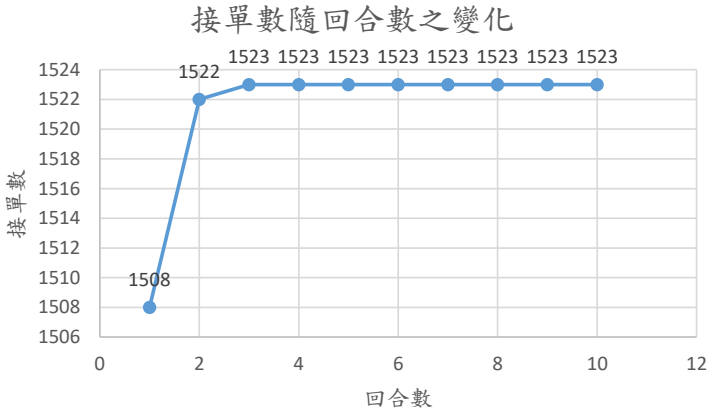

圖9 本研究平行化演算法的收斂速度

最後，我們觀察當允許共乘比例改變的時候會對接到的訂單數所造成的影響，結果如圖10所示，相當符合直覺。當允許共乘的比例下降時，本研究設計的演算法開始有機會漏掉訂單，這是因為不允許共乘的情況下，會造成在訂單即使有座位的情況下仍然不願意和別人共享，形成浪費座位的狀況。然而，本研究的演算法在客戶不願意共乘的情況下，仍然有不錯的接單率，當車輛數和訂單數為 100 車與 1300 單之下，允許共乘比例為 100%-0%，n=50，t=10，m=300，可以看到當允許共乘比例 0% 時，仍然可以接完 1277 張訂單（漏接單數=23）。

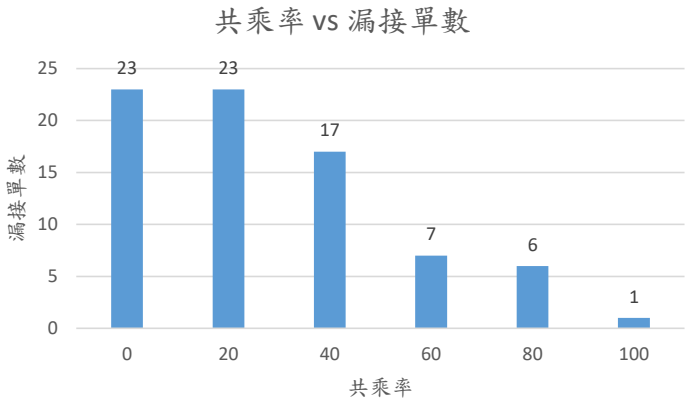

圖10 共乘率對漏接單數之影響

六、結論

本研究回顧了大量與復康巴士相關之排程文獻，整理整比較了各文獻的目標及解題方法。過去文獻中的解題情境，與本研究所要解決之題目仍存在著一定的落差，不僅測資規模較小、解題條件較少，並且題目大多為單一目標之最佳化問題。本研究首先釐清目標與條件限制，重新擬定題目模型，將題目定義為多目標組合最佳化之問題，透過參照過往文獻所使用的啟發式演算法與解題技巧，本研究使用了貪婪演算法、基因演算法、局部搜尋做組合進行演算法的設計與改良，同時採用平行處理的架構以加速程式的執行效率，用以解決本研究提出的復康巴士最佳化問題，也得到較佳的排程結果。

本研究提出的平行化演算法，在小型測資求解時與 Gurobi 具有相同品質的解答，但執行時間大幅縮短，對於大規模的訂單及車輛，演算法透過平行計算大幅度提高解題效率，同時調整接單之公平性，仍可在短時間內得到解答，排程結果亦滿足業者期待一台車接 9-10 單的品質，甚至可達到每台車平均接 12 單以上的成果。

本研究仍有部分未盡完善之處，首先是測試時所使用的測資仍不是真實測資，我們僅能依據文獻中所提供的統計分佈以測資產生器建立接近於真實測資的資料進行測試，所以在實務應用上能否達成本研究所期望的效能仍有待驗證。倘若未來可透過與復康巴士業者合作取得實際運營資料，將業者資料正規化並介接業者所提供的真實資料，便能進行資料分析以觀察資料的特性，進一步可搭配機器學習、強化式學習等方法，更精準地預知訂單時空分布，再根據資料特性進行演算法改良，以提升派車效益、減少車輛空車等待時間，並提升車輛稼動率，也可精進測資產生器以產生更實際的測試資料。此外，演算法亦可根據業者在實務應用上所面臨的需求，修改目標式以達到業者最理想情況之排程結果，加強本研究成果在實務應用上之成效。

參考文獻

王春鎰 (2007),「混合演化巨集啟發式解法應用於具時間窗車輛路線問題之研究」,國立交通大學運輸科技與管理學系碩士論文。

白峻安 (2015),「利用類電磁演算法求解於需求反應式撥召問題－以新北市復康巴士為例」,淡江大學運輸管理學系碩士論文。

交通部 (2020),「汽車運輸業管理規則」,法務部全國法規資料庫,網站:https://law.moj.gov.tw/LawClass/LawAll.aspx?pcode=K0040003。

辛孟鑫 (2007),「撥召運輸系統路線規劃問題之研究－以台北市復康巴士為例」,國立成功大學交通管理學系碩士論文。

李美儀 (2015),「車輛路線相關問題之回顧與國內發展之分析」,國立交通大學運輸與物流管理學系碩士論文。

李泰琳、張靖 (2010),「調適型導引螞蟻演算法求解時窗收卸貨問題之研究」,運輸計劃季刊,第 39 卷第 1 期,頁 99-132。

張朝能、史習平、張學孔、洪鈞澤、周文生、沈大維、林昶禛、黃哲勳、顏志平、郭佩棻、李亞君、柯旻嬋、彭怡萍、李孟衡、胡閔雁、廖洵顏、夏明達 (2018),「預約式無障礙小客車運輸服務之整合研究 (1/2)」,交通運輸研究所。

陳其華、張朝能、史習平、張學孔、洪鈞澤、周文生、李孟峰、柯旻嬋、黃哲勳、彭怡萍、黃建勛、洪勝宇、謝侑勳、林昱賢、傅宣維、吳宗叡、劉昭廷 (2019),「預約式無障礙小客車運輸服務之整合研究 (2/2)」,交通運輸研究所。

陳惠國、林奕隆、王宣 (2013),「應用修正式蜂群最佳化演算法求解撥召問題－以復康巴士問題為例」,運輸學刊,第 25 卷第 3 期,頁 279-308。

郭佳林 (2004),「求解具時間窗之多趟次車輛途程問題」,國立交通大學運輸科技與管理學系碩士論文。

許正良、謝益智 (2012),「免疫演算法於自動販賣機貨物配送路線規劃問題之探討」,2012 彰雲嘉大學校院聯盟學術研討會。

游珮暄 (2013),「復康巴士場站區位之最佳化研究」,國立台灣大學土木工程學研究所碩士論文。

黃漢瑄 (2004)，「撥召服務最佳化指派作業之研究」，淡江大學運輸管理學系碩士論文。

潘珮琪 (2018)，「由使用者觀點探討臺北市復康巴士服務成效」，國立臺灣大學政治學研究所碩士論文。

賴敬棠 (2010)，「螞蟻記憶系統應用於車輛旅途問題」，逢甲大學交通工程與管理所碩士論文。

AMPL (1984), A Mathematical Programming Language, website: https://ampl.com.

Arun K. K., Shivani V., Topon P., and Takufumi Y. (2019), "RL SolVeR Pro: Reinforcement Learning for Solving Vehicle Routing Problem," *1st International Conference on Artificial Intelligence and Data Sciences (AiDAS).*

Brahim R., Alain D., and Hussain A. S. (2006), "Handicapped Person Transportation: An application of the Grouping Genetic Algorithm," *Engineering Applications of Artificial Intelligence,* Vol. 19, No. 5, pp 511-520.

Dantzig G.B. and John H. R. (1959), "The Truck Dispatching Problem," *Management Science,* Vol. 6, No. 1, pp. 80-91.

Dominik K. and Roberto W. C. (2013), "A Granular Tabu Search Algorithm for the Dial-a-Ride Problem," *Transportation Research Part B: Methodological,* Vol. 56, pp.120-135.

Emanuel M., Ahmet B. I., and Hokey M. (2007), "A Dial-a-ride Problem for Client Transportation in A Health-Care Organization," *Computers & Operations Research*, Vol. 34, No. 3, pp. 742-759.

Frank H. C., John J. J., and H. Donald R. (1981) "Set Partitioning based Heuristics for Interactive Routing," *Networks,* Vol. 11, No. 2, pp. 125-143.

Google (2014), Google OR-Tools, website: https://developers.google.com/optimization.

Guo J. and Liu C. (2014), "A Spatio-temporal Based Parallel Insertion Algorithm for Solving Dial-A-Ride Problem," *The 26th Chinese Control and Decision Conference (CCDC).*

Gurobi (2008), The Gurobi Optimizer, website: https://www.gurobi.com.

Haibing Li and Andrew Lim (2003), "Local Search with Annealing-Like Restarts to Solve the VRPTW," *European Journal of Operational Research,* Vol. 150, No. 1, pp. 115-127.

Jean-F. C. and Gilbert L. (2003), "A Tabu Search Heuristic for the Static Multi-vehicle Dial-a-ride Problem," *Transportation Research Part B: Methodological*, Vol. 37, No. 6, pp. 579-594.

Jih W. and Hsu Y. (2004), "A Family Competition Genetic Algorithm for the Pickup and Delivery Problems with Time Window," *Bull. Coll. Eng. N.T.U.*, Vol. 90, pp. 89-98.

Liao X.-L., Chien C.-H., and Ting C.-K. (2014), "A Genetic Algorithm for the Minimum Latency Pickup and Delivery Problem," *2014 IEEE Congress on Evolutionary Computation (CEC)*.

Michel G., Jenny N., and Erwin P. (2015), "Mathematical Formulations for a 1-full-truckload pickup-and-delivery Problem," *European Journal of Operational Research*, Vol. 242, No. 3, pp. 1008-1016.

Mohammadreza N., Afshin O., Martin T., and Lawrence V. S. (2018), "Reinforcement Learning for Solving the Vehicle Routing Problem," *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*.

Nasser A. E.-S. (2010), "Vehicle Routing with Time Windows: An Overview of Exact, Heuristic and Metaheuristic Methods," *Journal of King Saud University (Science)*, Vol. 22, No. 3, pp. 123-131.

Nigel H. M. W., Richard W. W., and John R. H. (1976), "Advanced Dial-a-ride Algorithms Research Project: Final Report," Dept. of Materials Science and Engineering, Massachusetts Institute of Technology.

Osaba E., Onieva E., Diaz F., Carballedo R., Lopez P., and Perallos A. (2015), "An Asymmetric Multiple Traveling Salesman Problem with Backhauls to Solve a Dial-a-Ride Problem," *IEEE 13th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*.

OSRM (2011), Open Source Routing Machine, website: http://project-osrm.org.

Sophie N. P., Karl F. D., and Richard F. H. (2010), "Variable Neighborhood Search for the Dial-a-ride problem," *Computers & Operations Research*, Vol. 37, No. 6, pp. 1129-1138.

Zhihao P., Zaher C., Hervé M., and Marie-A. M. (2018), "A Particle Swarm Optimization for Selective Pickup and Delivery Problem," *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*.