國　立　交　通　大　學
運　輸　與　物　流　管　理　學　系

博 士 論 文

No. 022

針對分割配送相關車輛路線問題特性設計之多重起點啟發式方法與其應用

A Multi-start Heuristic with a Problem-specific Operator for the Split Delivery Vehicle Routing Problem and Its Variants

研 究 生：朱佑旌

指導教授：韓復華　教授
卓裕仁　副教授

中　華　民　國　一〇六　年　七　月

針對分割配送相關車輛路線問題特性設計之多重起點啟發式方法與其應用

A Multi-start Heuristic with a Problem-specific Operator for the Split Delivery Vehicle Routing Problem and Its Variants

研 究 生：朱佑旌　　Student: Yu-Ching Chu

指導教授：韓復華　　Advisor: Anthony Fu-Wha Han

卓裕仁　　Yuh-Jen Cho

國 立 交 通 大 學

運 輸 與 物 流 管 理 學 系

博 士 論 文

A Dissertation

Submitted to Department of Transportation & Logistics Management

College of Management

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy

in

Traffic and Transportation

July 2017

Hsinchu, Taiwan, Republic of China

中　華　民　國　一〇六　年　七　月

針對分割配送相關車輛路線問題特性設計之多重起點啟發式方法與其應用

研究生：朱佑旌　　指導教授：韓復華 教授
卓裕仁 副教授

國立交通大學運輸與物流管理學系博士班

摘　要

分割配送車輛路線問題 (Split Delivery Vehicle Routing Problem, SDVRP) 允許單一顧客的需求可分批被多部車輛服務;它是傳統車輛路線問題的衍生問題。由於配送限制的放鬆，SDVRP 具有降低運輸成本的效益，進可提昇企業之競爭力。但就實務而言，需求分割配送會造成顧客的不便，亦伴隨作業成本的增加。有鑑於此，如何使 SDVRP 兼顧成本效率與服務品質是一重要議題。近年已有文獻 (Gulczynski *et al.* [29]) 將最小配送量列入考慮，提出最小配送量限制之分割配送車輛路線問題 (SDVRP with Minimum Delivery Amounts, SDVRR-MDA)。本研究則首次提出以配送次數作為限制的「具配送次數限制的分割配送車輛路線問題」(SDVRP with Limited Number of Deliveries, SDVRP-LND)，以期分割配送能有更高的應用價值。

針對分割問題特性，本研究提出一套一體適用於求解 SDVRP，SDVRP-MDA 與 SDVRP-LND 的多重起點啟發式解法，稱為 SRC+IMP (Split-delivery Route Construction + solution IMProvement)。SRC 為起始解構建模組，適應性地選擇「插入路線」或是「新增路線」的方式逐點構建起始路線。IMP 為多鄰域求解改善模組，其中包含一套針對分割問題特性新設計的驅逐鏈鄰域搜尋法。整個演算法 SRC+IMP 則採用多重起點策略整合 SRC 與 IMP 以兼顧搜尋的深度與廣度。

SRC+IMP 演算法以目前所有 SDVRP 相關之國際標竿題庫進行測試，在 IMP 模組部分採用變動鄰域下降 (Variable Neighborhood Decent,VND) 與隨機變動鄰域下降 (Randomized VND, RVND) 兩種方式執行。測試結果發現 SRC+RVND 的績效較 SRC+VND 佳；其結果在 SDVRP 方面，對 32 個標竿題求解的平均誤差為 0.36%，其績效僅次於 ILS (Silva *et al.* [44])；在 SDVRP-MDA 方面，對 128 個標竿題求解的平均誤差為-0.27%，且求得 81 題並突破 36 題已知最佳解；在 SDVRP-LND 方面，22 題標竿例題中則求得 5 題已知最佳解並突破 5 題，平均求解誤差為 0.07%。

本研究亦分析配送次數上限 *kmax* 對成本的影響，結果發現當 $kmax = 3$ 時，SDVRP-LND 的成本節省效益為 4.11%，其與無次數限制的 SDVRP 相差僅 0.06%；這顯示以最多配送三次執行 SDVRP-LND 可對物流界提供相當的實用價值。

關鍵詞：車輛路線問題、分割配送、最小配送量、配送次數限制、多重起點、變動鄰域下降、節點驅逐鏈

A Multi-start Heuristic with a Problem-specific Operator for the Split Delivery Vehicle Routing Problem and Its Variants

Student: Yu-Ching Chu

Advisor: Dr. Anthony Fu-Wha Han
Dr. Yuh-Jen Cho

Department of Transportation and Logistics Management National Chiao Tung University

ABSTRACT

The split-delivery vehicle routing problem (SDVRP), in which the demand of a customer can be split and delivered by several vehicles, would result in less cost than the conventional vehicle routing problem. However, split deliveries also incur extra work for logistics operations and undesirable distractions to customers. How to tradeoff cost efficiency and customer service in the split-delivery related problems thus is an important issue. Recently, Gulczynski *et al*. [29] proposed a variant of the SDVRP called the SDVRP-MDA, in which the customers each require a minimum delivery amount (MDA) every time they are visited. In this dissertation, we propose a new variant of the SDVRP, i.e., the spit-delivery VRP with limited number of deliveries (SDVRP-LND), which considers the number of deliveries for each customer is restrained to *kmax*.

We propose a unified multi-start heuristic method, i.e., SRC+IMP (Split-delivery Route Construction + solution IMProvement), to solve the SDVRP and its variants of the SDVRP-MDA and the SDVRP-LND. The initial-solution generator SRC adaptively applies either node-insertion or route-addition procedures, which were designed specifically for the SDVRP and its variants, to generate an initial solution. The IMP conducts local search with multiple neighborhoods for solution improvement, in which we developed a novel node ejection-chain operator for the split-delivery related problems. The overall SRC+IMP combines the SRC and the IMP modules to implement a multi-start solution procedure with a single parameter to control the restart.

The proposed SRC+IMP algorithms are tested with all the related benchmark problems. We conduct the IMP module with two variable neighborhood descent methods, i.e., VND and RVND. It is found that the performance of the SRC+RVND is better than that of the SRC+VND. The results of the SRC+RVND for the SDVRP show that, out of 32 instances tested, the average deviation from the best known solutions (BKS) is 0.36%, which is only second to the ILS algorithm (Silva *et al*. [44]). For the SDVRP-MDA, out of the 128 instances tested, the results reveal 81 BKS and 36 new best solutions; the average deviation is -0.27%. For the SDVRP-LND, out of the 22 instances tested, we find 5 BKS and 5 new best solutions; the average deviation is 0.07%.

We also investigate the impact of *kmax* to the potential cost saving of the SDVRP-LND. It is found that, for *kmax*=3, the average cost saving is 4.11%, which is only 0.06% less than that of the SDVRP. This implies that the SDVRP-LND with *kmax*=3 can provide a valuable and easy-to-implement tool for logistics operations.

Keywords: Vehicle routing, Split delivery, Minimal delivery amount, Limited number of delivery, Multi-start, Variable neighborhood descent, Node ejection chain

誌　謝

隨著時光的流逝，博士班修業的生涯即將在這個夏天結束。在這過程中，我得到許多人的幫助以及支持，也才能夠完成博士學位。首先，我必須要感謝的是兩位亦師亦友指導老師　韓復華教授以及卓裕仁教授。多年來，兩位恩師在學生的研究想法上提供許多寶貴的建議，並在研究的過程也不斷地給予鼓勵以及支持。真的非常感謝您們對學生所付出的心力，帶領著學生走過無數的障礙與迷霧，今天才能夠幸運地完成學位論文。學生在此謹致上最誠摯的敬意與感恩。

論文口試審查期間，感謝口試委員元智大學丁慶榮教授、中華大學蘇昭銘教授、本校卓訓榮教授及張宗勝教授願意於百忙之中撥冗審閱學生的論文，並不吝給予諸多寶貴的意見及指正，使得論文能夠更詳盡完善。

在交大的學生生活即將結束，我特別感謝東吳大學賈凱傑教授與胡凱傑教授，由於您們的鼓勵讓我對於完成學位的目標能夠更堅定。感謝研究室中政威學長以及俊德學長，在我每次面臨低潮的時候，你們總是會適時地關心我並伸出援手給予協助。亦特別謝謝明穎學長，每當沮喪時第一時間總是想到您，謝謝您給予我無限振作的動力。此外，感謝在這段時間有緣相遇的學弟妹們，謝謝你們的陪伴，感謝你們的協助。若是沒有以上的各位，我想我早已被擊敗，無法走到終點。

最後要感謝一路上陪伴的父母親。從小到大，我想我並不是一個人人眼中的所認為的聽話的孩子。雖然我有段叛逆的過往，感謝您們從未對我有過放棄的念頭。感謝培仁，讓遠離家鄉的我不擔心家裡的情況，能夠安心在他鄉奮鬥學業。感謝雅雯，謝謝你這些年來的付出以及陪伴，這段時間真的辛苦你也委屈你了。因為你們，「家」永遠是那樣的溫暖。感謝你們在過去這段時間的全力支持與鼓勵，我才能夠堅持到最後取得博士學位。僅將此論文獻給我摯愛的家人。

記得在入學之初，韓老師曾對學生說過：「求學有如化蛹成蝶的過程，蝴蝶雖然美麗，但破繭的過程是十分辛苦的」。雖然過程辛苦，但因為諸位的陪伴，我才能夠順利地完成。感謝過去曾經幫助過我的所有人，你們著實豐富了我的人生。謹將這份榮耀與喜悅，與你們分享！

朱佑旌　謹誌

于 交大網路實驗室 2017.08

目　錄

圖目錄

表目錄

第一章、前言

1.1 研究動機與目的

車輛路線問題 (Vehicle Routing Problem, VRP) 首先由 Dantzig and Ramser [21] 於 1959 年提出。該問題是考慮單一型態的車輛如何以最小成本 (旅行距離) 由單一場站出發滿足多個顧客點取貨或送貨的服務，再返回該場站。在傳統 VRP 的限制下，每一個顧客點需求的服務僅能夠由單獨一部車輛路線進行服務。隨著物流型態的演變，配送實務開始出現多元的配送需求以及作業條件。為了滿足這些不同的要求，車輛路線問題亦逐漸產生多種不同的衍生類型，例如：時間窗、多車種、回程取貨、委外等不同作業特性。一般將這些車輛路線相關問題稱為 VRPs 或是 VRRP (vehicle routing related problem)。

在 VRP 相關問題中，除關注於物流業者到最終消費者這一段的「外部配送作業」以外，亦有適合物流業者「內部物流作業」的問題型態。所謂內部物流是指物品從集貨營業站所至發貨營業站所的物流配送作業。這部分的物流作業特性為配送迄點的需求量相對較大，以致單車次配送無法滿足，需要利用多部車輛共同進行服務，需求必然發生分割。需求分割車輛路線問題 (Split Delivery Vehicle Routing Problem, SDVRP) 便是因應內部物流的配送特性產生。

SDVRP 是 VRP 放鬆「顧客需求不可分割」條件所衍生的問題。由於條件放鬆，在其他條件相同下，SDVRP 必可求得較 VRP 更小成本，或至少相同成本的解答。最早於 1989 年，Dror and Trudeau [23] 便已開始探討 SDVRP 對於 VRP 在成本上節省的潛在效益。Archetti *et al.* [9] 於 2006 年證明在允許需求分割的的情況下進行配送，最多能夠節省 50%的距離成本。此外，Archetti *et al.* [8] 於 2008 年證明 SDVRP 也可以節省 VRP 至多一半的車輛數。因此，SDVRP 在節省成本的潛在效益是相當可觀的。

圖 1.1 示意 SDVRP 的節省效益，假設顧客 1、2 以及 3 的需求分別為 7、35、95，車容量為 100 時。圖 1.1 (a) 顯示 VRP 的求解結果，由於車容量限制使得每輛車只能服務一個顧客，因此完成所有服務需使用 3 輛車，總旅行距離為 48。然而，SDVRP 求解時允許需求可分割配送，僅需要 2 部車輛便能夠服務完所有需求，並且總旅行距離也降低為 38。由此簡例便能夠了解到允許顧客需求在配送時分割的益處。

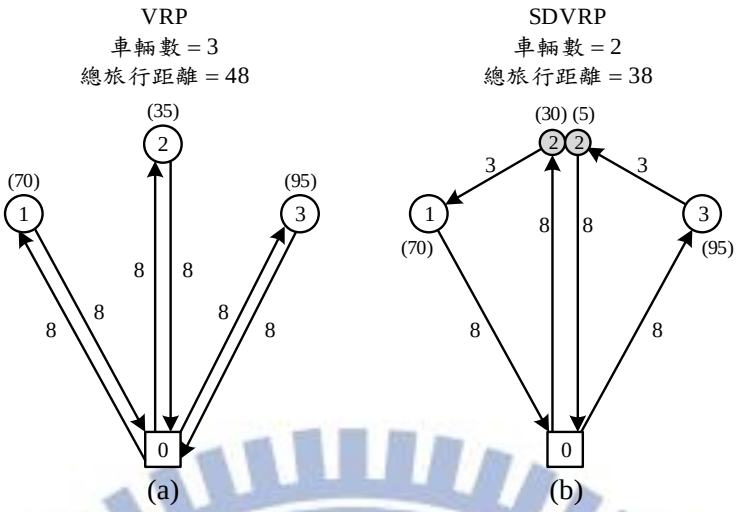

圖 1.1　VRP 與 SDVRP 比較圖 (灰色節點表示分割配送顧客)

資料來源：Aleman *et al.* [4]

目前，SDVRP 相關問題除 Dror and Trudeau [23] 提出的 SDVRP 外，另有 2010 年 Gulczynski *et al.* [29] 提出的 SDVRP-MDA (SDVRP with Minimum Delivery Amounts)。SDVRP-MDA 是考慮在分割配送的條件下，單一路線對單一顧客 i 的配送量應給予適當的最小配送量限制 MDA_i。Gulczynski *et al.* [29] 明確指出允許分送雖然能夠降低配送作業的運輸成本，但對於負責運輸的一方或是收貨方則會產生其他的成本，包含紙本作業、時間消耗、資料傳輸以及收貨作業等成本。此時，為了能夠平衡節省的效益以及額外增加的成本，雙方會利用「最小配送量」或是「最小配送金額」的協定來彌補收送貨所產生的成本。這種協定在實務上是存在的，例如國內最大的連鎖超商 7-Eleven [1] 以及英國 Tesco [47] 便有購買咖啡滿 300 新台幣以及消費滿 40 英鎊的條件始可提供外送的服務。

Gulczynski *et al.* [29] 利用最小配送比例 p 值設定 MDA 的大小，進一步限制每次配送顧客 i 的配送量必須大於等於其總需求 d_i 的 p 比例，因此顧客 i 的最小配送量可計算為 $MDA_i=\lceil d_i \times p \rceil$。圖 1.2 為 SDVRP 與 SDVRP-MDA 的比較圖，其中圖 1.2a 為 SDVRP 的路線配置，旅行成本為 38 (同圖 1.1b)；右側為 $p = 0.2$ 的 SDVRP-MDA 的路線配置，旅行成本為 43。如圖 1.2 所示，由於 SDVRP-MDA 的限制，原 SDVRP 對顧客 2 的分割是不可行的($5 < MDA_2= \lceil 35 \times 0.2 \rceil = 7$)。此時，為符合 MDA 限制，分割顧客 3 是較好且可行的方案 ($MDA_3 = \lceil 95 \times 0.2 \rceil = 19$)。

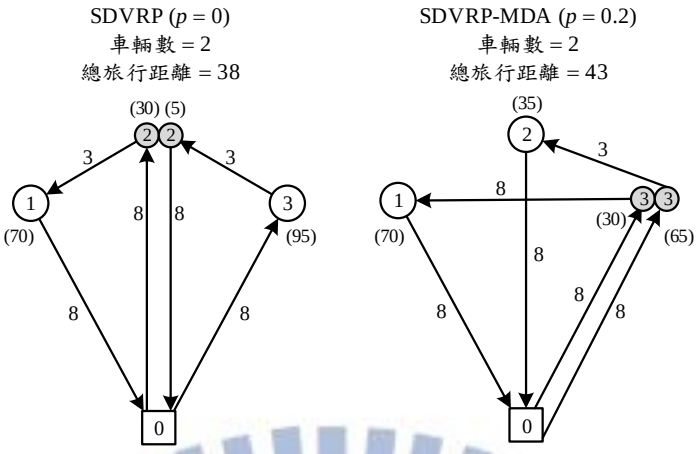

圖 1.2　SDVRP 與 SDVRP-MDA 比較圖 (灰色節點表示分割配送顧客)

本研究將需求分送的議題實際與國內物流業者進行討論，業者受訪談時提及分割配送的確能夠有效降低運輸成本，但亦須考慮收貨端驗收貨物的作業成本。並說明收貨端主要的作業成本受到收貨次數的影響，認為分割配送的次數應給予適當的限制。因此本研究提出一個新形態的需求分割車輛路線問題，稱之為「具配送次數限制分割配送車輛路線問題」 (SDVRP with Limited Number of Delivery, SDVRP-LND)。

需求分割配送雖會造成顧客的不便，但其節省運輸成本的潛在效益是值得公司企業或供應鏈集團的內部物流作業考慮應用的。需求分割的配送是實際作業上面對的課題，若能有效處理，便可提昇該企業之競爭力。SDVRP 相關課題之研究雖可追溯至 1989 年 [23]，但直至近 10 年才開始在國際文獻中受到重視與廣泛探討 (相關之文獻將於第二章回顧)。反觀國內對此課題之研究，目前還尚不多見。故本研究欲針對分割配送問題的特性設計一套能夠一體適用於多種不同分割配送車輛路線相關問題，且兼具求解效率與效果的求解方法架構。

1.2　研究流程與步驟

本研究之主題分割配送車輛路線問題以及其衍生問題皆屬於傳統 VRP 問題的衍生問題，在問題的複雜度上已證明屬於 *NP*-Hard 的問題 [24]。由於問題的高複雜度，使得目前的精確解法在求得最佳解的時間上，會隨著問題的規模呈現指數成長，即無法多項式時間 (Polynomial Time) 內求得。為了提高問題在實務應用上的可行性，啟發式解法是十分適合應用的求解方法。啟發式解法屬於近似解法，目前已被大量應用於求解高度複雜的問題上。其優點是能夠在可接受的時間範圍內，針對問題的限制求得一個品質優良的可行解。為求方法在實務應用上的可行性，本研究即以啟發式解法作為研究方法。整體的研究流程如圖 1.3 所示，相關研究步驟簡要說明如下：

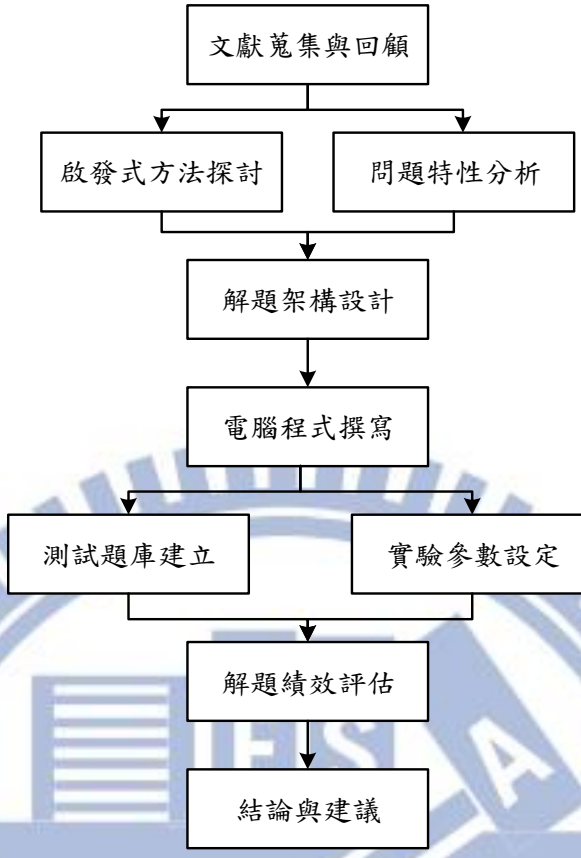

圖 1.3　研究流程圖

1. 文獻蒐集與回顧：蒐集 SDVRP 與其衍生問題的相關文獻，了解研究主題目前發展的情況。
2. 啟發式方法探討：回顧相關傳統 VRP 啟發式方法目前發展的情況，並特別針對應用於需求分割車輛路線問題特有啟發式方法進行回顧以及探討。
3. 問題特性分析：　分析分割配送問題之特性，了解適合產生分割的情況以及方式。
4. 解題架構設計：　根據前述相關回顧，將所得之結論彙整進而設計本研究之核心求解演算法細部流程。
5. 電腦程式撰寫：　將所設計之演算法利用電腦語言進行撰寫編譯，以利後續方法求解的績效測試。
6. 實驗參數設定：　對演算法所需之參數進行測試，決定參數值設定。
7. 標竿題庫建立：　蒐集 SDVRP 與其衍生問題的相關測試題庫，建立評估演算法求解績效的標竿題庫。
8. 解題績效評估：　彙整本研究所提出之演算法求解結果，並與其他相關文獻以及已知最佳解進行比較，以評估演算法之求解績效。
9. 結論與建議：　提出本研究之發現與結論，並建議後續研究方向。

1.3 章節簡述

本研究後續章節擬組織如下：

第二章：分割配送相關車輛路線問題文獻回顧。針對分割配送相關車輛路線問題的啟發式研究進行回顧，包含文獻中方法論所提出的起始解路線構建方法、鄰域搜尋法以及巨集啟發式演算法。

第三章：分割配送相關車輛路線問題數學定義與問題性質。針對 SDVRP、SDVRP-MDA 以及 SDVRP-LND 各問題說明數學模式、整數解性質以及潛在效益。

第四章：針對本研究所提出之 SRC+IMP 多重起點啟發式演算法架構進行說明。分別敘述所設計之起始解模組、鄰域改善模組以及多重起點機制的細部流程操作。

第五章：說明所提出之演算法於分割配送相關車輛路線問題所用之測試題庫、測試環境以及演算法參數設定。

第六至八章：以 SRC+IMP 分別對於 SDVRP (第六章)、SDVRP-MDA (第七章) 以及 SDVRP-LND (第八章) 進行標竿題庫的求解測試，並將求解績效與其他文獻解法進行比較。

第九章：綜合比較分析。將本研對於分割配送相關車輛路線問題的整體求解結果進行分析比較。並探討本研究設計之求解模組對求解效果的影響以及貢獻。最後再以求解結果的差異分析實際上分割配送所能帶來的效益為何。

第十章：結論與建議。根據本研究所得之求解結果提出結論與後續研究建議。

第二章、文獻回顧

2.1 啟發式解法回顧

2.1.1 起始解路線構建方法

在啟發式方法求解 VRP 的過程中，第一步驟便是要考慮如何獲得一組起始解路線以供後續的改善流程進行優化。起始解，一般研究中稱為「s_0」，意即整個搜尋流程中的起點解。起始解的路線構建，最廣為被使用的應屬「節省法」以及「插入法」兩種啟發式方法。這兩個啟發式方法都是旅行推銷員問題 (travelling salesman problem, TSP) 基本的路線構建方法。由於方法的概念容易了解且執行簡單，所以被廣泛應用在 VRPs 的研究中。

(1) 節省法

節省法的概念最早由 Clarke and Wright [18] 於 1964 年提出，概念為利用兩兩路線的合併進行路線解的構建。節省法藉由路線合併後所產生的節省值大小，決定路線合併的先後順序。節省法執行時，首先計算任兩兩顧客點合併後產生的節省值。以圖 2.1 說明節省值的計算方式，當路線 $(0-i-0)$ 以及路線 $(0-j-0)$ 進行合併時，節省值 S_{ij} 的計算為: $S_{ij} = c_{i0} + c_{0j} - c_{ij}, i \neq j$。所有任兩點間的節省值計算完畢後，由大到小進行排序產生節省值清單。產生清單後，便開始依照清單中的順序，進行「可行」節省值的合併。所謂「可行」節省值的合併意指，產生節省值 S_{ij} 的顧客點 i 與顧客點 j 必須分屬不同路線的端點 (路線中的起點顧客或是終點顧客)，且兩路線合併後必須遵守問題的限制，例如車容量限制或路線長度限制等等。依序執行合併，直到節省值清單上無可行的節省值可以進行合併則結束程序。

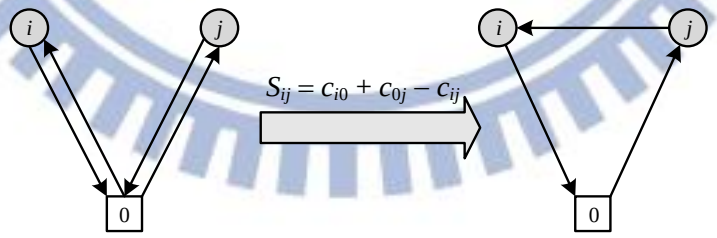

圖 2.1 節省值 S_{ij} 計算示意圖

資料來源：本研究整理

(2) 插入法

插入法在路線構建的應用最早是由 Rosenkrantz *et al.* [43] 於 1974 年提出的最近插入法 (Nearest Insertion Method)。插入法構建路線的方式是將未服務的顧客點逐一插入現行路線中的特定兩點之間。以最近插入法為例，插入的顧客是選擇目前距離路線最近的節點，並將之插入於插入成本最低的位置。插入成本的計算圖 2.2 說明，I_{ijk} 表示顧客

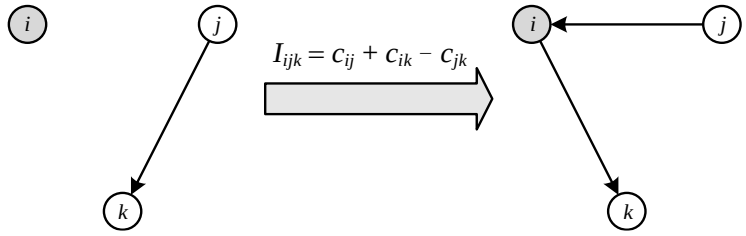

圖 2.2　插入成本 I_{ijk} 計算示意圖

資料來源：本研究整理

點 i 插入節線 (j, k) 的插入成本，插入成本 I_{ijk} 為: $I_{ijk} = c_{ji} + c_{ik} - c_{jk}$。由於插入成本的計算牽涉到三個節點之間的距離，因此插入法開始執行時，需選擇一個未服務的顧客做為種子點產生一條路線做為後續插入的現行路線。在種子點的挑選方面，通常建議以距離最遠或是需求量最大做為選擇的標準。插入法會反覆的進行選擇插入點以及插入最省位置的動作，直到所有顧客點皆被服務。

2.1.2　鄰域搜尋法

鄰域搜尋 (Neighborhood Search, NS 或是 Local Search, LS) 是指利用啟發式方法改善現行解 s 的程序。鄰域搜尋方法以固定的變化方式探索現行解 s 的鄰域解 $s' \in N_k(s)$，$N_k(\ )$ 表示所使用鄰域搜尋法 N_k 搜尋的鄰域解集合。當發現鄰域解的成本 $c(s')$ 優於現行解的成本 $c(s)$，便將現行解 s 更新為 s'。一般而言，鄰域搜尋法將反覆執行搜尋，直到沒有任何鄰域解 $s' \in N_k(s)$ 優於現行解 s 為止。在 VRPs 研究中，鄰域搜尋法根據其鄰域解的結構可分成兩大類，分別是「節線交換法 (Arc Exchange)」以及「節點交換法 (Node Exchange)」。節線交換法尋找鄰解的方式是刪除現行解若干節線後，重新進行連接成新的解;「節點交換法」則是針對現行解嘗試將一個或多個節點移動至不同的位置，以產生新的鄰域解。以下回顧若干 VRPs 研究中經常使用的鄰域搜尋法。

(1)　*k-opt* 節線交換法

k-opt 交換法是為目前 VRPs 研究中最常被採用的路線內交換法。*k-opt* 中的 k 指的是交換的節線數量，k 必須大於等於 2。*k-opt* 的交換方式是將路線中原本的 k 條不相鄰節線刪除後，再以 k 條新的節線重新連接產生新解。本研究以 2-*opt* 進行說明，2-*opt* 最早由 Croes [20] 於 1958 提出，圖 2.3 為 2-*opt* 交換法的示意圖。圖 2.3 中，兩條節線 (i, si) 以及 (j, sj) 於現行解中移除，並重新以 (i, j) 以及 (si, sj) 兩節線進行連接。特別值得注意的是，當路線完成重新連接之後，原先節點 si 到 j 中間經過的節線需要進行方向的反轉，使路線能夠保持方向性一致。

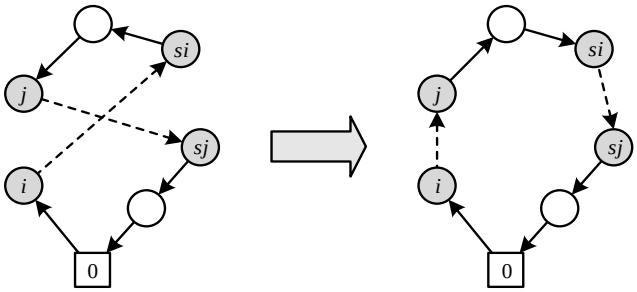

圖 2.3　2-*opt* 交換法示意圖

資料來源：本研究整理

2-*opt* 交換法的搜尋中，刪除某兩節線之後只有一種能夠避免造成子迴圈的重新連結方式，因此執行速度快。在優化效果上也顯著，是 VRPs 相關研究中最有必要應用的啟發式搜尋法之一。*k*-*opt* 的鄰域解集合隨著 *k* 值的提高也越大，雖然能夠找到品質更佳的結果，但相對也會降低搜尋的效率。因此，*k*-opt 交換法應用時，*k* 最常被設定為 2 或者是 3，也就是 2-*opt* 以及 3-*opt* [35]。Lin and Kernighan [36] 以 *k*-*opt* 為基礎，提出適應性的 *Lin-Kernighan algorithm*，該交換法的特色在於並未設定固定的 *k* 值，而是在交換過程中適應性地增加 *k* 值。意即重置的節線組數會適應性地增加，直到無法在獲得更多節省。因此 *Lin-Kernighan algorithm* 的搜尋鄰域包含 2-*opt* 以及 3-*opt*，求解精度也較佳。

(2)　*Or-opt* 節點交換法

Or-opt 交換法屬於路線內的節點交換法，為 1976 年 Or [39] 於其博士論文中提出。*Or-opt* 是將 *p* 個相連的節點移動至路線內其他位置的方式進行鄰域解搜尋，一般 *p* 值的設定以及執行的順序為 3, 2 以及 1，圖 2.4 為 *Or-opt* 交換法的示意圖。圖 2.4 (a), (b) 以及 (c) 分別示意 $p = 3, 2$ 以及 1 的移動方式。以圖 2.4 (a) 說明 *Or-opt* ($p = 3$)，將 *i*, *j* 以及 *k* 三個連續節點移動至節點 *u* 以及 *v* 之間，則圖 2.4 (a) 中右側的路線為左側路線的其中一個 *Or-opt* 鄰域解。從移動的結果來看，交換的結果皆是重置 3 條節線，因此 *Or-opt* 其實是一種的 3-*opt* 交換法的一種特例。*Or-opt* 亦由於操作上較 3-*opt* 簡易，故在應用上也較為普遍。

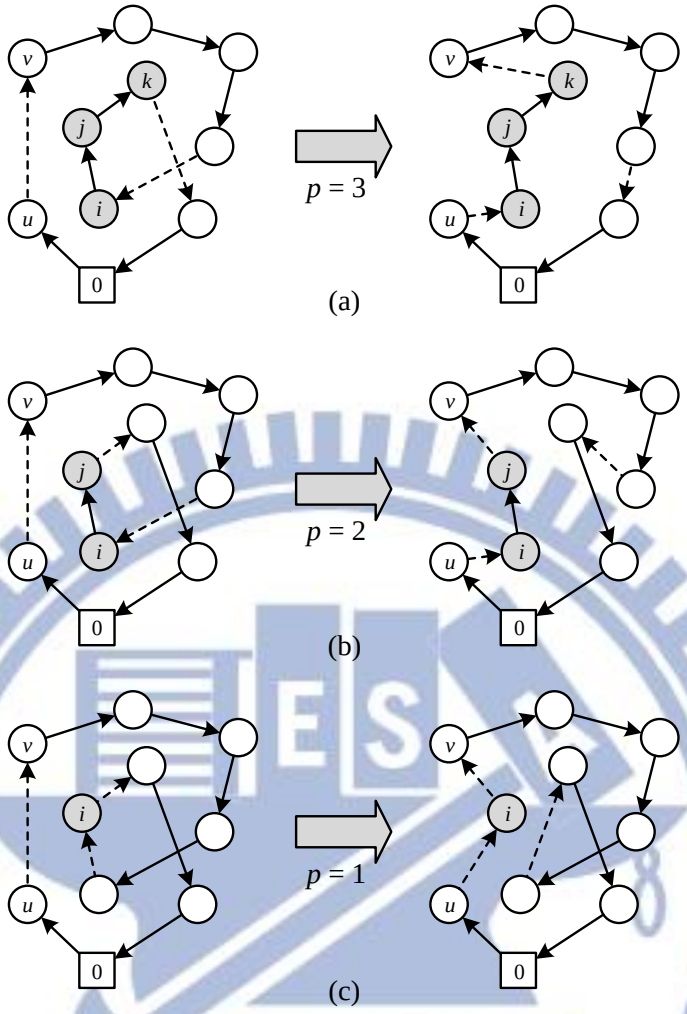

圖 2.4　*Or-opt* 交換法示意圖 (p = 1, 2, 3)

資料來源：本研究整理

(3)　2-*opt**節線交換法

Potvin and Rousseau [41] 於 1995 年首次提出 2-*opt**交換法，2-*opt**交換法依交換方式是屬於路線間節線交換法的一種。2-*opt**的概念是將原本只能應用在路線內改善的 2-*opt* 交換法延伸至路線間。2-*opt**交換法會從兩條不同的路線中各自挑選一條節線進行刪除，並以另外兩條節線進行重新連接，圖 2.5 為 2-*opt**交換法的示意圖。圖 2.5 中左側為現行解，2-*opt**交換時將節線 (i, si) 以及 (j, sj) 於現行解中移除，並重新以 (i, sj) 以及 (j, si) 兩節線進行連接產生圖 2.5 右側的鄰域解。

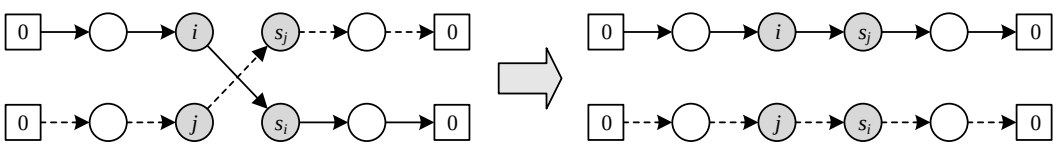

圖 2.5　2-*opt**交換法示意圖

資料來源：本研究整理

(4) *λ-interchanges* 節點交換法

λ-interchanges 節點交換法的概念最早由 Osman [40] 提出，泛指在兩條路線間交換顧客至對方路線的節點交換法。為了能夠更貼切的表達*λ-interchanges* 交換法的含意，Cordeau and Laporte [19] 於 2005 年以 (λ_1, λ_2) 表示*λ-interchanges*，意指兩條路線分別選擇λ_1與λ_2個節點交換至對方路線。λ, λ_1與λ_2的關係式為 $1 \le \lambda_1 \le \lambda$ 且 $0 \le \lambda_2 \le \lambda_1$。換言之，當$\lambda$值設為 2 便表示含有五種節點交換法，分別為(1, 0)、(1, 1)、(2, 0)、(2, 1) 以及 (2, 2)。其中依據節點移位的方式又分成移位 (shift moves) 以及交換 (swap moves) 兩類。移位是指$\lambda_2 = 0$ 的情況，例如 *shift* (1, 0) 以及 *shift* (2, 0)；交換是指$\lambda_2 \neq 0$ 的情況，例如 *swap* (1, 1)、*swap* (2, 1) 以及 *swap* (2, 2)，圖 2.6 為此五種節點交換法的示意圖。圖 2.6 中圖 (a) 為 *shift* (1, 0)，(b) 至 (e) 分別為其餘四種λ-interchanges 交換法，各交換法示意圖的左側為執行交換法前的路線，右側則為經交換法搜尋後產生的鄰域解。以圖 2.6 (a) 與圖 2.6 (c) 分別為 *shift* (1, 0) 以及 *swap* (1, 0) 的交換示意圖，*shift* (1, 0) 是將 r_1 路線上的一個節點 *i* 移動至 r_2 路線上；*swap* (1, 0) 則是將 r_1 路線上的一個節點 *i* 和 r_2 路線上的一個節點 *j* 進行位置互換的結果。

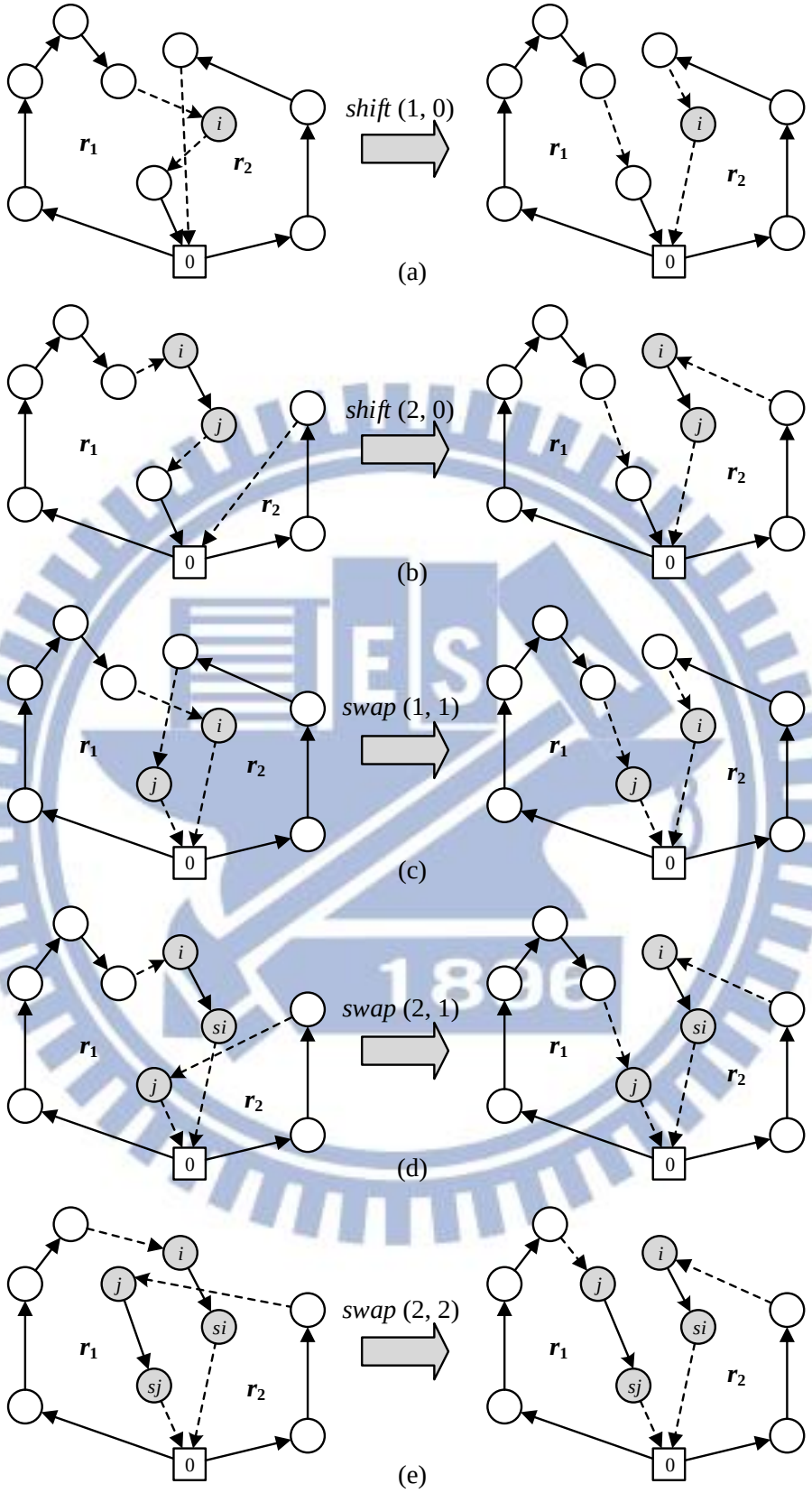

圖 2.6　*λ-interchange* 節點交換法示意圖 ($\lambda = 2$)

資料來源：本研究整理

(5) *node-ejection chain* 節點交換法

node-ejection chain 是一個同時考慮路線間以及路線內的多重節點移動的交換法。第一個 *ejection chain* 的概念是由 Glover [26] 於 1992 年提出並應用於 TSP，Rego [42] 則於 2001 年參考 Glover 的研究，將 *ejection chain* 的移動方式修改成 *node-ejection chain* 交換法，並用於求解 VRP。*node-ejection chain* 首先須要事先挑選 $l+1$ 組「三點聯」，始進行鄰域搜尋。以圖 2.7 說明 $l = 2$ 的 *node-ejection chain* 搜尋，圖 2.7 (a) 為準備進行 *node-ejection chain* 鄰域搜尋的 3 組三點聯，*node-ejection chain* 鄰域搜尋的過程由驅逐 (ejection) 與嘗試 (trial) 兩種動作構成。驅逐是把各層三點聯的中央點依序由第 z 層移至 $z+1$ 層中央點的位置 ($z = 0, 1, \ldots, l-1$)。意即將節點 i 移至點 j 的位置，節點 j 移至點 k 的位置，而節點 k 形成一個落單的點。此時，必須嘗試兩種型式的連接方式，將節點 k 重新連接至原解的特定位置。具體而言，第一類 (Type I) 的連接，是把節點 k 移至最上層的中央點位置 (原本節點 i 的位置)，如圖 2.7 (b)。第二類 (Type II) 的連接，是在 $l+1$ 層結構之外，新找兩個連續的節點 u 與 v，然後把節點 k 插入在該兩點之間，如圖 2.7 (c)。搜尋的結果將以 Type I 以及 Type II 中較佳的方案執行。

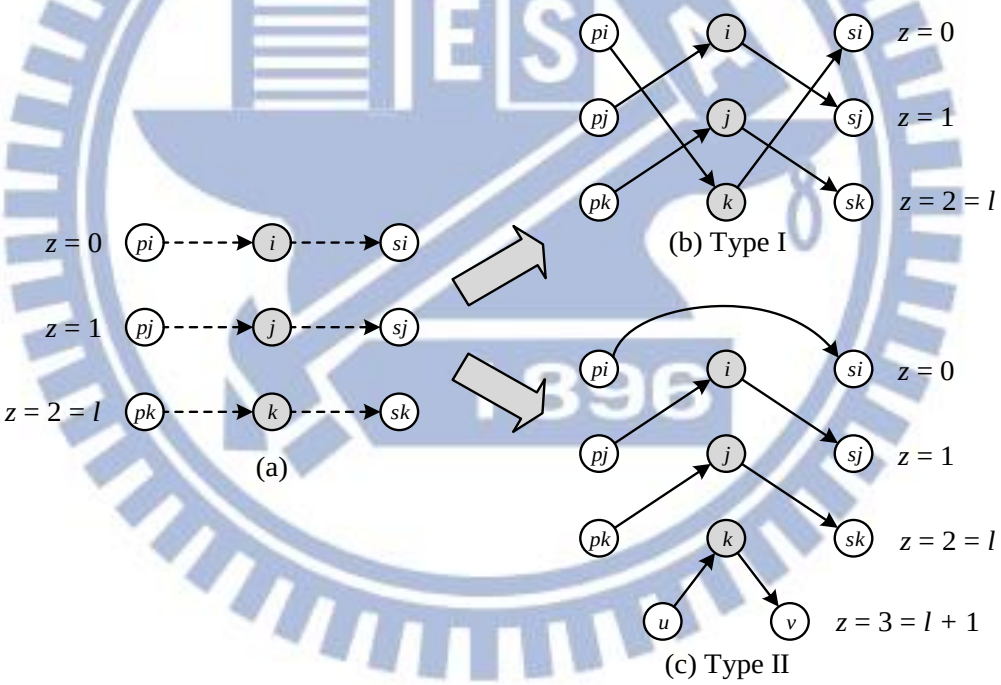

圖 2.7　*node-ejection chain* 鄰域結構示意圖 ($l = 2$)

資料來源：本研究整理

(3) 一般化插入解繫法

Gendreau *et al.* [25] 於 1992 年調整傳統插入法僅能插入兩連續點之間的限制，提出增加考慮讓節點插入至不連續兩點間的一般化插入解繫法 (generalized insertion / unstring and string, GENIUS)，GENIUS 會利用 Type I 和 Type II 兩種策略嘗試將節點於路線中拔出 (繫出，unstring) 與插入 (繫入，string) 以搜尋更好的鄰域解。由於 unstring 與 string 彼此互為反向動作，本研究以 string 為例進行 GENIUS 中 Type I 和 Type II 的

說明。圖 2.8 為 string 第一種方式 Type I 的示意圖，圖中 *si* 表示節點 *i* 的下一節點。圖 2.8 中點 *v* 欲插入 *i* 與 *j* 之間，Type I 會於現行路線中選擇第三個節點 *k*，並分別刪除節線 (i, si)、(j, sj) 以及(k, sk) 後，重新連接節線 (i, v)、(v, j)、(si, k) 以及 (sj, sk)。連結完成後應注意，若路線具有方向性，應反轉路段 $(si,\ldots, j)$ 以及 $(sj,\ldots, k)$ 的路段方向。

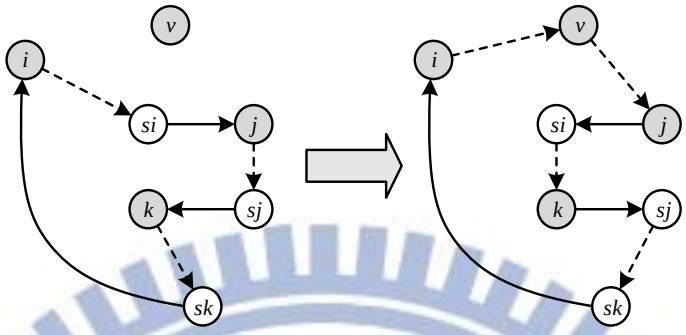

圖 2.8　string Type I 插入法示意圖

資料來源：本研究整理

圖 2.9 為 string Type II 插入法的示意圖，圖中 *pi* 表示節點 *i* 的前一節點，*si* 表示節點 *i* 的下一節點。圖 2.9 中點 *v* 欲插入 *i* 與 *j* 之間，Type II 會於現行路線中選擇點 *k* (為點 *i* 的前一點) 以及 *l* (為點 *j* 的前一點)，並分別刪除節線 (i, si)、(j, sj)、(pk, k) 以及 (pl, l) 後，重新連接節線 (i, v)、(v, j)、(l, sj)、(pk, pl) 以及 (si, k)。連結完成後應注意，若路線具有方向性，應反轉路段 $(j,\ldots, l)$ 以及 $(si,\ldots, pl)$ 的路段方向。

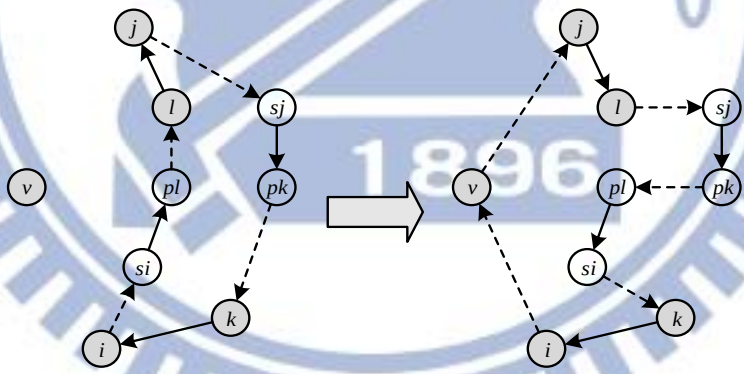

圖 2.9　string Type II 插入法示意圖

資料來源：本研究整理

特別值得一提的是，string 的機制本身就是插入法的一種，所以 Gendreau *et al.* [25] 便直接利用 string 的方式直接構建起始解。起始解構建完成後，再搭配 unstring 進行改善。後續有使用 GENIUS 的 VPRs 研究中，一般也都會採用這樣的方式進行求解。

2.2 分割配送相關車輛路線問題啟發式解法回顧

分割配送車輛路線問題 (SDVRP) 於 1989 年提出至今，已有相當豐富的研究，在求解方法上亦顯示多元性。相關研究所使用的求解方法，包含精確解法的數學規劃法 (Mathematical Programming)，近似解法的啟發式方法 (Heuristic 或 Meta-Heuristic) 或是混合演算法 (Hybrid method) 等等。Dror and Trudeau [24] 在提出 SDVRP 之後，立即證明了 SDVRP 具有 *NP*-hard 的特性。Archetti *et al.* [7] 在 2005 年針對 SDVRP 複雜度的研究成果，更確定當 $Q > 2$ 時，SDVRP 無法在多項式時間 (polynomial time) 內求得精確解。因此，本研究為了能在合理的運算時間內求得品質良好的可行解，故在求解方法以啟發式方法為主。本章節則針對 SDVRP 以及其延伸問題 SDVRP-MDA 的啟發式、巨集啟發式以及混合演算法等解法進行回顧。

SDVRP 的問題型態最早由 Dror and Trudeau [23, 24] 於 1989 年提出。該研究提出兩階段演算法來求解 SDVRP，第一階段先利用 Clarke-Wright 節省法 [18] 構建一個 VRP 的可行起始解，第二階段再以啟發式鄰域搜尋方法對起始解的進行改善。在第二階段改善的過程中，作者除了傳統的交換法外，特別針對 SDVRP 問題特性設計的兩套啟發式方法，分別稱為「*k-split interchange*」以及「*route addition*」。*k-split interchange* 考慮將顧客 i 的全部需求 d_i 重新利用 k 條路線進行服務，是能夠對顧客需求「創造分割」或「改變分割」狀態的鄰域搜尋法。

圖 2.10 為 *k-split interchange* ($k = 2$) *k-split interchange* 在進行交換時會將顧客點 2 的需求 d_2 於原路線中移出，在車容量允許的情況下，由其他兩條路線共同進行服務。因此，圖 2.10 (a) 經過 *k-split interchange* 交換後，整體的成本由原先的 48 可改善為圖 2.10 (b) 的 38。

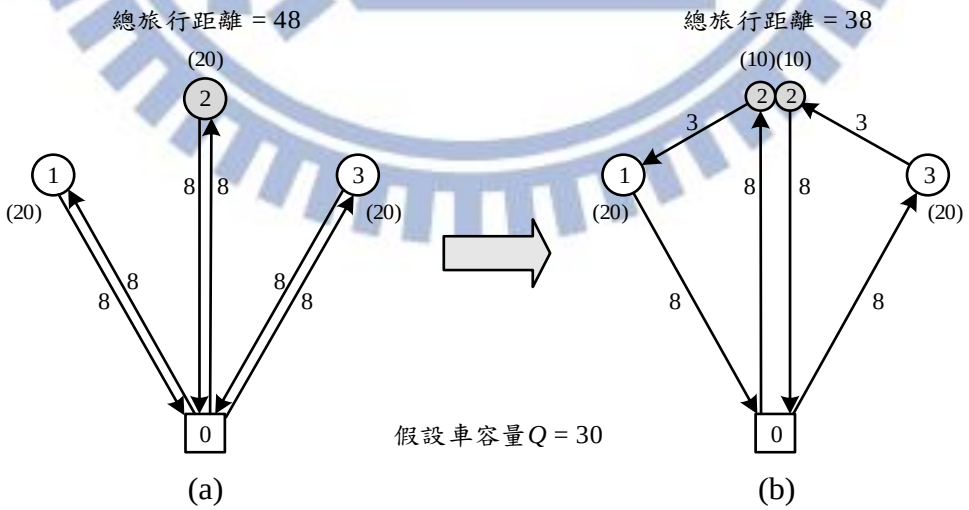

圖 2.10 *k-split interchange* 示意圖 ($k = 2$)

資料來源：本研究整理

route addition 是嘗試將現行解中需求已分割的顧客 *i* 於各路線抽出後，單獨新增一條路線進行服務服務，圖 2.11 為 *route addition* 的示意圖。圖 2.11 (a) 中，顧客 2 的需求被分割由 2 條路線經過服務，*route addition* 執行時會將顧客 2 的所有需求移出，再新增第 3 條路線單獨進行服務。觀察圖 2.10 與圖 2.11，我們可以發現，*k-split interchange* 以及 *route addition* 基本上是互為反向移動，前者是嘗試利用分割需求的方式進行改善，後者則是合併需求以獲得改善。

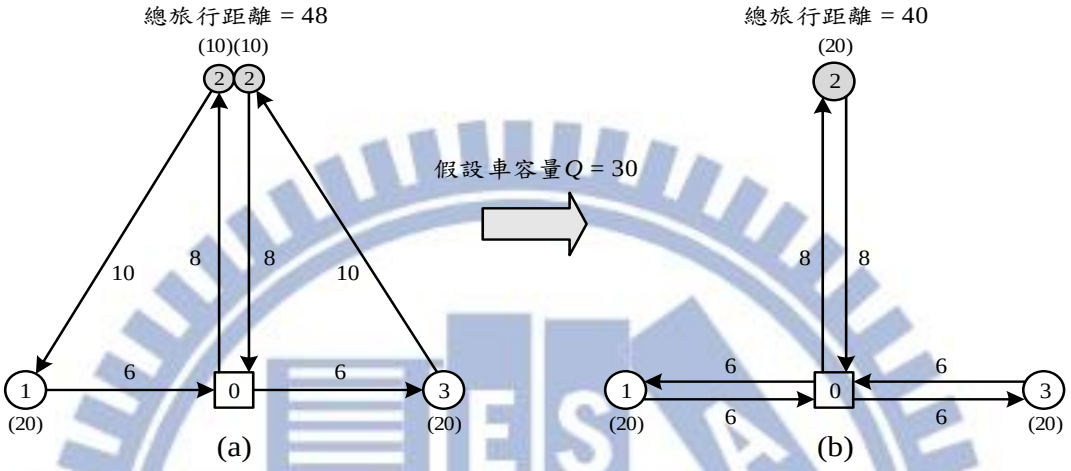

圖 2.11　*route-addition* 示意圖

資料來源：本研究整理

Dror and Trudeau [23, 24] 根據傳統 VRP 的標竿例題為基礎，修改顧客需求後提出 SDVRP 的標竿題庫用以估計需求分割所能產生的節省。顧客需求是依據公式 (1) 進行修改：

$$d_i = \lfloor \alpha Q + \delta(\gamma - \alpha)Q \rfloor \qquad (1)$$

式 (1) 中δ為 [0, 1] 的隨機參數，α以及γ則是用以限制各顧客點需求量的下限以及上限。Dror and Trudeau [23, 24] 分別以下列 6 組不同的α以及γ產生 SDVRP 例題：(0.01, 1), (0.1, 0.3), (0.1, 0.5), (0.1, 0.9), (0.3, 0.7) 以及 (0.7, 0.9)。研究結果發現在需求量相對大的例題中，SDVRP 對於 VRP 的節省越為明顯。

2006 年，SDVRP 被提出後的 16 年，第二個啟發式方法被提出，並且在這之後的研究大多是以巨集啟發式方法或是混合型演算法作為研究求解的架構。Archetti *et al.* [11] 於 2006 年提出以禁制搜尋法 (Tabu Search, TS) 為基礎的 SPLIT-TABU 法求解 SDVRP。SPLIT-TABU 先以 GENIUS 對所有的顧點節點產生巨網 (giant tour)，再依車容量分割成一個沒有需求分割的 VRP 解作為起始解。接續在 TS 的改善階段採用 Dror and Trudeau [23, 24] 提出的 *k-split interchange* 以及 *route addition* 作為鄰域搜尋的核心外，並搭配若干傳統 VRP 啟發式交換方法。與 Dror and Trudeau [23, 24] 的求解結果比較，在求解效果上有顯著的改善。

Boudia *et al.* [15] 於 2007 年提出一套名為 MA|PM 的瀰母演算法 (Memetic

Algorithm, MA) 進行求解 SDVRP，MA 是結合基因演算法 (Genetic Algorithm, GA) 與鄰域搜尋法的巨集啟發式方法。Boudia *et al.* [15] 在演算法中加入一個染色體族群的管理機制，該機制以距離來衡量染色體間的差異，進而加強求解搜尋的廣度。在鄰域搜尋的過程中，除了應用傳統的 VRP 交換法以外，另採用三個允許分割配送的搜尋法。第一是以 Dror and Trudeau [23, 24] 提出的 *k-split interchange* 為基礎，Boudia *et al.* [15] 加入新的演算程序排序各顧客進行分割配送的適合度，再依排序後的結果進行 *k-split interchange*。第二以及第三分別是兩路線間 1 對 1 以及 1 對 2 節點的交換。此二種交換法在提出時並無命名，但在後續 Silva *et al.* [44] 的研究中被引用時，分別稱為 *Swap* (1, 1)* 以及 *Swap* (2, 1)*。不同於傳統 VRP 的節點交換法，這兩種交換法在節點交換需求量時允許利用需求分割來避免發生路線違反車容量限制的情況。以 *Swap* (1, 1)* 舉例說明，圖 2.12 為 *Swap* (1, 1)* 交換法的示意圖，當點 *i* 與點 *j* 於兩路線間進行交換時，假設 $d_i > d_j$，則將點 *j* 完全移至點 *i* 前，點 *i* 則由原本的路線分割等同於 d_j 的需求量至點 *j* 原本的位置。如此一來，兩路線的總乘載量不會改變，能夠保持車容量的可行性。求解的結果上，該研究利用 Dror and Trudeau [23, 24] 以及 Belenguer *et al.* [13] 提出的標竿題庫進行測試，兩題庫的求解結果皆有相當大的突破。

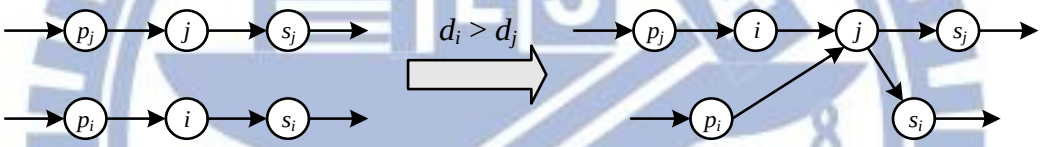

圖 2.12　*swap* (1, 1)* 交換法示意圖

資料來源：本研究整理

於 2007 年，Chen *et al.* [17] 首次採用混合演算法求解 SDVRP，演算法稱為 EMIP+VRTR。起始解部分以 Clarke-Wright 節省法 [18] 先構建一個 VRP 起始可行解。然後，接以 EMIP (Endpoint Mixed Integer Program) 混合整數規劃法在允許需求分割的情況下重新配置各車輛路線適合的起點以及末點。

EMIP 的概念在於作者認為利用距離場站較近的顧客點進行需求分割可得到較大的利益，反之分割較遠的顧客則會造成相對較大的成本。經過 EMIP 求解過後的答案接續以 VRTR (Variable length Record-To-Record travel algorithm) 搭配傳統 VRP 交換法進行改善。作者首先以 EMIP + VRTR 架構求解 Archetti *et al.* [11] 測試的標竿題庫，發現求解績效上優於先前的結果。此外，另提出 21 題例題的 SDVRP 標竿題庫，該題庫中的例題顧客點數為 8 到 288，車容量均為 100，顧客需求為 60 或者是 90。顧客位置為對稱型成同心圓的型態，發車場站位於圓心處。由於例題具有對稱的特性，容易利用視覺判斷的方式求得品質好的估計解，以三輛車服務四個顧客的方式 (如圖 2.13 所示，括弧內為需求量) 對所有的例題求得估計解，再將演算法的求解結果與估計解進行比較，誤差約為 3%，且大規模例題需花費較長的時間。

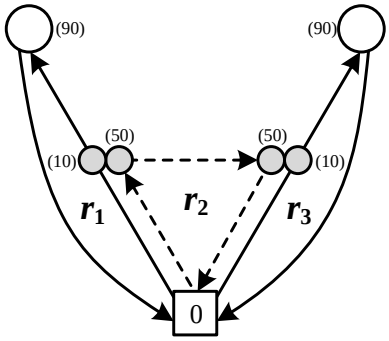

圖 2.13　Chen *et al.* [17] 視覺估計解使用的模組

資料來源：本研究整理

第二個採用混合演算法求解 SDVRP 的研究是 2008 年的 Archetti *et al.* [12]，該研究方法的邏輯是品質好的路線互相具有共同的路線特徵。該研究認為某條節線若鮮少出現在品質好的答案中，則表示該條節線越不要需考慮。同理，若某個節點在品質好的答案中皆未發生分割，則表示該節點越不適合進行分割配送。因此，該研究先利用 Archetti *et al.* [11] 提出的 SPLIT-TABU 法進行求解數次，並以上述原則對問題進行節線的刪減，以及限制某些顧客不可進行需求分割。作者提出一個 route-based 的數學模式，針對簡化後的問題以線性規劃產生個別路線，再接以 set-covering 混合整數規劃產生最終可行解。該研究以 Archetti *et al.* [11] 提出的標竿題庫進行測試，並求得品質更佳的結果。

Mota *et al.* [38] 於 2007 年以分散式搜尋法 (Scatter Search, SS) 求解 SDVRP，該研究將最少可用車輛數 m 的限制加入 SDVRP 中。由於 SDVRP 允許需求進行分割配送，所以只要總車容量大於總需求量則必定存在 SDVRP 的可行路線解。最小車輛數 m 的參數值可以利用下列公式 (2) 估計：

$$m = \left\lceil \sum_{i=1}^{n} d_i \Big/ Q \right\rceil \tag{2}$$

Mota *et al.* [38] 利用兩種方法產生多個起始解以利 SS 進行求解。第一個方法是使用 Lin and Kernighan [36] 啟發式方法將所有顧客排成一個巨網 (該研究稱之為 Big Tour)，再利用不同的顧客點作為分割路線的起點，產生多個 m 條路線的 SDVRP 起始解。另一個方法則是將隨機亂數加入節省法計算節省值的公式中，變動節省值的計算結果，並產生多個起始解。改善階段是將傳統 VRP 鄰域搜尋法在允許需求分割的條件下進行調整應用。測試結果發現 SS 的求解結果有數題優於 Archetti *et al.* [11]。

由於 Mota *et al.* [38] 提出具最少可用車輛數限制的 SDVRP 問題，此亦使得後續相關 SDVRP 研究的求解比較分成有無限制可用車輛數兩種。在後續 Archetti *et al.* [6] 的研究中將無限制可用車輛數的 SDVRP 稱為 SDVRP-UF (SDVRP with Unlimited Fleet)，具可用車輛數限制的 SDVRP 稱為 SDVRP-LF (SDVRP with Limited Fleet)。

Aleman *et al.* [4] 於 2010 年提出一個具有改善機制的路線構建方法 ICA (Iterative Construction Algorithm) 搭配變動鄰域下降 VND (Variable Neighborhood Decent) 方法求解 SDVRP-LF。ICA 是一個以最省插入法為基礎的構建法，ICA 首先利用各點與場站的距離，由大到小排列產生顧客插入順序的清單 L，再依序進行顧客點的插入。插入位置的選擇，除了該位置的插入成本較低以外，ICA 同時考慮插入後，使路線開角 (polar angle) 增加幅度小的位置。構建完成後，ICA 會嘗試減少各路線的開角，調整插入順序清單 L 中的排序，再重新構建另外一個新的起始解。反覆調整直到無法改善後，即開始進入 VND 改善模組。該研究 VND 中包含三個考慮需求分割特性的鄰域搜尋法：*shift*、*swap* 以及 *shift**。*shift* 與 *swap* 兩種鄰域搜尋法同傳統 VRP 的路線間節點移動以及節點交換，兩交換法在執行後若有需求屬於同顧客則會進行合併。*shift** 則是作者針對需求分割的特性設計的鄰域搜尋法，圖 2.14 為其示意圖。首先於路線 r_1 中選取顧客點 i 移至 r_2 服務，交換後若 r_2 發生超載的情況，則於 r_2 中選取另一顧客點 j 分割適當需求量至 r_1 服務，以保持可行性。

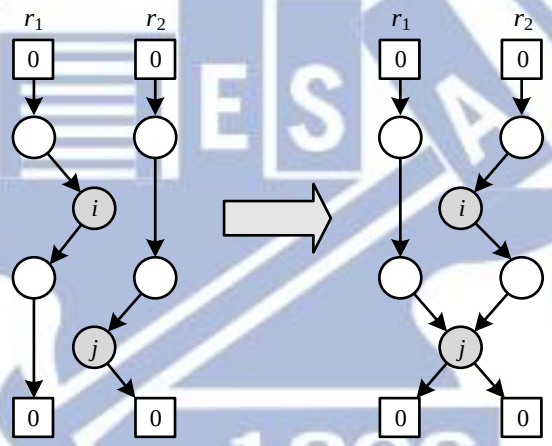

圖 2.14　*shift** 交換法示意圖

資料來源：本研究整理

Aleman *et al.* [4] 以 ICA+VND 針對包括 Archetti *et al.* [11]、Belenguer *et al.* [13]、Chen *et al.* [17] 以及 Jin *et al.* [32] 等 SDVRP 標竿題庫進行測試比較。其求解表現不論在績效或是效率上皆顯示有相當的競爭力，特別在 Chen *et al.* [17] 提出的 21 題例題中便突破 16 題當時最佳解。

隨後 Aleman and Hill [3] 於同年再提出一套以禁制搜尋法為基礎的 TSVBA (Tabu Search with Vocabulary Building Approach) 法求解 SDVRP-LF。該方法是一個利用多個起始解互相擷取資訊的方法，概念上類似 Archetti *et al.* [12]。該研究將 Aleman *et al.* [4] 所提出的 ICA 構建法進行調整，使之能產生多個起始解。作者另修改 Aleman *et al.* [4] 所提出的 VND 模組對各起始解進行改善，並存入一個解的集合中。隨後分析這個集合之中的解，保留好的解的特徵，再產生新的解。新的解若優於目前集合中的任一解，則將新解加入集合中並從集合中移除一個最差的解，反覆執行直到滿足停止條件輸出集合中的最佳解。求解的結果發現 TSVBA 能夠有效的改善 ICA+VND 的求解結果。

2010 年，Derigs *et al.* [22] 提出一套鄰域搜尋法搭配 SA、TA、RRT、ABHC (Attribute Based Hill Climber) 以及 ABLBS (Attribute Based Local Beam Search) 等五種巨集啟發式策略求解 SDVRP-UF。起始解部分以先排程後分群 (route first, cluster second) 的方式隨機構建 SDVRP 的可行解，之後便開始進入該研究的重點鄰域搜尋模組。該研究所提出的鄰域搜尋模組中共有四種考慮分割需求特性的鄰域交換法，分別為 2-*OPT**、*EXCHANGE*、*RELOCATE I* 以及 *RELOCATE*，其中以 *RELOCATE* 相對複雜。*RELOCATE* 是在兩路線間進行節點移動的交換法，在執行時若發生目標路線無足夠車容量的情況，演算法能夠適應性地採取不同的策略使目標路線移出部分原路線上的需求量，進而讓節點移動能夠執行以加強搜尋的廣度。該研究以 Archetti *et al.* [11] 以及 Chen *et al.* [17] 兩組標竿題庫進行測試比較，結果發現以 ABHC 的求解績效最好並突破大部分的當時已知最佳解。

Gulczynski *et al.* [29] 於 2010 年所提出的混合型演算法 EMIP-MDA+ERTR。該研究所使用的方法與 Chen *et al.* [17] 的 EMIP+VRTR 有部分類似，皆利用數學建模的方式選擇與決定分割的節點以及需求量的分配，再將子問題的求解結果接以巨集啟發式方法優化，最後輸出答案。Gulczynski *et al.* [29] 在起始解部分採用 1970 年時 Yellow [49] 提出的修正型節省法搭配三組不同的參數構建出三個無需求分割的 VRP 路線解，並選擇其中成本最低者做為起始解。起始解產生後接以 EMIP-MDA (Endpoint Mixed Integer Program with Minimum Delivery Amounts) 進行混合整數規劃求解，EMIP-MDA 是基於 Chen *et al.* [17] 提出的 EMIP，可以最佳化各路線乘載路線中首位與末位顧客點的需求分配。經過 EMIP-MDA 數學規劃求解後，該研究續採用 2009 年 Groër *et al.* [27] 提出的 ERTR (Enhanced Record-To-Record travel algorithm) 進行優化求解。

該研究提出兩個 SDVRP-MDA 標竿題庫以進行的求解測試。第一個題庫是修改自 Chen *et al.* [17] 的 21 題 SDVRP 標竿題庫。由於作者為了比較演算法求解的績效，因此採用 Chen *et al.* [17] 同樣的方式設計一套顧客點位置對稱的標竿題庫。對稱型的例題對於利用視覺估計方式產生估計解 (Estimated Solution) 較為容易，估計解便可作為標竿題庫的已知最佳解 (Best Known Solution, BKS) 以比較 EMIP-MDA+ERTR 的解題績效。此外，由於 SDVRP-MDA 具有最小配送量限制，且該限制受到最小配送比例 p 的影響。Gulczynski *et al.* [29] 為了讓同一組 BKS 路線在不同 p 值時皆為可行路線。因此在題庫中，例題的顧客點需求需要對應不同的 p 值進行調整，以圖 2.15 說明之。

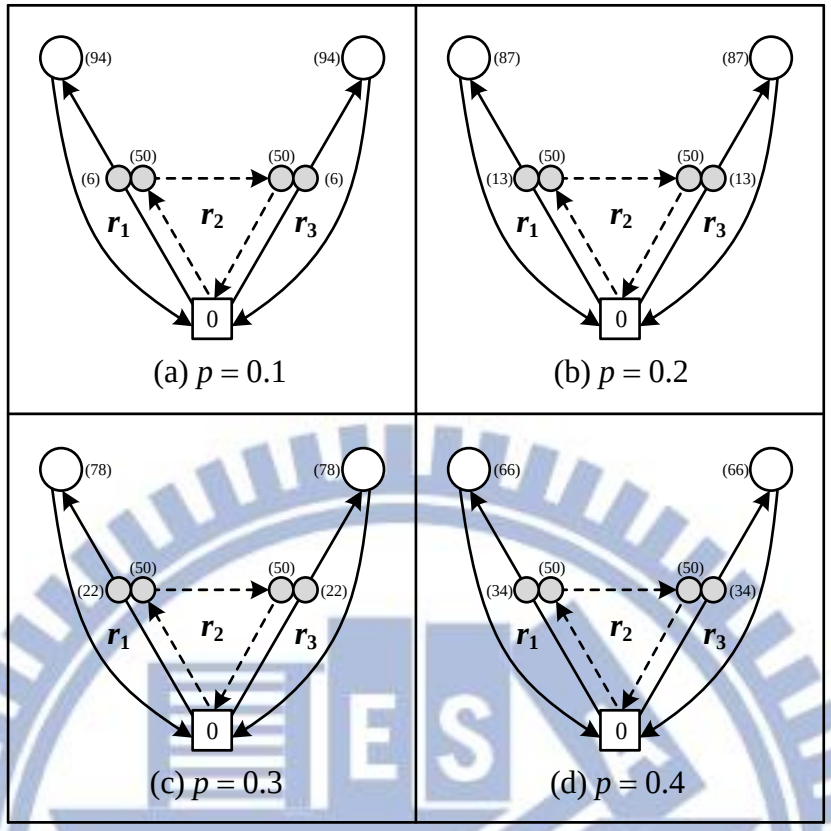

圖 2.15　Gulczynski *et al.* [29] 視覺估計路線型態以及路線中配送需求的分配

資料來源：本研究整理

Gulczynski *et al.* [29] 設定顧客編號為基數時，該顧客需求等於$\lceil 50/(1-p) \rceil$，編號若為偶數時則為 $150-\lceil 50/(1-p) \rceil$。舉例 $p=0.1$ 時，顧客 1 的需求為 56，顧客 2 的需求為 94；例 $p=0.2$ 時，顧客 1 的需求為 63，顧客 2 的需求為 87。需求調整後，該研究採用的視覺估計路線型態以及路線中配送需求的分配如圖 2.15 所示。配合 4 個不同的 p 值 (p = 0.1, 0.2, 0.3 與 0.4)，第一個題庫共 84 (21 × 4) 題。第二個題庫為採用 Belenguer *et al.* [13] 14 題標竿題庫中的 11 題例題，配合 4 個不同的 p 值，共 44 題 (11 × 4) 例題。本題庫在顧客需求量方面則保持原本的數值。比較階段中，作者針對第一個題庫將視覺估計解與 EMIP-MDA+ERTR 的求解結果進行比較，誤差為 1.34%，並有 2 題的求解結果優於視覺估計解。第二個題庫的結果則於文獻中列出，以供後續研究進行參考。

Berbotto *et al.* [14] 於 2014 年以 Randomized Granular Tabu Search (RGTS) 求解 SDVRP-LF。RGTS 首先以允許需求分割的平行 Clarke-Wright 節省法 [18] 構建一 SDVRP 起始解，後續進入 TS 的階段。在 TS 深度優化的階段，作者除了採用數個文獻已知的交換法以外，亦提出兩種考慮三條路線間具分割特性的節點移動交換法，分別是 *DelSplitNew* 以及 *DelSplit*。*DelSplitNew* 交換法的示意圖如圖 2.16 所示，*DelSplitNew* 能夠同時消除以及產生需求的分割，執行時選擇一個被路線 r_1 以及 r_2 共同服務的節點 i，將 i 在 r_1 的需求量 y_{ir1} 移至 r_2。然後，在 r_2 中找一個節點 j 分割其同等 y_{ir1} 的需求量至

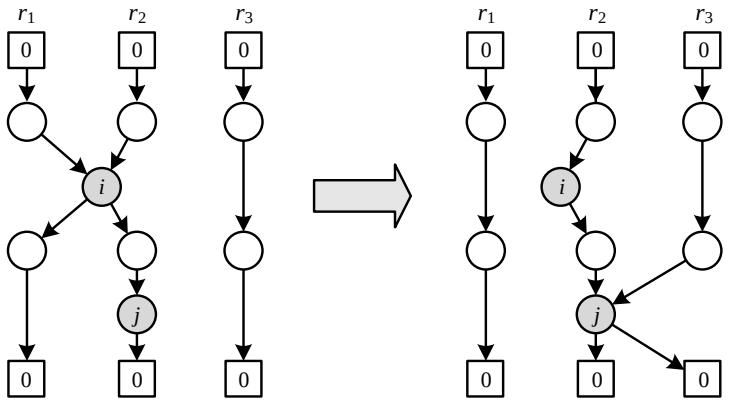

圖 2.16　*DelSplitNew* 交換法示意圖

資料來源：本研究整理

r_3。圖 2.17 為 *DelSplit* 的示意圖，*DelSplit* 執行時會選擇一個被路線 r_1 以及 r_3 共同服務的節點 i，將 i 在 r_1 以及 r_3 的需求量合併後移至 r_2 以消除需求分割的情況。作者利用 Archetti *et al.* [11]、Chen *et al.* [17]、Belenguer *at al.* [13] 提出的標竿題庫進行測試，發現 RGTS 相較其他演算法有不錯的求解表現。

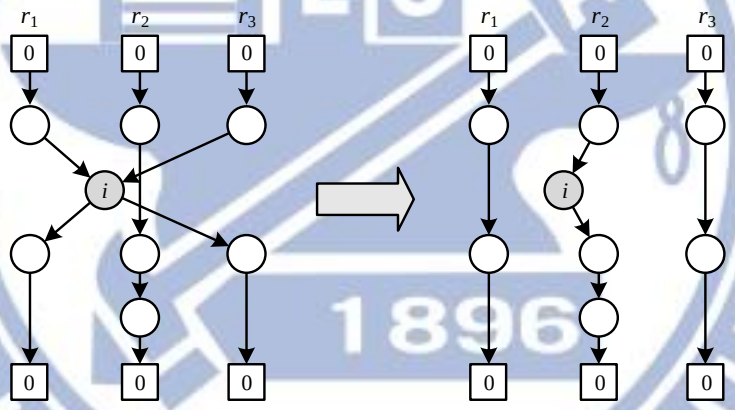

圖 2.17　*DelSplit* 交換法示意圖

資料來源：本研究整理

Silva *et al.* [44] 於 2015 年提出 SplitILS-LF 以及 SplitILS-UF 分別求解 SDVRP-UF 以及 SDVRP-LF。SplitILS-LF 以及 SplitILS-UF 主要是以迭代區域搜尋法 (ILS) 做為求解架構。起始解部分採用隨機最近插入法產生無分割需求的 VRP 起始解，後續則進入 ILS 重複執行擾動機制 (Perturbation Mechanism) 以及隨機變動鄰域下降 (Randomized VND, RVND) 直到預設的停止條件到達，如此為搜尋一次。作者重複執行數次上述搜尋的流程，最後求解結果為搜尋過程中最佳解。該研究同時使用了 14 種不同的鄰域搜尋法於 RVND 中，包含 10 個傳統 VRP 交換法以及 4 個 SDVRP 的交換法。該研究以 Belenguer *at al.* [13]、Archetti *et al.* [11]、Archetti *et al.* [12] 以及 Chen *et al.* [17] 等標竿題庫進行 ILS 求解績效測試。測試結果於 324 題例題中求得 55 題當時已知最佳解，並突破 243 題。

表 2.1 為分割配送相關車輛路線問題啟發式研究的文獻回顧彙整表。由表中可以觀察到 SDVRP 的問題型態在 1989 年就被提出，但啟發式方法的研究幾乎集中在 2006 到 2015 年的近十年左右的期間。在研究方法方面，僅有少數文獻是自行提出針對需求分割特性所設計的鄰域搜尋法，其他大部分的文獻則是額外搭配巨集策略進行求解。

表 2.1　分割配送相關車輛路線問題啟發式解法文獻彙整表

問題類型	年分	作者	分割特性的鄰域搜尋法	巨集策略
SDVRP	1989	Dror and Trudeau [23]	*k*-split interchange route addition	–
	2006	Archetti*et al*. [11]	–	Tabu Search
	2007	Boudia *et al*. [15]	*swap* (1, 1)* *swap* (2, 1)*	Memetic Algorithm
	2007	Chen *et al*. [17]	–	Record-to-Record Travel
	2007	Mota *et al*. [38]	–	Scatter Search
	2008	Archetti *et al*. [12]	–	Tabu Search
	2010	Aleman *et al*. [4]	*shift**	–
	2010	Aleman and Hill [3]	–	Hybrid Tabu Search
	2010	Derigs *et al*. [22]	*RELOCATE*	Simulated Annealing Threshold Acepting Record-to-Record Travel Atribute Based Hill Climber Atribute Based Local Beam Search
	2014	Berbotto *et al*. [14]	*DelSplitNew* *DelSplit*	Tabu Search
	2015	Silva *et al.* [44]	–	Iterated Local Search
SDVRP-MDA	2010	Gulczynski *et al.* [29]	–	Record-to-Record Travel
	2016	Han and Chu [30]	*SNEC*	Multi-Start

分割配送相關車輛路線問題提出至今亦有若干精確解法被提出。Belenguer *et al.* [13] 於 2000 年提出 SDVRP 標竿題庫，並利用有效不等式 (Valid Inequalities) 搭配割平面法 (Cutting-plane) 求解最佳解目標成本的上下限。Jin *et al.* [32] 於 2007 年提出的兩階段解法，該方法可求得 22 個顧客點例題的精確解。2008 年 Jin *et al.* [33] 以變數產生法 (Column Generation) 求解 SDVRP 的上下限，結果改善了 Belenguer *at al.* [13] 當時的研究。Archetti *et al.* [5] 於 2014 年則嘗試以 branch-and-cut 方法求解 SDVRP，並成功改善了數題 SDVRP 標竿例題的上下限。目前 SDVRP 精確解法的研究已求得標竿題庫中若干題例題的最佳解，獲得精確解證明的大多是顧客點數小於等於 50 點的例題。相關 SDVRP 的精確解研究可參考 Archetti and Speranza [10] 的回顧型文章，該文獻內容中對於 SDVRP 最佳解性質、近似解法、精確解法以及 SDVRP 延伸問題進行詳細且完整的回顧。

第三章、分割配送相關車輛路線問題數學定義與問題性質

3.1 分割配送相關車輛路線問題數學模式

本章節將分別對 SDVRP 以及 SDVRP-MDA 兩種不同的分割需求車輛路線問題的數學定義進行回顧，並針對本研究新提出之 SDVRP-LND 問題建立其數學規劃模式。由於三個問題的數學模式類似，因此先於此處將共用的問題變數進行定義，共用變數定義說明如表 3.1 所示：

表 3.1 共用變數定義表

變數	定義
G	完全性網路
V	節點集合，V=不厭不厭，點 0 為場站，點 1 至 n 為顧客節點
A	節線集合，$A=\{(i,j)\mid i,j\in V, i\neq j\}$
c_{ij}	點 i 到點 j 的旅行成本，$c_{ij}>0$，$i,j\in V$
d_i	點 i 的需求量，$i\in V-\{0\}$
Q	車容量
x_{ij}^v	車輛 v 是否行經節線 (i,j) 的決策變數，是則等於 1；否則為 0，$i,j\in V$
q_{iv}	車輛 v 服務點 i 的需求量，$i\in V-\{0\}$

3.1.1 分割配送車輛路線問題定義與數學規劃模式

分割配送車輛路線問題 (SDVRP) 為傳統 VRP 問題的延伸，其主要放鬆了 VRP 中各顧客僅能由單一部車輛進行服務的限制，意即顧客需求能夠進行分割，使得多部車輛共同服務同一個顧客。SDVRP 於 1989 年由 Dror and Trudeau [23] 提出時，僅以文字描述問題之特徵以及限制，直到 2006 年始由 Archetti *et al.* [11] 針對 SDVRP 提出數學規劃模式。Archetti *et al.* [11] 將 SDVRP 定義於一個完全的網路 $G=(V, E)$ 上。假設各車輛 v 之容量相同均為 Q，且有足夠數量 m 的車隊可供使用，則 SDVRP 可定義為如下之混合整數規劃定式。

$$\text{Min} \quad \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{v=1}^{m} c_{ij}x_{ij}^v \tag{3}$$

Subject to

$$\sum_{i=0}^{n}\sum_{v=1}^{m} x_{ij}^v \geq 1, \qquad \text{for } j=1,\ldots,n \tag{4}$$

$$\sum_{i=0}^{n} x_{ip}^v - \sum_{j=0}^{n} x_{pj}^v = 0, \qquad \text{for } p=0,\ldots,n;\ v=1,\ldots,m \tag{5}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij}^v \leq |S|-1, \qquad \text{for } v=1,\ldots,m;\ S\subseteq V-\{0\} \tag{6}$$

$$q_{iv} \le d_i \sum_{j=0}^{n} x_{ij}^{v}, \qquad \text{for } i = 1,\ldots, n;\ v = 1,\ldots, m \tag{7}$$

$$\sum_{v=1}^{m} q_{iv} = d_i, \qquad \text{for } i = 1,\ldots, n \tag{8}$$

$$\sum_{i=1}^{n} q_{iv} \le Q, \qquad \text{for } v = 1,\ldots, m \tag{9}$$

$$x_{ij}^{v} \in \{0,1\}, \qquad \text{for } i = 0,\ldots, n;\ j = 1,\ldots, n;\ v = 1,\ldots, m \tag{10}$$

$$q_{iv} \ge 0, \qquad \text{for } i = 1,\ldots, n;\ v = 1,\ldots, m \tag{11}$$

在 Archetti *et al.* [11] 所提出的 SDVRP 數學規劃模式中，式 (3) 為 SDVRP 的目標函數，其求解目標為最小化距離成本。式 (4) 則為 SDVRP 放鬆 VRP 中各顧客僅可由一輛車輛服務至少一次的限制式，強調服務顧客的車輛數可以超過一輛。特別說明，在原本 Archetti *et al.* [11] 所提出的數學模式中式 (4) 是針對節點 $j = 0,\ldots, n$ 的限制式。本研究為了讓限制式強調顧客需求允許分割的目的更為明確，因此將對應的節點改為 $j = 1,\ldots, n$，此部分的修改對問題的本質並不會造成影響。式 (5) 與 (6) 為傳統 VRP 之節點流量守恆以及清除子迴圈限制式。式 (7) 至式 (9) 為分配顧客之需求：式 (7) 表示唯有在車輛 v 服務 i 點時，q_{iv} 才可以存在且小於或等於顧客 i 的需求總量 d_i，式 (8) 則是要求各節點 i 的需求總量 d_i 須被滿足，式 (9) 為限制各車之承載量不得大於 Q。式 (10) 與式 (11)則定義變數 x_{ij}^{v} 與 q_{iv} 之值域。

3.1.2 具最小配送量限制之分割配送車輛路線問題定義與數學規劃模式

具最小配送量限制之分割配送車輛路線問題 (SDVRR-MDA) 為 SDVRP 的變形問題，該問題限制每一輛車輛 v 對單一顧客 i 的服務量 q_{iv} 分別必須大於等於顧客 i 的最小配送量 MDA_i 的限制。以下說明 SDVRP-MDA 的問題定義。

SDVRR-MDA 的問題目標以及相關限制最早由 Gulczynski *et al.* [29] 於 2010 年所提出。問題數學模式則由 Han and Chu [30] 於 2016 年參考 Archetti *et al.* [11] 對於 SDVRP 的數學模式進一步修改後提出。以下為 SDVRP-MDA 的問題數學定式:

$$\text{Min} \quad \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{v=1}^{m} c_{ij} x_{ij}^{v} \tag{3a}$$

subject to

$$\sum_{i=0}^{n}\sum_{v=1}^{m} x_{ij}^{v} \ge 1, \qquad \text{for } j = 1,\ldots, n \tag{4a}$$

$$\sum_{i=0}^{n} x_{ik}^{v} - \sum_{j=0}^{n} x_{kj}^{v} = 0, \qquad \text{for } k = 0,\ldots, n;\ v = 1,\ldots, m \tag{5a}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij}^{v} \le |S| - 1, \qquad \text{for } v = 1,\ldots, m;\ S \subseteq V\text{–}\{0\} \tag{6a}$$

$$q_{iv} \le d_i \sum_{j=0}^{n} x_{ij}^{v}, \qquad \text{for } i = 0,\ldots, n;\ v = 1,\ldots, m \tag{7a}$$

$$\sum_{v=1}^{m} q_{iv} = d_i, \qquad \text{for } i = 1,\ldots, n \tag{8a}$$

$$\sum_{i=1}^{n} q_{iv} \le Q, \qquad \text{for } v = 1,\ldots, m \tag{9a}$$

$$x_{ij}^{v} \in \{0,1\}, \qquad \text{for } i = 0,\ldots, n;\ j = 0,\ldots, n;\ v = 1,\ldots, m \tag{10a}$$

$$q_{iv} \ge \text{MDA}_i \sum_{j=0}^{n} x_{ij}^{v}, \qquad \text{for } i = 0,\ldots, n;\ v = 1,\ldots, m \tag{11a}$$

其中式 (3a) 為 SDVRP-MDA 的目標函數，表示總旅行成本最小化。限制式部分式 (4a) – (10a) 對應的是 SDVRP 的限制式 (4) – (10)，限制式的目的說明可參考 3.1.1 節內文。式 (11a) 則是將 SDVRP 問題轉變成為 SDVRP-MDA 的關鍵限制式。原 SDVRP 中的式 (11) 只限制 q_{iv} 值的值域為非負。但 SDVRP-MDA 不同，式 (11a) 限制若車輛 v 服務顧客 i，則 q_{iv} 必須大於或等於顧客 i 的最小配送量 MDA_i。

在 Gulczynski *et al.* [29] 的研究中，MDA_i 值是以 d_i 的最小配送比例 p 來設定。Gulczynski *et al.* [29] 研究中對於 p 值值域的設定為 $0 \le p \le 1$。但本研究認為 p 值的合宜設定應調整為 $0 \le p \le 0.5$。原因是當 $p > 0.5$ 時，由於任一顧客 i 的需求量 d_i 一旦發生分割配送，任份配送量便不可能大於原本的 50%，若要滿足最小配送量 MDA_i 的限制，最後的結果將不會產生任何分割。亦即當 $p > 0.5$ 時，SDVRP-MDA 的求解結果應為 VRP 的可行解。進一步推論，SDVRP 以及 VRP 其實皆為 SDVRP-MDA 的特例，分別在 $p = 0$ 以及 $p = 0.5$ 時。

3.1.3 具配送次數限制之分割配送車輛路線問題定義與數學規劃模式

具配送次數限制分割車輛路線問題 (SDVRP with Limited Number of Delivery, SDVRP-LND) 為本研究新提出之分割配送類型的車輛路線問題，其概念是參考國內物流業者對分割配送次數上的考量。本節將對 SDVRP-LND 的問題定義以及數學模式進行說明。

本研究基於 Archetti *et al.* [11] 對於 SDVRP 的數學模式進一步修改為 SDVRP-LND 的數學模式，其中修改的部分為新增加一個變數 *kmax* 作為配送次數的上限 (式 12)，變數 *kmax* 代表的是顧客被服務的次數上限。故本研究提出的 SDVRP-LND 數學模式如下。

$$\text{Min} \quad \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{v=1}^{m} c_{ij} x_{ij}^{v} \tag{3b}$$

Subject to

$$\sum_{i=0}^{n}\sum_{v=1}^{m} x_{ij}^{v} \geq 1, \qquad \text{for } j = 1,\ldots, n \tag{4b}$$

$$\sum_{i=0}^{n} x_{ip}^{v} - \sum_{j=0}^{n} x_{pj}^{v} = 0, \qquad \text{for } p = 0,\ldots, n;\ v = 1,\ldots, m \tag{5b}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij}^{v} \leq |S| - 1, \qquad \text{for } v = 1,\ldots, m;\ S \subseteq V-\{0\} \tag{6b}$$

$$q_{iv} \leq d_i \sum_{j=0}^{n} x_{ij}^{v}, \qquad \text{for } i = 1,\ldots, n;\ v = 1,\ldots, m \tag{7b}$$

$$\sum_{v=1}^{m} q_{iv} = d_i, \qquad \text{for } i = 1,\ldots, n \tag{8b}$$

$$\sum_{i=1}^{n} q_{iv} \leq Q, \qquad \text{for } v = 1,\ldots, m \tag{9b}$$

$$x_{ij}^{v} \in \{0,1\}, \qquad \text{for } i = 0,\ldots, n;\ j = 1,\ldots, n;\ v = 1,\ldots, m \tag{10b}$$

$$q_{iv} \geq 0, \qquad \text{for } i = 1,\ldots, n;\ v = 1,\ldots, m \tag{11b}$$

$$\sum_{i=0}^{n}\sum_{v=1}^{m} x_{ij}^{v} \leq kmax, \qquad \text{for } j = 1,\ldots, n \tag{12}$$

其中，式 (3b) 為 SDVRP-LND 的目標函數，與 SDVRP 以及 SDVRP-MDA 相同為總旅行成本最小化。式 (4b) – (11b) 與 SDVRP 的式 (4) – (11) 相同，限制式的目的說明可參考 3.1.1 節內文。式 (12) 為 SDVRP 轉換為 SDVRP-LND 的關鍵新增限制式，式 (12) 對所有的顧客點 j 限制服務其的車輛總數不可超過 $kmax$ 輛，意即受限制的分割配送次數。

本研究設定配送次數上限 $kmax$ 值的值域為 $1 \leq kmax \leq m$，意即每個顧客至少需要有一輛車路線經過服務，至多則為 m 輛車輛 (車數上限) 經過。$kmax$ 值的大小將會影響 SDVRP-LND 的求解結果，當 $kmax$ 為 m 時，等同於對分割配送的次數僅售車輛數上限的限制，因此求解結果會與 SDVRP 相同；反之，當 $kmax$ 為 1 時，每個節點被限制只能由一輛車輛服務，求解結果會與 VRP 相同，故 SDVRP 與 VRP 為 SDVRP-LND 的特例。

3.2 分割配送車輛路線相關問題整數解性質與潛在效益

3.2.1 分割配送車輛路線問題特性

分割配送車輛路線問題性質在整數解性質方面已於 Archetti *et al.* [11] 的研究中進行證明，假設若 SDVRP 問題 (P) 具有可行解，則必然有一組配送量變數 $q_{iv} \in Z^{+}$。以 s^* 表示 (P) 的最佳解，z^* 表示為 s^* 的目標函數值，$x_{ij}^{v}{}^*$ 與 q_{iv}^* 分別代表最佳解所對應的變數數值，B 代表顧客點與車輛配對 (i, v) 的集合，$i \in V-\{0\}$, $v \in \{1, ..., m\}$，滿足 $\sum_{j=0}^{n} x_{ij}^{v}{}^* \geq 1$。假設經過顧客 i 的車輛集合為 $S(i) = \{v \mid (i, v) \in B\}$，車輛 v 所經過的顧客點集合為

$D(v) = \{i \mid (i, v) \in B\}$。考慮以下限制式：

$$\sum_{v \in S(i)} q_{iv} = d_i, \qquad \text{for } i = 1,\ldots, n \qquad (8')$$

$$\sum_{i \in D(v)} q_{iv} \le Q, \qquad \text{for } v = 1,\ldots, m \qquad (9')$$

$$q_{iv} \ge 0, \qquad \text{for } (i, v) \in B \qquad (11')$$

可以發現這些限制式為傳統指派問題的限制式，當限制式的右手邊值 (right-hand side) 為整數時必定存在一組 q_{iv} 為整數的指派方式可以滿足限制式。因 q_{iv} 變數並未出現在目標函數中，故將此整數的指派方式帶回 s^*中的 q_{iv}^*則依然可以保持目標函數值 z^*不變，亦證明的整數性質。

在節省的部分，由於 SDVRP 允許顧客需求由多部車輛共同進行服務，因此有機會能夠求得較 VRP 節省的求解結果。在 Archetti *et al.* [9] 2006 年的研究中證明，當車輛數沒有上限時，SDVRP 的結果最多能較 VRP 的結果節省 50%。並於 Archetti *et al.* [8] 2008 年的研究中證明，在車輛數方面亦有機會節省 50%。

3.2.2 具最小配送量限制之分割配送車輛路線問題特性

分割配送車輛路線問題性質在整數解性質方面已於 Han and Chu [30] 的研究中進行證明，假設 SDVRP-MDA 問題 (P) 具有可行解，則必然有一組配送量變數 $q_{iv} \in Z^+$。以 s^*來表示 (P) 的最佳解，z^*表示為最佳解 s^*的目標函數值，$x_{ij}^v{}^*$與 q_{iv}^*分別代表最佳解所對應的變數值，B 代表顧客點 i 與車輛 v 配對 (i, v) 的集合，$i \in V-\{0\}$, $v \in \{1, ..., m\}$，滿足 $\sum_{j=0}^{n} x_{ij}^v{}^* \ge 1$。假設經過顧客 i 的車輛集合為 $S(i) = \{v \mid (i, v) \in B\}$，車輛 v 所經過的顧客點集合為 $D(v) = \{i \mid (i, v) \in B\}$。考慮以下限制式：

$$\sum_{v \in S(i)} q_{iv} = d_i, \qquad \text{for } i = 1,\ldots, n \qquad (8a')$$

$$\sum_{i \in D(v)} q_{iv} \le Q, \qquad \text{for } v = 1,\ldots, m \qquad (9a')$$

$$q_{iv} \ge \mathrm{MDA}_i \sum_{j=0}^{n} x_{ij}^v{}^* \qquad \text{for } (i, \mathrm{v}) \in B \qquad (11a')$$

將 q_{iv} 以 y_{vi} 代替，可以將式 (8a′)、(9a′) 以及 (11a′) 改寫如下：

$$\sum_{v \in S(i)} y_{vi} = d_i, \qquad \text{for } i = 1,\ldots, n \qquad (8a'')$$

$$\sum_{i \in D(v)} y_{vi} \le Q, \qquad \text{for } v = 1,\ldots, m \qquad (9a'')$$

$$y_{vi} \ge \mathrm{MDA}_i \sum_{j=0}^{n} x_{ij}^v{}^* \qquad \text{for } (i, v) \in B \qquad (11a'')$$

可以發現這些限制式為一個最小成本流量問題 (Minimum Cost Flow Problem) 的限制式。其中式 (9a′′) 的 v 點可視為供給點，式 (8a′′) 的 i 點可視為需求點，y_{vi} 則為流量變數。由於相關變數值是基於問題 (P) 的最佳解，故符合限制式，亦表示此最小成本流量問題具有可行解。在最小成本流量問題中，當限制式的右手邊值 (d_i、Q 以及 MDA$_i$) 皆為整數時，必定存在一組可行解具有整數性質 (Ahuja *et al.* [2])。意即具有一組整數 y_{vi} 值可帶回 q_{iv}*，此時 (P) 的最佳目標函數值依然為 z*，因為 q_{iv} 並未出現於目標函式中。故 SDVRP-MDA 的整數解性質得證。

在潛在效益的的部分，SDVRP-MDA 雖然增加了最小配送輛限制，但依然能夠求得較 VRP 節省的求解結果。在 Xiong *et al.* [48] 的研究中證明，當車輛數沒有上限的前提下，當 $0 < p < 0.5$ 時，SDVRP-MDA 的結果最多能較 VRP 的結果節省 50%。當 $p = 0.5$ 時，則最多有可能節省 33.33%。

3.2.3 具配送次數限制之分割配送車輛路線問題特性

在 SDVRP-LND 的數學模式中，唯一與 SDVRP 不同的地方在於式 (29)。式 (29) 增加了參數 *kmax* 以限制各顧客點被服務的次數上限。*kmax* 雖然與 x_{ij}^v 有關，但 x_{ij}^v 本身即為 0 或是 1 的決策變數，因此並不會影響變數 x_{ij}^v 的整數性質。其他在 q_{iv} 的部分則可引用前述 SDVRP 整數性質的證明方式，證明變數 q_{iv} 的整數性質。根據上述說明可得證 SDVRP-LND 的最佳解具有整數性質。

節省部分，本研究嘗試利用 SDVRP-MDA 中不同 p 值所對應的潛在效益對 SDVRP-LND 可能產生的節省進行推導證明。首先說明 SDVRP-LND 的 *kmax* 值與 SDVRP-MDA 中 p 值所對應的關係如下公式：

$$kmax = \min\left(m, \left\lfloor \frac{1}{p} \right\rfloor\right) \tag{13}$$

透過式 (13) 可算出不同的 p 值所對應的 *kmax* 值，在配合顧客分送次數最多等於車輛數的限制下，本研究將 p 與 *kmax* 兩者對應的關係列於表 3.2 中的第一欄以及第二欄中。

首先說明式 (13) 中求得的 p 與 *kmax* 兩者對應的關係為，SDVRP-MDA 對應 p 值的解集合為 SDVRP-LND 對應 *kmax* 值的解集合的子集合。舉例說明，當 $p = 1/3$ 時，q_{iv} 皆必須大於或等於 $(1/3)\,d_i$，因此 $p = 1/3$ 時 SDVRP-MDA 的可行解中任一節點 i 的分送次數 k_i 最多為三次。意即 SDVRP-MDA 在 $p = 1/3$ 限制下的所有解都包含在 SDVRP-LND 限制 $kmax = 3$ 的解集合中。

SDVRP-LND 潛在效益的推導便是利用上述的概念。利用 p 值與 *kmax* 值彼此之間的關係，再配合 p 值可能產生的潛在效益，便可以間接推導 SDVRP-LND 所能產生的潛在效益。表 3.2 將不同的 p 值、該 p 值對應的 *kmax* 值以及該 p 值可能產生的最大節省效益進行整理。由表 3.2 中可以觀察出，當 $p = 0.0$ 時對應 $kmax = m$，問題即為 SDVRP，最大的成本節省值已由 Archetti *et al.* [9] 的研究證明可達到 50%。故 $kmax = m$ 時的

SDVRP-LND 潛在的成本節省效益可達到 50%。$0 < p \le 0.33$ 對應 $3 \le kmax < m$，依照 Xiong *et al.* [48] 的結論，節省值依然能夠達到 50%。意即 $3 \le kmax < m$ 時，SDVRP-LND 的節省亦可達到 50%。以此類推，$0.33 < p \le 0.5$ 對應 $kmax = 2$，此時可分成兩種情況討論，首先 $0.33 < p < 0.5$ 可產生的潛在效益為 50%，$p = 0.5$ 為 33.33% [48]。由於 $kmax = 2$ 包含上述兩種情況，因此推論 $kmax = 2$ 時，SDVRP-LND 的節省最大依然可達到 50%。綜合由上述的推導結果便可以得知當 $2 \le kmax < \infty$ 時，最大的潛在效益可達到 50%。

表 3.2　*kmax* 與 *p* 對應所得之最大節省表

p	*kmax*	max cost saving	remark
$p = 0.0$	$kmax = m$	≤50%	Archetti *et al*. [9]
$0 < p \le 0.33$	$3 \le kmax < m$	≤50%	Xiong *et al.* [48]
$0.33 < p < 0.5$	$kmax = 2$	≤50%	Xiong *et al.* [48]
$p = 0.5$	$kmax = 2$	≤33.33%	Xiong *et al.* [48]

第四章、SRC+IMP 多重起點啟發式演算法設計

4.1 SRC+IMP 多重起點啟發式演算法整體架構說明

本研究針對 SDVRP、SDVRP-MDA 以及 SDVRP-LND 三個問題，提出一套一體適用的啟發式解法 SRC+IMP (Split-delivery Route Construction + solution IMProvement)。由於 SDVRP-MDA 以及 SDVRP-LND 在分割配送上具有特殊的限制，因此並無法像 SDVRP 能夠保證在最少車輛數的情況下求得可行解。因此，為了讓 SRC+IMP 能夠同時處理不同的分割配送車輛路線問題，本研究所設計之方法屬於未限制車輛數的 SDVRP-UF 方法，而非車輛數受限 SDVRP-LF 方法。求解目標是「總旅行距離最小化」。

一套優良的啟發式演算法在搜尋求解的能力上必須能夠兼具深度 (Intensification) 以及廣度 (Diversification)。因此本研究將這兩種不同的搜尋能力放入演算法的設計中。本研究所設計之 SRC+IMP 在方法架構中主要包含分割配送起始解構建模組 SRC (Split-delivery Route Construction) 以及鄰域改善模組 IMP (solution IMProvement)，再搭配簡單的迴圈控制機制以產生多重起點的廣度搜尋效果。首先介紹分割配送起始解構建模組 SRC，其特點在於能夠針對選擇的顧客點，適應性地選擇插入現行路線 (Node Insertion, NI) 或是新增路線 (Route Addition, RA) 的排入方式構建起始解。SRC 中具有一個引數 α，變化α值可以改變選擇顧客點的方式，故 SRC 搭配α引數的控制迴圈即可構建出多重起始解以增加搜尋起點的多樣性，細部的流程說明於 4.2 節。另外一部份為鄰域改善模組 IMP，IMP 是一套利用多個鄰域搜尋法改善求解的模組，主要是接於 SRC 各個起始解產生後進行改善，是本研究強化深度搜尋的主要方式。因為鄰域搜尋順序變動的方式不同，在 IMP 中包含 VND (Variable Neighborhood Descent) 以及 RVND (Randomized VND) 兩種策略。相關鄰域變動策略以及 IMP 中所使用的各鄰域搜尋法細部說明於 4.3 節介紹。

本研究 SRC+IMP 演算法的整體架構則如表 4.1 的虛擬碼所示。SRC+IMP 中具有一個引數α值，不同的α值將影響 SRC 模組構建起始解起始解的方式。SRC+IMP 中唯一需要設定的參數為$\Delta\alpha$ (line 1)，$\Delta\alpha$搭配迴圈控制可以系統性地變化引數α值進而產生多重解起始解 (lines 4-16)。α的初始值為 0 (line 3)，以遞增$\Delta\alpha$的方式直到$\alpha = 1$ (line 4)。α在 SRC+IMP 中具有相當重要地位，對於求解的效果以及效率皆有影響，相關闡述將於後續 4.2 節進行說明。SRC+IMP 執行時先對最優解成本值 c_{best} 以及α值進行初始化設定 (lines 2-3)，接著開始執行多重起始解構建以及鄰域改善的迴圈。在單一回迴圈中，SRC 設計兩個不同的起始解策略 SRC_0 (line 5) 與 SRC_1 (line 7)，可分別對應產生兩個起始解 s_0 以及 s_1。並利用 IMP 分別進行改善求得 s_0'以及 s_1' (lines 6 與 8)。優化結束後，比較 s_0' 以及 s_1'的成本值 $c(s_0')$ 以及 $c(s_1')$，選擇成本小的解作為本次迴圈的最佳解 s^* (lines 9-12)。接著，若 $c(s^*) < c_{best}$，則將 c_{best} 更新為 $c(s^*)$，最優路線解 s_{best} 記錄為 s^*。反覆執行迴圈直到滿足停止條件後，回傳最優解 s_{best} 以及最優解成本 c_{best}。

表 4.1　SRC+IMP 演算法虛擬碼

```
1   Procedure SRC+IMP (Δα)
2   c_best ← ∞;
3   α ← 0;
4   while α ≤ 1 do
5   |  s_0←SRC(0, α);
6   |  s_0′←IMP (s_0);
7   |  s_1←SRC(1, α);
8   |  s_1′←IMP (s_1);
9   |  if c(s_0′) < c(s_1′)
10  |  |  s* ←s_0′;
11  |  else
12  |  |  s* ←s_1′;
13  |  if c(s*) < c_best
14  |  |  c_best ← c(s*);
15  |  |  s_best ← s*;
16  |  α ← α + Δα;
17  return s_best and c_best;
```

SRC+IMP 為本研究針對 SDVRP 以及其相關問題所提出之求解架構，其中不論是 SRC 或是 IMP 中皆有針對需求分割特性設計的演算流程。由於不同的 SDVRP 相關問題具備的問題限制不同，SRC+IMP 在求解上亦需配合這些限制進行微調以避免違反可行性。相關 SRC+IMP 配合不同的 SDVRP 相關問題的執行細節於後續章節進行說明。

4.2　分割配送起始解構建模組設計 (SRC)

分割配送起始解構建模組 (SRC) 能夠透過輸入不同的α值，構建多個不同且具有分割特性的起始解。α值是用以計算顧客點插入成本的權重參數，此種利用權重的方式影響起始解構建程序的概念在 Solomon [45] 1987 年對具時間窗車輛路線問題 (VRP with Time Windows, VRPTW) 的研究中即被使用，日後亦有其他研究將之應用 [16, 34]，以下說明 SRC 模組的執行流程。

我們利用表 4.2 的 SRC 虛擬碼進行說明。在 SRC 執行時，首先須將參數 *type* 以及 α輸入 (line 1)。*type* 的作用在於決定 SRC 構建起始解的策略為何，在本研究中 *type* 設定為 0 或 1。當 $type = 0$ 時，對應的演算法流程我們稱為 SRC_0，產生的起始解為 s_0；當 $type = 1$ 時，對應的演算法流程我們稱為 SRC_1，產生的起始解稱 s_1。α值則用於稍後插入權重的計算公式之中 (line 18)。參數輸入之後，路線集合 R 初始化為空集合，建立需求未服務完的顧客清單 $L = N$，各顧客 i 未滿足的需求量 $u_i = d_i$ 以及禁制路線集合 F_i 初始為空集合 (lines 2-4)，若求解的問題為 SDVRP-MDA 則需要設定節點的最小配送量 MDA_i，若求解的問題為 SDVRP-LND 則需要初始化節點的配送次數 k_i。完成初始化後，挑選清單 L 中距場站最遠顧客點 f 建立第一條路線 $r_1 = (0-f-0)$ (lines 5-7)。若是求解 SDVRP-LND，則須設定 k_f 為 1 (line 8)。對於 r_1 的處理方式上，不同的策略 (*type*) 將有不同。當 $type = 0$ 的 SRC_0 時，r_1 構建完成後即加入現行路線集合 R 中，以供後續構建路

表 4.2　SRC 虛擬碼

```
1   Procedure SRC(type, α)
2   Initialize R ← ϕ, L ← N;
3   for each i in L do
4   |  set u_i ← d_i, F_i ← ϕ, MDA_i ← ⌈p × d_i⌉ and k_i ← 0;
5   f ← argmax_{i∈L}{c_0i};
6   r_1 ← route(0 – f – 0);
7   Update L ← L\{ f }, Load_r1 ← u_f, q_fr1 ← u_f; u_f ← 0;
8   if (P) = SDVRP-LND, then k_f ← 1;
9   if Type = 0
10  |  R ← R ∪ {r_1};
11  while L ≠ ϕ do
12  |  W* ← ∞;
13  |  for each i in L do
14  |  |  IC_i ← ∞;
15  |  |  for each r in R/F_i do
16  |  |  |  I_ir ← get the cheapest cost of inserting node I into route r;
17  |  |  |  if I_ir < IC_i
18  |  |  |  |  Update r_i ← r, IC_i ← I_ir;
19  |  |  W_i ← αIC_i – (1–α)c_0i;
20  |  |  if W_i < W*
21  |  |  |  Update W* ← W_i, b ← i, t ← r_i;
22  |  L ← L\{b};
23  |  R ← NIRA(b, t) ;
24  if Type = 1
25  |  R ← R ∪ {r_1};
26  return s_type from R;
```

線時進行插入其他顧客點 (lines 9-10)；反之若 *type* = 1 的 SRC$_1$ 時，則不會立即將 r_1 新增到 R 中。接著，更新路線 r_1 構建結束後相關的數值，包括將顧客 f 於清單 L 中移出、路線 r_1 的乘載量 $Load_{r1}$、路線 r_1 服務顧客 f 的需求量 q_{fr1} 以及顧客 f 未服務的需求量 u_f。

完成第一條路線的構建後，開始選擇顧客點插入的流程 (lines 11-23)。首先初始化最佳的 W^* 值為無限大 (line 12)。接著，還列於清單 L 中的各顧客點 i 開始計算插入現行解的最省成本 IC_i，IC_i 為顧客點 i 對應在路線集合 $R\backslash F_i$ 中最小插入成本，意即 $IC_i = \min_{r\in R/Fi} I_{ir}$，$I_{ir}$ 為顧客點 i 在路線 r 中的最小插入成本，IC_i 對應的插入路線則以 r_i 表示 (lines 16-18)。求得 IC_i 之後，利用顧客點 i 的最省插入成本 IC_i 以及與場站之距離 c_{0i} 計算該顧客點 i 的插入權重 W_i (line 20)，計算公式如下：

$$W_i = \alpha IC_i - (1-\alpha)\, c_{0i} \tag{14}$$

公式 (14) 中 α 為執行 SRC 時所引入的權重參數。所有顧客點皆完成計算後，選擇權重值最小的顧客點 $b = \text{argmin}_{i\in L}\, W_i$ (lines 20-21)，並於清單 L 中移出 (line 22)。將選出的顧

客點 b 以及其對應的最小插入成本路線 t 作為引數，呼叫執行 *NIRA* (Node Insertion and Route Addition) 模組進行現行解插入 (line 23)。不斷的重複執行選點插入的動作，直到 $L = \phi$。而後，若 $type = 1$ (SRC_1)，則將 r_1 放入路線集合 R 中 (lines 24-25)。最後以路線集合 R 輸出路線解 (line 26)。

在 SRC 起始解構建模組中，SRC_1 是一個特別的想法，如此設計的原因有兩個：(1) 期望能讓起始解 s_0 以及 s_1 在路線型態上產生差異，提高起始解的多樣性；(2) 由於構建過程中未允許其他任何顧客點插入至 r_1，因此後續開始執行 IMP 改善時，r_1 有相對多的剩餘容量使得交換搜尋更有彈性，進而提高深度優化的可能性。

W_i 公式的權重項目包含顧客點 i 目前的「最省插入成本」以及「與場站距離」兩者，選擇最小 W_i 值之顧客優先插入則表示，該顧客為插入成本低且距離場站較遠之顧客。插入成本低為一般的「貪婪」概念，表示將該顧客排入現行解所增加的成本較少。另一方面，選擇距離場站較遠之顧客優先插入，是因為較遠的顧客一般對於解成本會造成較大的影響。優先將其排入現行路線，能夠讓路線解先有一個基本的雛形。後續插入距離場站較近的顧客點時，自然能夠配合當下的路線架構，配置在適當的位置。

α值為 SRC 用以產生多重起始解的參數，不同的α值會改變 W_i 公式計算的結果，對應選擇的插入點亦會改變。在 SRC 中，α值從 0 開始，透過迴圈的方式不斷遞增$\Delta\alpha$直至 1 為止。在策略的層面上，當$\alpha = 0$ 時，插入點的選擇是以與場站之距離為選擇的標準，越遠的顧客點越先考慮插入，構建起始解的方式為最遠插入法。反之當$\alpha = 1$ 時，是以插入成本作為選擇插入點的標準，成本越小的顧客點越先插入，因此在選點策略上為最省插入法。簡而言之，不同的α值會改變插入時選點的策略，進而產生不同的起始解。

$\Delta\alpha$為 SRC 中唯一的參數，其影響在於 SRC 產生起始解的數量多寡，當$\Delta\alpha$設定的越小，對應不同的α會越多，產生的起始解數量也越多。舉例而言，當$\Delta\alpha = 0.05$ 時，SRC 過程中將對應出 $(1 / 0.05) + 1 = 21$ 個α值，$\Delta\alpha = 0.01$ 時則可產生 101 個α值。搭配前述兩種不同的起始解策略 (*type*)，當$\Delta\alpha = 0.05$ 以及 0.01 時，便可產生 42 個以及 202 個起始解。

SRC 利用權重公式選擇出欲排程的顧客點後，即進入 *NIRA* 模組對該顧客點進行排程。*NIRA* 是一個以插入法作為基礎的排程方法，並且具能夠適應性的選擇排程策略。此外，*NIRA* 將考慮在排程路線的同時允許適當的需求點進行分割，使得 SRC 能夠構建出具有分割性質的起始解。亦由於 *NIRA* 有分割特性的考慮，因此在不同型態的 SDVRP 相關問題使用時，將個別對應問題的限制進行微調，相關細節於後續小節進行說明。

4.2.1 *NIRA* 模組設計

本研究參考傳統最遠起點最省插入法 (Farthest-start Cheapest Insertion)，修改後提出 *NIRA* (Node Insertion and Route Addition) 節點排程模組。傳統最省插入法需決定構建策略採用的是「循序式」或是「平行式」。「循序式」以車容量為限，反覆插入同一路線，直至無可行插入點，始考慮新增路線服務尚未排程的顧客；「平行式」則需事先計算使用

的車輛數，選出對應車輛數的種子點並產生種子路線，供後續尚未排程的顧客進行插入。本研究提出之 *NIRA* 構建策略非循序也非平行構建，而是適應性的選擇「插入節點」或「新增路線」兩者中較佳的方案執行，*NIRA* 執行的虛擬碼如表 4.3 所示。

表 4.3　*NIRA* 虛擬碼

```
1   Procedure NIRA(i, r)
2   if (P) = SDVRP-LND, then k_i ← k_i + 1;
3   if I_ir < 2c_0i
4   |  r ← insert i into route r;
5   |  Update q_ir ← u_i, Load_r ← Load_r + q_ir, u_i ← 0;
6   |  if Load_r > Q
7   |  |  O_r ← Load_r − Q;
8   |  |  v ← argmin_{p∈r} {c_0p | q_pr ≥ O_r};
9   |  |  if (P) = SDVRP
10  |  |  |  RA_v ← O_r;
11  |  |  if (P) = SDVRP-MDA
12  |  |  |  RA_v ← max (O_r, MDA_v);
13  |  |  |  if q_vr − RA_v < MDA_v
14  |  |  |  |  RA_v ← q_vr;
15  |  |  if (P) = SDVRP-LND
16  |  |  |  if k_v < kmax
17  |  |  |  |  RA_v ← O_r;
18  |  |  |  else
19  |  |  |  |  RA_v ← q_vr;
20  |  |  Load_r ← Load_r − RA_v;
21  |  |  u_v ← u_v + RA_v and q_vr ← q_vr − RA_v;
22  |  |  Update L ← L ∪ {v}, F_v ← F_v ∪ {r};
23  |  |  if (P) = SDVRP-LND and q_vr = 0, then k_v ← k_v − 1;
24  else
25  |  l ← route(0 − i − 0);
26  |  Update q_il ← u_i, Load_l ← u_i, u_i ← 0;
27  |  R ← R ∪ {l};
28  return R;
```

首先，*NIRA* 需要輸入顧客點 i 以及其對應的最小插入成本路線 r (line 1)。接著若目前求解的問題 (P) 為 SDVRP-LND 時，則更新顧客 i 的配送次數 (line 2)。判斷最小插入成本 I_{ir} 是否小於為顧客 i 新增路線的成本 $2c_{0i}$ (line 3)，是則執行節點插入 (lines 4-23)；反之，則執行新增路線的程序 (lines 25-27)。首先說明新增路線程序，當 $I_{ir} \geq 2c_{0i}$ 時，表示新增路線服務為更節省的方案，因此新增路線 l 服務顧客 i (line 25)，更新相關資訊後將路線 l 放入現行路線集合 R 中 (lines 26-27)。反之，當 $I_{ir} < 2c_{0i}$ 時則執行節點插入程序，將顧客 i 未服務的需求量 u_i 插入路線 r 中最省的位置並更新相關資訊 (lines 4-5)。插入後，檢查路線 r 是否有超載的情形 (line 6)，是則進入移出需求的步驟。移出需求時，首先計算路線 r 目前的超載量 O_r (line 7) ，並嘗試於路線 r 中移出大於或等於 O_r 的配送量以保持路線 r 的可行性。*NIRA* 將於路線 r 中挑選一個顧客 v 移出服務量，挑選公式如下式：

$$v = \text{argmin}_{p \in r}\{c_{0p} | \, q_{pr} \geq O_r\} \quad (15)$$

式 (15) 中，q_{pr} 為顧客 p 在路線 r 上的服務量，p 為路線 r 上服務的顧客點，即 $p \in r$。利用式 (15) 可選出距離場站最近且在路線 r 上的服務量 q_{vr} 大於等於 O_r 的顧客 v (line 8)。

選出顧客 v 後，則要決定移出量 RA_v 的大小。RA_v 依不同的分割問題進入對應的計算程序，主要可以分成求解的問題模式 (P) 為 SDVRP (lines 9-10)、SDVRP-MDA (lines 11-14) 以及 SDVRP-LND (lines 15-19) 三個部分。首先若問題 (P) = SDVRP，則顧客 v 需要移出的量 RA_v 直接設為超載量 O_r (line 10)。SDVRP-MDA 的部分則需考慮移出 RA_v 的大小必須同時大於等於超載量 O_r 以及顧客 v 的 MDA_v，使路線 r 符合可行性之外，並讓之後服務 v 的配送量滿足 MDA 的限制 (line 12)。RA_v 決定之後，另需檢查移出 RA_v 後，點 v 剩餘在路線 r 中的配送量是否無法滿足 MDA 的限制 (line 13)，若是則 RA_v 調整為點 v 目前在路線 r 的配送量 q_{vr}，意即將點 v 全部移出 (line 14)，若否則不需調整。SDVRP-LND 的部分則需考量點 v 目前已配送的次數 k_v 是否還未達到 $kmax$ 的上限 (line 16)，若是則設定 $RA_v = O_r$ (line 17)，若否則 $RA_v = q_{vr}$ (line 19)。簡而言之，決定移出量 RA_v 的原則即是，若能夠部分移出，則移出能使路線解保持可行性的最小量；否則將顧客 v 在路線 r 的服務量 q_{vr} 全部移出。RA_v 計算完畢後，則執行服務量的移出以及更新相關的資訊 (lines 20-21)。完成移出後，將顧客點 v 加回 L 清單中；將路線 r 新增到 F_v 中，以避免剩餘的 u_v 再次插入路線 r 中 (line 22)。完成節點插入或是路線新增後，*NIRA* 將現行路線的集合回傳至 SRC 繼續構建路線 (line 28)。

NIRA 的特色在於其適應性的構建方式，其能夠根據 I_{ir} 與 $2c_{0i}$ 的大小決定插入現有路線或是新增路線。在插入現有路線的部分，需求量會以不考慮車容量限制的情況下，完全安排至最佳的位置。一旦顧客 i 插入後的車輛乘載量發生超載的情況，則透過選擇特定顧客點，並以移出其部分或是全部配送量的方式維持該路線的可行性。移出點選擇的方式是以路線中最靠近場站同時服務量大於等於超載量者執行，原因是與場站越接近的顧客後續再次進行排程時，產生的成本相對較小。此種選擇方式亦呼應 Chen *et al.* [17] 研究的結論，其方法論 EMIP+VRTR 中 EMIP 的部分便是針對各路線的起末兩點如何配置需求量所設計的整數規劃模式。而路線的起末兩點通常為距離場站較接近的顧客點，故本研究在 NIRA 中將這樣的概念延伸至分割點的選擇機制中。

4.3 鄰域改善模組設計與應用 (IMP)

本研究利用 SRC 模組產生起始解 s_0 或 s_1 之後，即利用鄰域改善模組 IMP 進行求解改善。改善的階段中，本研究共採用五種鄰域搜尋法，包含了四個傳統鄰域搜尋法以及本研究針對問題特性所提出的分割配送節點驅逐鏈交換法 *SNEC* (Split-delivery Node Ejection Chains)。

鄰域改善模組 IMP 是一套採用多個不同的鄰域搜尋法進行搜尋改善的流程，不同的鄰域使用順序將造成不同的求解結果。基於鄰域使用次序的不同，本研究以兩種策略

進行應用，分別是鄰域變動次序固定的 VND 以及鄰域變動次序隨機的 RVND 兩種不同的變動策略，相關機制的執行方式於 4.3.1 節說明。此外，傳統鄰域搜尋法在應用於 SDVRP 與其衍生問題時應有對應調整，調整之細節於 4.3.2 節說明。最後，本研究針對問題特性設計之 *SNEC* 交換法，其交換概念與執行方式則於 4.3.3 節進行說明。

4.3.1 IMP 模組之變動鄰域策略：VND 與 RVND

同時使用多種不同的鄰域進行最佳化問題的改善機制是為近年最佳化問題啟發式研究的主流。本研究利用 SRC 模組搭配不同的權重參數以及策略產生起始解之後，接著以改善模組 IMP 將解改善至區域最佳解。由於 IMP 中同時使用多個鄰域搜尋法，變動鄰域策略便是模組應用需要考慮的問題。變動鄰域策略意即是不同搜尋法間應用的次序策略，不同的應用次序將因搜尋的路徑不同使得最後的求解結果造成影響。以下便說明本研究所使用的 VND 以及 RVND 兩套變動鄰域搜尋策略。

變動鄰域下降 VND 的搜尋方式最早由 Mladenović and Hansen [37] 於 1997 年提出，VND 是一套利用多個不同鄰域進行搜尋的執行程序，在求解組合最佳化問題中經常被使用。VND 的特點在於只允許現行解往改善 (優化) 的鄰域解進行移動或是更新，當 VND 停止搜尋時所找到的解同時為所有採用的鄰域搜尋法的區域最佳解，求解結果的品質上具有更高的機率優於僅使用單一搜尋法進行求解的結果。VND 變動鄰域搜尋法可以利用表 4.4 的 VND 流程虛擬碼進行說明。在表 4.4 中，VND 在執行時需要給予一個欲進行搜尋的解 s 以及排序欲採用的鄰域搜尋法 (line 1)。VND 將會不斷變換鄰域搜尋法對現行解 s 進行改善，直到所有鄰域無法對現行解改善則停止 (lines 2-10)，最後輸出現行解 s (line 11)。各個鄰域搜尋法 N_h 在改善時會反覆進行搜尋，直到無任何鄰域解 $s' \in N_h(s)$ 優於現行解 s 才會停止，並更換下一個鄰域搜尋法 (lines 4-9)。

表 4.4　VND 流程虛擬碼

```
1   Procedure VND(s, N1, N2, …, Nhmax)
2   do
3   |  Set h ← 1 and improved ← 0;
4   |  while h ≤ hmax
5   |  |  Find the best neighbor s′ of s, s′ ∈ Nh(s);
6   |  |  if c(s′) < c(s)
7   |  |  |  Update s ← s′ and improved ← 1;
8   |  |  else
9   |  |  |  h ← h + 1;
10  until improved = 0;
11  return s;
```

Subramanian *et al.* [46] 發現了傳統 VND 在鄰域排序上的困擾，因此於 2012 年提出隨機鄰域搜尋順序的架構，稱之為隨機變動鄰域下降 RVND。RVND 與傳統 VND 不同的地方在於，RVND 是以隨機的方式選取鄰域搜尋法進行鄰域改善。表 4.5 為 RVND 執行虛擬碼，與 VND 相同的，RVND 需要給予一個欲進行搜尋解 s 以及定義採用的鄰域搜尋法 $N_1, N_2, \ldots, N_{h\max}$ (line 1)。接著，定義鄰域集合 LS，並將所有的欲採用的鄰域搜尋

法放入 *LS* 中 (line 2)。接著，從 *LS* 隨機挑選一個鄰域搜尋法 N_h (line 4)，並利用該鄰域對解 *s* 搜尋改善解 $s' \in N_h(s)$ (line 5)。若是能找到 s'，則更新解 *s* 為 s' 並將 *LS* 集合重新設為所有的鄰域搜尋法 (lines 6-7)。反之，若無找到 s'，則表示解 *s* 已經是鄰域 N_h 的區域最佳解，N_h 已無法改善之，因此將於 *LS* 集合中移除 (lines 8-9)。RVND 的整個流程會反覆地從 *LS* 集合中隨機挑選鄰域搜尋法，直至 *LS* 為空集合停止 (lines 3-9)，並輸出改善後的最終結果 (line 10)。

表 4.5　RVND 流程虛擬碼

```
1   Procedure RVND(s, N1, N2, …, Nhmax)
2   Initialize LS ← {N1, N2, …, Nhmax };
3   while LS ≠ ϕ
4   |  Select a neighborhood Nh∈LS at random;
5   |  Find the best neighbor s′ of s, s′∈Nh(s);
6   |  if c(s′) < c(s)
7   |  |  Update s ← s′ and LS ← {N1, N2, …, Nhmax};
8   |  else
9   |  |  Update LS ← LS \ Nh;
10  return s;
```

由於 VND 以及 RVND 在鄰域變動的流程上不同，亦可能產生不同的求解結果。為將之進行區分，本研究求解架構 SRC+IMP 應用時，根據不同的變動鄰域策略可細分為 SRC+VND 以及 SRC+RVND 兩種。本研究在後續標竿題庫的求解績效測試時，將分別對 SRC+VND 以及 SRC+RVND 兩種不同的求解方式皆進行測試，並將測試結果進行比較分析。

4.3.2　傳統鄰域搜尋法(鄰域編號 $N_1 - N_4$)

本研究採用的鄰域改善模組中包含四個傳統的鄰域搜尋法，分別是 2-*opt* (Lin and Kernighan [36])、*Or-opt* (Or [39])、2-*opt** (Potvin and Rousseau [41])以及 *λ-interchanges* (Osman [40])。其中 2-*opt* 以及 *Or-opt* 屬於路線內的鄰域搜尋法，在 VND 中鄰域編號設為 N_1 以及 N_2。由於 2-*opt* 以及 *Or-opt* 交換時不會牽涉需求分割的處理，因此可以直接引用。2-*opt** 以及 *λ-interchanges* 則屬於路線間的交換改善法，在本研究針對 SDVRP 相關問題改善時應有適當調整。相關調整說明如下：

(1)　2-*opt** (鄰域編號 N_3)

2-*opt** 最早由 Potvin and Rousseau [41] 於 1995 年提出應用於改善 VRPTW，是利用路線間的節線交換進行搜尋的一種鄰域搜尋法，2-*opt** 的操作示意圖如圖 4.1 所示。圖 4.1 (a) 中，2-*opt** 首先挑選兩條路線，r_1 與 r_2，並嘗試將中 r_1 的節線 (i, s_i) 以及 r_2 的節線 (j, s_j) 移除，再挑選不同的節線將兩路線重新連接。重新連接的方式有兩種，如圖 4.1 (b) 與圖 4.1 (c) 所示。

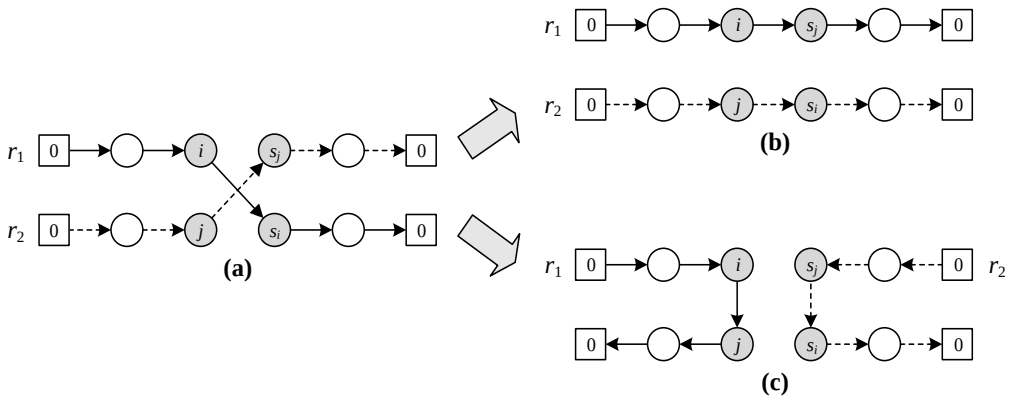

圖 4.1　本研究 2-*opt**交換法示意圖

圖 4.1 (b) 的連接方式是重新連接節線 (i, s_j) 與 (j, s_i) 的結果，這樣的接法為 2-*opt** 提出時所建議的接法。由於沒有線段需要進行轉向的關係，本研究將圖 4.1 (b) 的連接方式稱為順向接法。順向接法能夠保持路線中原本的顧客服務順序，適用於排程時有先後順序限制問題 (e.g. VRPTW)。圖 4.1 (c) 的連接方式是重新連接節線 (i, j) 與 (s_j, s_i) 的結果，這樣的接法需要分別反轉場站到點 j 以及場站到點 s_j 的路段方向。本研究 2-*opt** 交換時會同時比較圖 4.1 (b) 與圖 4.1 (c) 兩種交換結果的優劣，並選擇較佳的方案執行。交換後，同一條路線內可能存在兩份需求量屬於同一個顧客，此時會選擇將移除後路線成本節省較多的需求合併至另一份需求中。

(2)　*λ-interchanges* (鄰域編號 N_4)

λ-interchanges 節點交換法的概念最早由 Osman [40] 提出，泛指在兩條路線間選擇數個顧客互相交換至對方路線的節點交換法。為了能夠更貼切的表達*λ-interchanges* 交換法的含意，Cordeau and Laporte [19] 於 2005 年將*λ-interchanges* 改由 (λ_1, λ_2) 表示，意指兩條路線中分別選擇λ_1與λ_2個節點交換至對方路線。λ_1與λ_2的關係式為 $1 \le \lambda_1 \le \lambda$ 且 $0 \le \lambda_2 \le \lambda_1$。換言之，當$\lambda = 2$便表示含有五種節點交換法，分別為(1, 0)、(1, 1)、(2, 0)、(2, 1) 以及 (2, 2)。其中依據節點移動的方式又分成移位 (*shift* (1, 0) 與 *shift* (2, 0)) 與交換 (*swap* (1, 1)、*swap* (2, 1) 與 *swap* (2, 2)) 兩類。

本研究在*λ-interchanges* 的部分僅採用 *swap* (2, 1) 以及 *swap* (2, 2) 兩種交換方式，未採用 *shift* (1, 0)、*swap* (1, 1) 以及 *shift* (2, 0)。未採用 *shift* (1, 0)、*swap* (1, 1) 的原因是本研究新提出之 *SNEC* 鄰域搜尋法所搜尋的鄰域範圍已包含 *shift* (1, 0) 以及 *swap* (1, 1) 兩個鄰域搜尋法所能搜尋的鄰域。另外，未使用 *shift* (2, 0) 的原因有二，一是 *shift* (2, 0) 可能能夠透過連續的兩次 *shift* (1, 0) 來達成；二是分割需求的路線解在車容量的利用上較有效率，意即各車輛的剩餘空間較小，而 *shift* (2, 0) 又屬於在路線乘載量變化較大的交換法，交換的成功率較低。

4.3.3　分割配送節點驅逐鏈交換法 *SNEC* (鄰域編號 N_5)

除了上述四種傳統鄰域交換法以外，本研究利用 SDVRP 相關問題允許需求分割的

特性，提出分割配送節點驅逐鏈交換法 (Split-delivery Node Ejection Chain, *SNEC*)。本研究提出之 *SNEC* 是一種適應性程度相當高的交換法，當節點 i 一旦於路線 r_1 移動至 r_2，之後的交換過程則視 r_2 路線的情況而定，包含是否需要再於 r_2 中移動節點 j 至路線 r_3 以及節點 j 的需求量是整份移動或是創造新的分割都是有可能的交換選項。*SNEC* 是一個創新的適應性交換方式，至多可使得兩個不同的節點在三條路線間進行移動，圖 4.2 為整體 *SNEC* 適應性交換過程的示意圖。

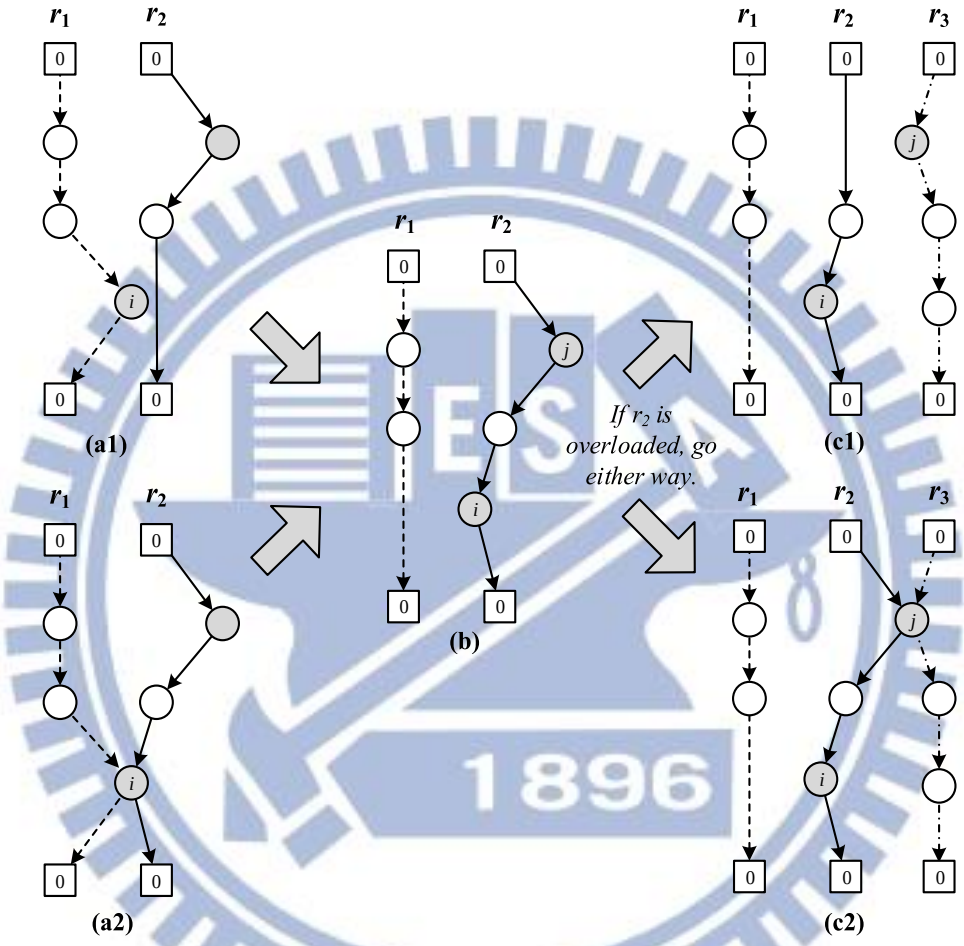

圖 4.2　*SNEC* 交換法示意圖

SNEC 首先將選擇一個需求點 i，並把點 i 於路線 r_1 驅逐至 r_2。如圖 4.2 (a1) 以及圖 4.2 (a2)，需求點 i 可以是完整的顧客需求，亦可以是同時被 r_1 以及 r_2 分割配送的需求。如圖 4.2 (b)，需求點 i 將插入至 r_2 的最省位置，此時 r_2 若未發生超載的情況，則新的路線結果為 *SNEC* 的鄰域解。反之，若 r_2 發生超載的情況，*SNEC* 則會繼續考慮移動 r_2 路線上點 j 的需求量至另一條路線 r_3。圖 4.2 (c1) 以及圖 4.2 (c2) 顯示需求點 j 於路線 r_2 移動至 r_3 的兩種可能結果。圖 4.2 (c1) 表示當 r_3 的剩餘容量大於等於 q_{jr2} 的情況，需求點 j 將完全移動至 r_3 的最省插入位置。反之，當 r_3 的剩餘容量小於 q_{jr2} 時，需求量 q_{jr2} 無法完全移動至 r_3。此時，以填滿 r_3 的剩餘容量為原則將 q_{jr2} 進行分割至 r_3 的最省插入位置，如圖 4.2 (c2) 所示。整體的 *SNEC* 的細部流程虛擬碼可參考表 4.6 *SNEC* 虛擬碼。

表 4.6　*SNEC* 虛擬碼

```
1  Procedure SNEC(s)
2  for each route r1∈R
3    for each node i∈r1
4      Set Δcost_SNEC ← 0, Load_r1 ← Load_r1 – q_ir1 and ShiftType ← 0;
5      for each route w1∈R/{r1}
6        Δcost_i ← the cost increment of ejecting i from r1 to the best position in w1;
7        if Δcost_i < 0
8          if Load_w1 + q_ir1 ≤ Q
9            if Δcost_i < Δcost_SNEC
10             Update Δcost_SNEC ← Δcost_i;
11             Update ShiftType ← 1 and r2 ← w1;
12         else
13           O_w1 ←(Load_w1 + q_ir1) – Q ;
14           for each node v∈w1/{i}
15             if q_vw1 ≥ O_w1
16               for each route w2∈R/{w1}
17                 SQ_w2 ← Q – Load_w2;
18                 if SQ_w2 ≥ O_w1
19                   if SQ_w2 ≥ q_vw1
20                     Δcost_v ← the cost increment of shifting v entirely from w1 to the best position in w2;
21                     if Δcost_i + Δcost_v < Δcost_SNEC
22                       Update Δcost_SNEC ← Δcost_i + Δcost_v;
23                       Update ShiftType ← 2, r2← w1, r3← w2 and j ← v;
24                   else
25                     if (P) = SDVRP
26                       g ←SQ_w2;
27                     if (P) = SDVRP-MDA
28                       if min( SQ_w2, q_vw1 – MDA_v) < max(O_w1, MDA_v – q_vw2)
29                         g ← 0; //denote the shift is infeasible
30                       else
31                         g ← min(SQ_w2, q_vw1 – MDA_v);
32                     if (P) = SDVRP-LND
33                       if q_vw2 = 0 and k_v = kmax
34                         g ← 0; //denote the shift is infeasible
35                       else
36                         g ←SQ_w2;
37                     if g ≠ 0
38                       Δcost_v ← the cost increment of shifting v partly from w1 to the best position in w2;
39                       if Δcost_i + Δcost_v < Δcost_SNEC
40                         Update Δcost_SNEC ← Δcost_i + Δcost_v;
41                         Update ShiftType ← 3, r2 ← w1, r3 ← w2, j ← v and Δq_jr2 ← g;
```

表 4.6　*SNEC 虛擬碼* (續)

```
42      if ShiftType = 0
43          Update Load_r1 ← Load_r1 + q_ir1;
44      else
45          if (P) = SDVRP-LND and q_ir2 > 0, then k_i ← k_i − 1;
46      if ShiftType = 1
47          Shift i from route r1 to the best position in route r2;
48          Update Load_r2 ← Load_r2 + q_ir1, q_ir2 ← q_ir2 +q_ir1 and q_ir1 ←0;
49      if ShiftType = 2
50          Shift i from route r1 to the best position in route r2;
51          Shift j from route r2 to the best position in route r3;
52          if (P) = SDVRP-LND and q_jr3 > 0, then k_j ←k_j − 1;
53          Update Load_r2 ← Load_r2 + q_ir1− q_jr2 and Load_r3 ← Load_r3 + q_jr2;
54          Update q_ir2 ← q_ir2 + q_ir1, q_ir1 ← 0, q_jr3 ← q_jr3 + q_jr2 and q_jr2 ← 0;
55      if ShiftType = 3
56          Shift i from route r1 to the best position in route r2;
57          Partly shift j from route r2 to the best position in route r3;
58          if (P) = SDVRP-LND and q_jr3 = 0, then k_j ←k_j + 1;
59          Update Load_r2 ← Load_r2 + q_ir1− Δq_jr2 and Load_r3 ← Load_r3 + Δq_jr2;
60          Update q_ir2 ← q_ir2 + q_ir1, q_ir1 ← 0, q_jr3 ← q_jr3 + Δq_jr2 and q_jr2 ← q_jr2 − Δq_jr2;
61  return s from R;
```

表 4.6 中，*SNEC* 執行時首先選擇一條路線 r_1 以及 r_1 路線中的一個需求點 i (lines 2-3)，並嘗試將 i 移動至其他路線 r_2 (以 w_1 進行挑選，line 5)，計算移動 i 的成本$\Delta cost_i$ (line 6)。此時，若第二條路線 r_2 的乘載量並未超載，則 *SNEC* 於此停止 (lines 8-11)。否則，於 r_2 路線中尋求點 i 以外的另一個節點 j (以 v 進行挑選，line 14)，並嘗試移動點 j 的需求量至路線 r_3 使得路線解可行。整體 *SNEC* 移動的方式大致上可分成 4 種方式並在搜尋過程中利用 *Shifttype* 的變數進行記錄。搜尋完成時，若 *Shifttype* = 0，表示整個搜尋並沒有找到可以改善解的移動方式；若 *Shifttype* = 1，表示點 i 於 r_1 的服務量移至 r_2 後，路線解可行且產生改善 (lines 9-11，對應圖 4.2 (a1) 或 (a2) 到圖 4.2 (b) 的交換方式)；若 *Shifttype* = 2，表示點 i 於 r_1 的服務量移至 r_2，點 j 於 r_2 的服務量完全移至 r_3 後，路線解可行且產生改善 (lines 21-23，對應圖 4.2 (a1) 或 (a2) 到圖 4.2 (c1) 的交換方式)；若 *Shifttype* = 3，表示點 i 於 r_1 的服務量移至 r_2，點 j 於 r_2 的服務量部分移至 r_3 後，路線解可行且產生改善 (lines 39-41，對應圖 4.2 (a1) 或 (a2) 到圖 4.2 (c2) 的交換方式)。

在 *SNEC* 中，*Shifttype* = 1 以及 2 兩種交換方式，對顧客點的需求移動後不會有新的分割產生，因此不論是在 SDVRP、SDVRP-MDA 或是 SDVRP-LND 皆不會有產生不可行之處。而 *Shifttype* = 3 的交換方式可能使點 v (點 j 的候選點) 的需求發生新的分割，因此在測試交換的成本變化前須要先應該不同的分割配送問題考慮其移動的可行性。表 4.6 中 lines 25-36 便是對應不同分割問題時交換可行性偵測程序。首先，若是求解 SDVRP (line 25)，準備移動至 r_3 的需求量 g 即設為路線 r_2 目前的超載量 O_{r2} (line 26)；若是求解 SDVRP-MDA (line 27)，則需事先判斷 r_2 最小需要移出的量必須同時滿足下列條件：(1) 大於目前的超載量以及 (2) 移動至另一條路線後，該路線上的配送量必須大於 MDA_v，

故可利用式 $\max(O_{w1}, \text{MDA}_v - q_{vw2})$ 另一方面，需要考慮是否能夠承接準備移過來的量。r_3 最大允許接受的量必須同時滿足下列條件：(1) 不可超過目前 r_3 的剩餘容量 SQ_{r3} 以及 (2) 由於為部分移出，因此留在 r_2 的量需要維持大於 MDA_v，故可利用式 $\min(SQ_{w2}, q_{vw1} - \text{MDA}_v)$ 求得 r_3 最大可接受的需求量。接著判斷 r_3 最大可接受的需求量是否小於 r_2 最小需要移出的量 (line 28)。若是，則表示無可行的移動方式，g 即設為 0；反之，g 即設為 r_3 最大允許接受的量 $\min(SQ_{w2}, q_{vw1} - \text{MDA}_v)$。再者，若是求解 SDVRP-LND (line 32)，則需事先判斷路線 r_3 上是否無點 v 的需求量且點 v 目前的配送次數是否已達到上限 (line 33)。若是，則表示點 v 需求無法再次分割，g 即設為 0 (line 34)；反之，g 即設為 SQ_{w2} (line 35)。利用 lines 26-35 對應不同的問題計算 g 之後，若 g 不等於 0 則表示交換後的解依然維持可行性 (line 37)，若交換後的成本可獲得改善，則記錄相關後續進行更新時所需要的資訊。

特別說明，*SNEC* 在交換節點以及路線的選擇上具有以下三個限制條件：(1) $r_1 \neq r_2$、(2) $j \neq i$，且 $q_{jr2} \geq O_{r2}$ 以及 (3) $r_2 \neq r_3$，且 r_3 的剩餘容量大於 O_{r2}。設定條件 (2) 以及 (3) 的目的在於能夠在點 j 被驅逐至 r_3 後，使得整個解滿足可行性的要求。此外，特別提及在 $r_3 = r_1$ 是被允許的。

由於 *SNEC* 在設計上強調的是適應性的彈性移動交換，相對其他 SDVRP 的鄰域搜尋法能夠探索的鄰域範圍更為寬廣，並且更為一般化。具體而言，*SNEC* 在特定的情況下能夠視同其他鄰域搜尋法，包含 Ho and Haugland [31] 的 relocate、exchange 以及 relocate split；Aleman *et al.* [3] 的 shift、swap 以及 *shift**；Derigs *et al.* [22] 的 EXCHANGE；Berbotto *et al.* [14] 的 *Relocate*、*Exchange*、*ExchSplit* 以及 *DelSplitNew*；與 Silva *et al.* [44] 的 *Shift* (1, 0)、*Swap* (1, 1) 以及 *Swap* (1, 1)。表 4.7 彙整五種 *SNEC* 的特殊情況，以及各情況所對應的文獻交換法。舉例而言，情境 4 為圖 4.2 中 (a2) 到 (c2) 的交換過程，並且在 $r_3 = r_1$ 的條件下，*SNEC* 包含 relocate split [31] 以及 *ExchSplit* [14] 兩個交換法的鄰域。由於 *SNEC* 能夠搜尋廣大的鄰域範圍，因此我們相信 *SNEC* 在 IMP 求解改善模組中將會是一個優化求解結果的重要鄰域搜尋法。

表 4.7 *SNEC* 在特殊情況下與其他文獻交換法對照表

	特殊情況			
	起始情況	起始情況	條件	對應之文獻交換法
1	(a1)	(b)	–	relocate[1], shift[2], *Relocate*[4], *Shift* (1, 0)[5]
2	(a1)	(c1)	$r_3 = r_1$	exchange[1], swap[2], EXCHANGE[3], *Exchange*[4], *Swap* (1, 1)[5]
3	(a1)	(c2)	$r_3 = r_1$	*shift**[2], *Swap* (1, 1)*[5]
4	(a2)	(c2)	$r_3 = r_1$	relocate split[1], *ExchSplit*[4]
5	(a2)	(c2)	$r_3 \neq r_1$	*DelSplitNew*[4]

[1] Ho and Haugland [31], Aleman *et al.* [3], [3] Derigs *et al.* [22], [4] Berbotto *et al.* [14], [5] Silva *et al.* [44]

第五章、分割配送相關車輛路線問題標竿題庫與參數設定

5.1 標竿題庫說明

本研究利用 SDVRP 以及 SDVRP-MDA 的標竿題庫作為測試 SRC+IMP 求解架構的績效測試例題。在 SDVRP 以及 SDVRP-MDA 各選擇兩套題庫進行測試。SDVRP 部分選擇 Chen *et al.* [17] 提出的標竿題庫 C (簡稱 Set C) 以及 Belenguer *et al.* [13] 提出的標竿題庫 B (簡稱 Set B)；SDVRP-MDA 選擇 Gulczynski *et al.* [29] 提出的標竿題庫 G (簡稱 Set G) 以及 Set B。由於 Set B 是 SDVRP 以及 SDVRP-MDA 的共用題庫，因此實際上共有三套，以下說明三套題庫的出處以及特性。

Set C 為 SDVRP 的標竿題庫，由 Chen *et al.* [17] 提出，共有 21 題例題。各例題中，車容量皆設定為 100，顧客點數為 8-288 個節點。所有例題的顧客位置皆以同心圓對稱的方式產生，顧客的需求量為 60 或是 90。例題的名稱部分，本研究以「"SD""題號"_"顧客點數"」表示，例如 SD1_8 表示題庫 Set C 中的第 1 題例題 8 個顧客點；SD14_120 表示第 14 題例題 120 個顧客點。

Set B 為 SDVRP 與 SDVRP-MDA 的共用題庫，由 Belenguer *et al.* [13] 所提出，車容量 Q 均為 160。原題庫為 14 題例題，Gulczynski *et al.* [29] 測試時使用其中 11 題，為方便與之比較，本研究選擇該 11 題做為 SDVRP 與 SDVRP-MDA 的標竿題庫。Set B 的例題，顧客位置參考自 TSPLIB 標竿題庫中的 eil51、eil76 以及 eil101 等三題，為均勻隨機散佈。顧客需求部分重新利用區間 $[aQ, bQ]$ 進行隨機產生，(a, b) 組合共有六種，其區間代號以及區間範圍為 D1：(0.01, 0.1)、D2：(0.1, 0.3)、D3：(0.1, 0.5)、D4：(0.1, 0.9)、D5：(0.3, 0.7) 以及 D6：(0.7, 0.9)。例題的名稱部分，本研究以「"S""節點數"_"區間代號"」表示，例如 S76_D3 表示題庫 Set B 中的例題，含 75 個顧客點與 1 個場站，需求區間為 D3：(0.1, 0.5)。

Set G 題庫為 SDVRP-MDA 的標竿題庫，由 Gulczynski *et al.* [29] 提出，共有 21 題。該題庫顧客的座標位置以及車容量參考 Set C，同樣為同心圓對稱的型態，車容量為 100。顧客需求部分，如 3.2.2 節所述，Gulczynski *et al.* [29] 對應不同的最小配送比例 p 值調整為 $\lceil 50/(1-p) \rceil$ 以及 $150 - \lceil 50/(1-p) \rceil$。例題的名稱部分，本研究以「"MD""題號"_"顧客點數"」表示，例如 MD9_48 表示題庫 Set G 中的第 9 題例題 48 個顧客點。

綜合上述題庫說明，SDVRP 的標竿題庫包括 Set C 以及 Set B，共有 33 題例題。SDVRP-MDA 的部分則包括 Set G 以及 Set B，共 33 個例題。然而，Gulczynski *et al.* [29] 在 SDVRP-MDA 的研究中，各例題另需對應求解四個不同的最小分割比例 p 值，$p = 0.1$, 0.2, 0.3 以及 0.4。因此 SDVRP-MDA 的標竿題庫有 $33 \times 4 = 128$ 題標竿例題。因此本研究 SDVRP 與 SDVRP-MDA 測試共 160 題標竿例題。

上述 Set C 與 Set G 的標竿例題可分別利用 Chen *et al.* [17] 以及 Gulczynski *et al.* [29] 研究文獻附錄中所提供的虛擬程式碼產生。Set C 與 Set G 兩組題庫間對應的題目的顧客數以及座標位置相同，僅在顧客需求的設定上有不同。本研究將標竿題庫 C 與 G 對應的例題資訊進行彙整 (e.q. 例題 SD1_8 與例題 MD1_8 彙整於同一個檔案)，並存放於以下網址：https://goo.gl/FCeYT6。Set B 標竿例題的部分則可利用 Belenguer *et al.* [13] 提供的網址獲得：http://www.uv.es/belengue/SDVRP/sdvrplib.tar.gz。

在 SDVRP-LND 的標竿題庫部分，由於問題型態為首次提出，目前無標竿題庫可以直接應用，故本研究擬利用現行分割配送相關的標竿題庫進行求解測試。現行標竿題庫中 Set C 以及 Set G 的顧客位置分布情況具有對稱特性，且顧客需求的設計有其特殊性(各例題皆只有兩種不同的顧客需求量)。由於其例題特性與實務情況較不相符的緣故，故未採用。標竿題庫 Set B 部分，顧客點數為 50-100 點，點位呈現均勻隨機散佈，需求量則是依據車容量的大小隨機產生。在例題特性上與實際情況較為符合，故本研究採用 Set B 作為 SDVRP-LND 求解測試的標竿題庫。

5.2 測試環境與參數設定

本研究提出之 SRC+IMP 演算法以 C#語言進行程式撰寫，於 Microsoft Visual Studio 2010 中編譯，測試環境為 Win 7 作業系統，CPU 為 i7-3770, 3.40GHz，RAM 為 16GB 的個人電腦。

SRC+IMP 架構可根據 IMP 使用的模組不同分為 SRC+VND 以及 SRC+RVND 兩種執行方式。在 SRC+VND 方面需要決定 VND 鄰域搜尋的順序以及設定；SRC+RVND 則更為容易，鄰域搜尋的順序以隨機的方式決定，因此僅需設定$\Delta\alpha$。本節首先以 SRC+VND 對鄰域搜尋次序以及$\Delta\alpha$進行求解測試，並以求解的績效對 VND 次序以及$\Delta\alpha$提出建議。在此前導測試中，本研究以 SDVRP-MDA 的 128 題標竿例題進行。

首先固定$\Delta\alpha = 0.01$ 進行不同鄰域排序的求解績效測試。由於鄰域搜尋模組中共使用 5 種鄰域搜尋法，亦即在排序上共有 5! = 120 種不同的順序。在測試不同鄰域排序的結果發現以 (2-*opt*, *λ-interchanges*, *Or-opt*, 2-*opt*, *SNEC*) 的排序結果最佳，意即 (N_1, N_4, N_2, N_3, N_5)，與目前的已知最佳解比較平均誤差為-0.18%。因此本研究將以 (N_1, N_4, N_2, N_3, N_5) 作為 SRC+VND 中鄰域搜尋執行的順序。

在 VND 順序確定之後，本研究進行$\Delta\alpha$的測試。本實驗中共測試了 6 種不同的$\Delta\alpha$值，包括$\Delta\alpha$= 0.0400, 0.0200, 0.0133, 0.0100, 0.0080 以及 0.0067，測試的結果如圖 5.1 所示。理論上，當$\Delta\alpha$值設定的越小，產生的起始解個數越多，求解績效越佳，求解時間則越長。從圖 5.1 中可以觀察得知，整體 SRC+VND 在搭配不同的$\Delta\alpha$值時，求解的績效與理論上相同。並且在$\Delta\alpha$= 0.0133 之後，整體求解績效趨於平緩，介於-0.17%到-0.20%之間，僅有 3‰的差距。其中以$\Delta\alpha$= 0.0080 的平均求解績效-0.20%最好，平均求解時間為 23.14 秒。

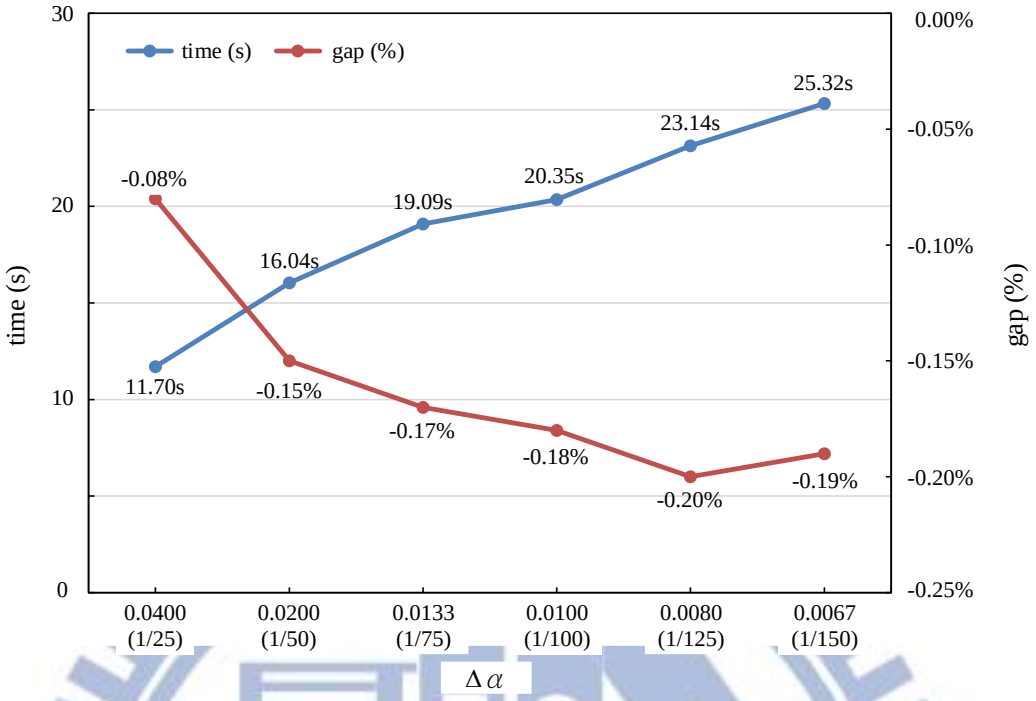

圖 5.1　Δα參數測試結果

根據上述參數測結果，本研究在後續演算法績效測試將以 2-*opt*, λ-*interchanges*, *Or-opt*, 2-*opt*, *SNEC* 作為 VND 的鄰域順序以及$\Delta\alpha = 0.0080$ 作為 SRC+VND 的測試參數。另外，利用窮舉的方式將 120 種不同的 VND 排列組合進行測試，並輸出最佳的求解結果，如此雖然運算的時間冗長但亦是一種演算法的求解架構。本研究將此種求解策略稱之為 $\text{SRC+VND}_{\text{all}}$，並於後續標竿題庫求解測試時提供 $\text{SRC+VND}_{\text{all}}$ 的求解結果以供參考。在 SRC+RVND 測試的部分，鄰域變動的順序由於執行的方式具有隨機性的關係，測試時各例題將以執行求解 20 次做為樣本數，並同時提供最佳與平均的相關結果以供參考。

第六章、SRC+IMP 於 SDVRP 測試結果與績效分析

本研究利用 Set C 與 Set B 兩個標竿題庫測試本研究所提出之演算法架構對於 SDVRP 的求解績效。表 6.1 與表 6.3 分別為 SDVRP 標竿題庫 Set C 以及 Set B 的各例題求解結果比較。本研究將求解的結果與目前績效最好的若干啟發式方法進行比較，包含 Gulczynski *et al.* [29] 的 EMIP-MDA+ERTR、Aleman and Hill [3] 的 TSVBA、Berbotto *et al.* [14] 的 RGTS 以及 Silva *et al.* [44] 的 SplitILS-LF 與 SplitILS-UF 等共五種演算法。

6.1 SDVRP 標竿題庫 C 求解績效分析

表 6.1 為 SDVRP 標竿題庫 Set C 的測試結果比較表，表中第 1 欄為例題名稱；第 2 欄為該例題目前已知最佳解 BKS；第 3-4 欄分別為 EMIP-MDA+ERTR 的求解結果 (cost) 以及運算時間 (time (s))；第 5-6 欄分別為 TSVBA 的求解結果以及運算時間；第 7 與第 8 欄分別為 RGTS 的最佳求解結果以及運算時間；第 9-10 欄分別為 SplitILS-LF 的最佳求解結果以及運算時間；第 11-12 欄分別為 SplitILS-UF 的最佳求解結果以及運算時間；第 13-14 欄分別為本研究 SRC+VND 的求解結果以及運算時間；以及第 14-15 欄分別為 SRC+RVND 的最佳求解結果以及運算時間。粗體之成本值為各題目前已知最佳解 BKS。$SRC+VND_{all}$ 的求解結果在本題庫並無求得改善 BKS，為求簡潔，因此 $SRC+VND_{all}$ 的各例題求解結果未於表 6.1 中列出。

本研究將表 6.1 中各演算法的求解績效進行彙整於表 6.2 呈現。表 6.2 中第 1 欄為演算法名稱；第 2-4 欄與演算法「最佳績效」相關，分別為 Set B 題庫的最佳結果誤差 (gap (%))、求得 BKS 的題數 (BKS found) 以及求得最佳結果的總執行時間 (total time (s))；第 5-6 欄與演算法「平均績效」相關，僅針對具隨機性的演算法的求解結果進行平均統計。第 5-6 欄分別為平均結果的誤差 (gap (%)) 以及平均時間 (time (s))。若演算法為確定性求解方法，求解結果是單一樣本，因此無平均值統計結果 (如 EMIP-MDA+ERTR、SRC+VND 與 $SRC+VND_{all}$)。由表 6.2 觀察，本研究提出的 SRC+VND 與 SRC+RVND 和 BKS 的最佳誤差均為 0.13%。在求解精準度上僅略劣於 Silva *et al.* [44] 的 SplitILS-LF (0.05%) 與 SplitILS-UF (0.04%)，但優於其他所有的文獻方法。此外，SRC+VND 在單次平均求解時間為 24.13 秒，SRC+RVND 為 26.05 秒，兩方法在求解效率上皆十分快速。

在考慮求解效果與效率兼具的情況下，SRC+VND 與 SRC+RVND 皆是非常具有競爭力的演算法。此外，在 Set C 的題庫測試結果中可以發現，SRC+VND、SRC+RVND 以及 $SRC+VND_{all}$ 與 BKS 的誤差均為 0.13%，三者間無明顯差異。推測為 Set C 題庫的顧客呈現對稱分布且顧客需求僅有 60 以及 90 兩型態，使得交換改善的深度搜尋並無法發揮太大的作用所導致。

表 6.1　SDVRP 標竿題庫 Set C (21 題) 測試結果

Set C		EMIP-MDA+ERTR		TSVBA[1]		RGTS[2]		SplitILS-LF[3]		SplitILS-UF[3]		SRC+VND		SRC+RVND	
instances	BKS	cost	time[a] (s)	cost	time[b] (s)	cost[c]	time[c, d] (s)	cost[e]	time[e,f] (s)	cost[e]	time[e,f] (s)	cost	time[g] (s)	cost[e]	time[e,g] (s)
SD1_8	**228.28**♦	**228.28**	3.10	**228.28**	0.08	**228.28**	0	**228.28**	0.05	**228.28**	0.05	**228.28**	0.02	**228.28**	0.05
SD2_16	**708.28**♦	**708.28**	424.10	**708.28**	0.40	**708.28**	0	**708.28**	0.58	**708.28**	0.63	**708.28**	0.06	**708.28**	0.09
SD3_16	**430.58**♦	**430.58**	20.80	**430.58**	0.47	**430.58**	0	**430.58**	0.59	**430.58**	0.62	**430.58**	0.06	**430.58**	0.08
SD4_24	**631.05**♦	**631.05**	454.70	**631.05**	1.87	633.98	0	**631.05**	2.16	**631.05**	2.26	**631.05**	0.11	**631.05**	0.13
SD5_32	**1390.57**♦	**1390.57**	655.50	**1390.57**	1.02	1401.28	3	**1390.57**	5.90	**1390.57**	6.07	**1390.57**	0.34	**1390.57**	0.36
SD6_32	**831.24**♦	**831.24**	656.00	**831.24**	3.72	846.16	2	**831.24**	5.62	**831.24**	5.81	**831.24**	0.23	**831.24**	0.27
SD7_40	**3640.00**♦	**3640.00**	669.90	**3640.00**	0.80	**3640.00**	3	**3640.00**	13.74	**3640.00**	14.12	**3640.00**	0.53	**3640.00**	0.65
SD8_48	**5068.28**♦	5092.36	678.30	**5068.28**	1.44	**5068.28**	2	**5068.28**	24.07	**5068.28**	24.93	**5068.28**	0.90	**5068.28**	0.98
SD9_48	**2044.20**♦	**2044.20**	662.30	2071.03	10.71	2044.73	1	**2044.20**	35.86	**2044.20**	38.78	2055.54	0.75	2055.54	0.85
SD10_64	**2684.88**♦	2704.69	700.90	2747.83	10.82	2701.55	6	**2684.88**	81.76	**2684.88**	101.10	2693.19	1.98	2693.19	2.24
SD11_80	**13280.00**♦	13358.30	709.00	**13280.00**	4.70	**13280.00**	15	**13280.00**	136.43	**13280.00**	152.42	**13280.00**	3.32	**13280.00**	3.82
SD12_80[h]	**7213.61**	7256.77	729.00	**7213.62**	18.26	**7213.62**	19	**7213.61**	179.19	**7213.61**	210.71	**7213.62**	4.07	**7213.62**	4.48
SD13_96	**10110.58**	10141.80	729.20	**10110.58**	8.87	10129.52	61	**10110.58**	168.07	**10110.58**	189.45	**10110.58**	6.72	**10110.58**	7.80
SD14_120	**10715.53**	10780.00	1718.10	10802.87	79.11	10783.00	41	**10715.53**	432.26	**10717.53**	479.85	10770.14	14.55	10767.00	15.37
SD15_144	**15093.85**	15216.30	1664.60	15153.45	50.01	15151.06	110	**15093.85**	658.54	15094.48	731.98	15138.63	24.99	15138.63	26.22
SD16_144[h]	**3381.26**	3382.16	1654.80	3446.43	237.64	3481.21	54	3395.11	580.27	**3381.26**	930.72	**3381.29**	16.58	**3381.28**	18.43
SD17_160[h]	**26493.56**	26640.70	1785.30	**26493.56**	87.53	26512.51	130	**26493.56**	484.43	26496.06	577.29	**26493.59**	28.97	**26493.59**	31.15
SD18_160	**14197.80**	14357.80	1723.00	14323.04	213.32	14293.49	61	**14197.80**	676.77	14202.53	834.60	14259.88	38.67	14270.35	40.05
SD19_192	**19989.95**	20348.20	1787.60	20157.10	190.03	20131.94	310	**19989.95**	1261.95	19995.69	1524.67	20070.07	58.73	20066.21	64.94
SD20_240	**39635.51**	39902.80	1839.80	39722.86	387.67	39701.96	560	39641.91	1518.12	**39635.51**	1563.38	39690.01	120.73	39685.87	126.64
SD21_288	**11271.06**[i]	11436.70	1825.10	11458.76	2532.80	11365.16	371	11344.96	4326.99	11345.68	5034.56	11278.96	184.28	11271.20	202.46
Average	9001.91	9072.51	1004.34	9043.31	182.92	9035.55	83.29	9006.39	504.45	9006.20	591.62	9017.32	24.13	9016.92	26.05
Avg. gap (%)[k]	–	0.47%		0.50%		0.49%		0.05%		0.04%		0.13%		0.13%	

粗體字表示目前已知最佳解

[1] Aleman and Hill [4]; [2] Berbotto *et al*. [14]; [3] Silva *et al*. [44]

♦表示該結果已由 Archetti *et al*. [5] 證明為最佳精確解

[a] P4, 3.0 GHz; [b] P4, 2.8 GHz; [c] 4 次執行的最佳; [d] 4 GB, 2.1 GHz; [e] 20 次執行的最佳; [f] i7, 2.93 GHz; [g] i7, 3.9 GHz.

[h] 進位的誤差造成 BKS 成本與本研究之結果不同

[i] 最佳解來源為 Derigs *et al*. [22]

[k] 各例題成本與 BKS 誤差之平均值

表 6.2　SDVRP 標竿題庫 Set C (21 題) 演算法求解績效彙整表

Algorithm	Best run			Average	
	gap (%)	BKS found	total time (s)	gap (%)	time (s)
EMIP-MDA+ERTR [28][a]	0.47%	8	1004.34	–	–
TSVBA [4][b]	0.50%	12	182.92	–	–
RGTS [14][c, f]	0.49%	7	324.76	0.63%	81.19
SplitILS-LF [44][d, g]	0.05%	18	10089.00	0.07%	504.45
SplitILS-UF [44][d, g]	0.04%	16	11832.40	0.08%	591.62
SRC+VND[e]	0.13%	13	24.13	–	–
SRC+RVND[e, g]	0.13%	13	521.00	0.13%	26.05
SRC+VND_{all}[e]	0.13%	13	2806.93	–	–

[a] P4, 3.0 GHz; [b] P4, 2.8 GHz; [c] 4 GB, 2.1 GHz; [d] i7, 2.93 GHz; [e] i7, 3.9 GHz
[f] 4 次執行; [g] 20 次執行

6.2　SDVRP 標竿題庫 B 求解績效分析

在 SDVRP 標竿題庫 Set B 的求解中，本研究採用比較的演算法與 Set C 相同，分別是 EMIP-MDA+ERTR [28]、TSVBA [4]、RGTS [14]、SplitILS-LF [44]以及 SplitILS-UF [44] 等 5 種演算法。表 6.3 為 SDVRP 標竿題庫 Set B 各例題的求解結果比較表，此表中第 1-16 欄的欄位分配與表 6.1 相同。由於 SRC+VND_{all} 在本題庫可求得優於 SRC+VND 以及 SRC+RVND 的求解結果，因此在表 6.3 中額外再增加 17 與 18 欄列出 SRC+VND_{all} 的求解結果以及求解時間以供參考。

表 6.4 為表 6.3 的各演算法求解績效彙整表，其中的欄位配置與表 6.2 相同。由表 6.4 可以發現，本題庫求解績效最好的演算法為 Silva *et al.* [44] 所提出的 SPLITILS-LF 以及 SPLITILS-UF，求解績效分別為 0.05%以及 0.02%。本研究所提出的 SRC+VND 以及 SRC+RVND 則僅次於上述演算法，求解績效分別為 1.05%以及 0.81%。就最佳結果的求解時間方面，SRC+VND 為最快速，平均一題只需要 12.94 秒便可完成求解。SRC+RVND 雖然平均需要花費 275.88 秒，但便能夠求得低於 1.00%以下的誤差精準度。SRC+VND_{all} 則在標竿題庫 Set B 中求解的最佳誤差為 0.73%，在求解精準度上明顯優於 SRC+VND 以及 SRC+RVND 的結果。但由於運算了 120 種 VND 的排序組合，所以求解時間加長至 1585.32 秒。

在 SDVRP 標竿題庫 Set B 的求解中，本研究提出之演算法的求解績效與標竿文獻相比亦有優勢。雖然無法優於目前所有其他演算法的求解結果，但在求解效率上則有優異的表現。就 SRC+VND 與 SRC+RVND 而言，平均一次的運算時間約在 13 秒左右，與 BKS 的誤差便能達到 1.00%的精準度。在綜合考慮求解效果與效率的情況下，本研究所提出之 SRC+VND 或是 SRC+RVND 依然具有其競爭力。

表 6.3　SDVRP 標竿題庫 Set B (11 題) 測試結果

Set B instances	BKS	EMIP-MDA +ERTR		TSVBA[1]		RGTS[2]		SplitILS-LF[3]		SplitILS-UF[3]		SRC+VND		SRC+RVND		SRC+VND$_{all}$	
		cost	time[a] (s)	cost	time[b] (s)	cost[c]	time[d, e] (s)	cost[f]	time[g,h] (s)	cost[f]	time[g,h] (s)	cost	time[i] (s)	cost[f]	time[g,i] (s)	cost	time[i] (s)
S51D2_50	**708.42**[j♦]	717.35	60.8	718.69	31.66	723.97	1	**708.42**	9.98	709.29	11.20	711.16	2.59	710.36	2.65	709.66	310.11
S51D3_50	**947.97**[j]	969.18	56.6	969.78	18.75	970.67	4	948.01	14.15	948.06	15.74	956.89	2.96	951.09	3.09	951.09	367.04
S51D4_50	**1560.88**[j]	1580.79	662.6	1628.20	19.77	1614.10	14	1561.01	59.96	1562.01	56.28	1577.56	3.98	1567.66	4.17	1571.81	497.24
S51D5_50	**1333.67**[j]	1356.37	660.0	1362.19	15.39	1381.68	3	**1333.67**	32.41	**1333.67**	36.69	1339.19	3.76	1341.04	3.93	1336.49	471.83
S51D6_50	**2169.10**	2186.29	677.8	2236.16	14.38	2213.93	2	**2169.10**	83.79	**2169.10**	62.55	2190.72	5.30	2186.41	5.37	2187.27	646.73
S76D2_75	**1087.40**	1105.19	31.4	1128.15	60.44	1113.43	10	1087.99	74.51	**1087.40**	69.36	1103.37	10.09	1099.84	10.07	1098.33	1169.77
S76D3_75	**1427.81**	1442.61	164.7	1472.92	51.13	1459.96	15	**1427.81**	88.72	1427.86	96.50	1444.46	11.79	1442.61	13.15	1433.61	1481.28
S76D4_75	**2079.76**	2104.87	980.0	2180.13	53.56	2103.05	14	**2079.76**	173.55	**2079.76**	188.38	2091.92	15.94	2091.10	17.03	2091.09	1945.91
S101D2_100	**1378.43**	1397.38	402.0	1409.03	219.52	1415.92	21	1383.35	129.94	**1378.43**	151.66	1396.11	30.51	1394.46	32.98	1395.23	3751.89
S101D3_100	**1874.81**	1921.67	961.1	1947.62	132.19	1907.92	19	1876.97	277.62	**1874.81**	317.29	1915.25	27.57	1901.27	30.72	1904.23	3462.55
S101D5_100	**2791.22**	2852.01	709.8	2910.71	131.16	2896.00	14	2792.01	696.64	**2791.22**	572.13	2821.41	27.86	2824.24	28.57	2821.41	3334.18
Average	1578.13	1603.06	487.89	1633.05	68.00	1618.24	10.64	1578.92	149.21	1578.33	143.43	1595.28	12.94	1591.38	13.79	1590.93	1585.32
Avg. gap (%)[k]	–	1.56%		3.22%		2.52%		0.05%		0.02%		1.05%		0.81%		0.73%	

粗體字表示目前已知最佳解;

[1] Aleman and Hill [4]; [2] Berbotto *et al*. [14]; [3] Silva *et al*. [44];

♦表示該結果已由 Archetti *et al*. [5] 證明為最佳精確解

[a] P4, 3.0 GHz; [b] P4, 2.8 GHz; [c] 4 次執行的最佳; [d] 4 次執行的平均; [e] 4 GB, 2.1 GHz; [f] 20 次執行的最佳; [g] 20 次執行的平均; [h] i7, 2.93 GHz; [i] i7, 3.9 GHz

[j] 最佳解來源為 Archetti *et al*. [5]; [k] 各例題成本與 BKS 誤差之平均值

表 6.4　SDVRP 標竿題庫 Set B (11 題) 演算法求解績效彙整表

Algorithm	Best run			Average	
	gap (%)	BKS found	total time (s)	gap (%)	time (s)
EMIP-MDA+ERTR [28][a]	1.56%	0	487.89	–	–
TSVBA [4][b]	3.22%	0	68.00	–	–
RGTS [14][c, f]	2.52%	0	42.56	2.57%	10.64
SplitILS-LF [44][d, g]	0.05%	5	2984.20	0.20%	149.21
SplitILS-UF [44][d, g]	0.02%	7	2868.60	0.21%	143.43
SRC+VND[e]	1.05%	0	12.94	–	–
SRC+RVND[e, g]	0.81%	0	275.88	1.09%	13.79
SRC+VND_{all}[e]	0.73%	0	1585.32	–	–

[a] P4, 3.0 GHz; [b] P4, 2.8 GHz; [c] 4 GB, 2.1 GHz; [d] i7, 2.93 GHz; [e] i7, 3.9 GHz; [f] 4 次執行; [g] 20 次執行

6.3　SDVRP 標竿題庫 C 與 B 綜合求解績效比較分析

本節將 SRC+IMP 求解 SDVRP 標竿題庫 C 與 B 的求解結果進行綜合分析。本研究共求解 32 題 SDVRP 標竿例題 (Set C 21 題以及 Set B 11 題)，整體求解績效與其他演算法比較結果彙整於表 6.5，表 6.5 的欄位分配與表 6.2 相同。由表 6.5 中觀察，目前在 SDVRP 的標竿題庫中，SRC+VND 與 SRC+RVND 皆求得 13 題 BKS 且誤差精準度小於 0.50%的求解績效。相較於目前其他文獻所提之方法，不論在求解的效果或效率上，SRC+VND 與 SRC+RVND 皆為相對優良的演算法。本研究挑選 Set C 中 SD16_144 的求解結果於附錄表 A.1 呈現，其餘例題的本研究最佳求解結果則存放於網址: http://goo.gl/qBdHGP 以供參考。

表 6.5　SDVRP 標竿測試結果 (32 題) 績效彙整比較表

Algorithm	Best run			Average	
	gap (%)	BKS found	total time (s)	gap (%)	time (s)
EMIP-MDA+ERTR [28][a]	0.84%	8	826.81	–	–
TSVBA [4][b]	1.44%	12	143.42	–	–
RGTS [14][c, f]	1.19%	7	227.75	1.30%	56.94
SplitILS-LF [44][d, g]	0.05%	23	7646.73	0.11%	382.34
SplitILS-UF [44][d, g]	0.03%	23	8751.09	0.12%	437.55
SRC+VND[e]	0.45%	13	20.28	–	–
SRC+RVND[e, g]	0.36%	13	436.74	0.46%	21.84
SRC+VND_{all}[e]	0.34%	13	2387.00	–	–

[a] P4, 3.0 GHz; [b] P4, 2.8 GHz; [c] 4 GB, 2.1 GHz; [d] i7, 2.93 GHz; [e] i7, 3.9 GHz; [f] 4 次執行; [g] 20 次執行

綜合 Set C 與 Set B 兩個標竿題庫 32 題的求解績效比較，本研究的 SRC+VND 以及 SRC+RVND 皆可求得 13 題 BKS。平均求解績效方面，SRC+VND 與 SRC+RVND 與 BKS 的平均誤差分別為 0.45%以及 0.36%，僅劣於 ILS 演算法(Silva *et al.* [44])。求解效率的部分，SRC+VND 與 SRC+RVND 的平均求解時間分別為 20.28 秒以及 436.74 秒，在效率上亦十分快速。

第七章、SRC+IMP 於 SDVRP-MDA 測試結果與績效分析

本研究以 Set G 以及 Set B 作為 SRC+IMP 在 SDVRP-MDA 求解績效比較的標竿題庫。兩個題庫皆分別利用 0.1、0.2、0.3 以及 0.4 等 4 個最小配送比例 p 值進行求解。目前的文獻中僅有 Gulczynski *et al.* [29] 的研究與 SDVRP-MDA 有相關，因此本研究在 SDVRP-MDA 的求解績效測試將引用該研究的結果進行比較。此外，本研究亦利用此部分標竿例題測試單獨使用 SRC 多重起始解的求解效果。相關結果於表 7.1 至表 7.10 呈現。

7.1 SDVRP-MDA 標竿題庫 G 求解績效分析

本研究首先利用 Set G 進行 SDVRP-MDA 的求解績效測試。Gulczynski *et al.* [29] 提出的 Set G 共有 21 個例題，各例題皆對應 p = 0.1、0.2、0.3 以及 0.4 等不同的 p 值進行求解，並將測試結果分別於表 7.1 至表 7.4 呈現。表 7.1 至表 7.4 的欄位分配方式相同，第 1 欄為對應求解的 p 值；第 2 欄為例題名稱；第 3 欄為該例題的 BKS 成本值；第 4 欄為 Gulczynski *et al.* [29] 利用視覺求得的估計解成本 (estimated solutions)；第 5-6 欄為 EMIP-MDA+ERTR [29] 的求解結果以及運算時間；第 7-8 欄、9-10 欄以及 11-12 欄分別為 SRC、SRC+VND 以及 SRC+RVND 的求解結果以及運算時間。表 7.1 到表 7.4 中最下兩列為各例題結果的平均值以及各方法在該題組下與 BKS 的平均誤差。由表 7.1 到表 7.3 觀察，p = 0.1、0.2 以及 0.3 的 BKS 成本目前皆為估計解成本，EMIP-MDA+ERTR [29] 並未有優於估計解的求解結果。但在表 7.4 中 p = 0.4 的測試結果則有不同，EMIP-MDA+ERTR [29] 更新了例題 MD6_32 以及 MD16_144 的 BKS 結果，使得 p = 0.4 的例題組的 BKS 結果與前面三組不同。

將表 7.1 至表 7.4 整體 Set G 標竿題庫 84 題的求解結果彙整於表 7.5 進行呈現。表 7.5 的欄位分配中第 1 欄為演算法名稱；第 2-5 欄與演算法「最佳績效」相關，分別為 Set G 題庫的最佳結果誤差 (gap (%))、求得 BKS 的題數 (BKS found)、改善 BKS 的題數 (BKS improved) 以及求得最佳結果的總執行時間 (total time (s))；第 6-7 欄與演算法「平均績效」相關，僅針對具隨機性的演算法的求解結果進行平均統計，分別為平均結果的誤差 (gap (%)) 以及平均時間 (time (s))。同樣的，若演算法為確定性方法則無平均值統計結果。

由表 7.5 觀察，SRC+VND 與 SRC+RVND 兩演算法對 Set G 整體 84 題求解平均誤差為-0.014%，皆可求得 71 題 BKS 以及突破 1 題，平均求解時間分別為 25.82 秒以及 28.40 秒。整體的求解相較於 EMIP-MDA+ERTR 平均 1.13%的求解誤差以及 933.03 秒的求解效率，SRC+VND 與 SRC+RVND 在 Set G 標竿題庫的表現是明顯的更為優秀。此外，若單獨利用 SRC 進行求解，平均誤差即可達到 1.06%並求得 36 個 BKS，且單題平均運算時間僅需 0.83 秒。

表 7.1　SDVRP-MDA 標竿題庫 Set G 之 $p = 0.1$ (21 題) 測試結果

				EMIP-MDA +ERTR[a]		SRC		SRC+VND		SRC+RVND	
p	Set G instance	BKS	estimated solutions	cost	time[b] (s)	cost	time[c] (s)	cost	time[c] (s)	cost[d]	time[e,c] (s)
0.1	MD1_8	**228.28**	**228.28**	**228.28**	1.06	**228.28**	0.05	**228.28**	0.06	**228.28**	0.07
	MD2_16	**720.00**	**720.00**	**720.00**	291.81	**720.00**	0.03	**720.00**	0.11	**720.00**	0.08
	MD3_16	**430.58**	**430.58**	**430.58**	5.52	**430.58**	0.05	**430.58**	0.08	**430.58**	0.09
	MD4_24	**631.05**	**631.05**	**631.05**	227.62	640.01	0.06	**631.05**	0.09	**631.05**	0.09
	MD5_32	**1402.40**	**1402.40**	1402.43	644.03	**1402.40**	0.09	**1402.40**	0.25	**1402.40**	0.21
	MD6_32	**831.24**	**831.24**	**831.24**	644.95	**831.24**	0.09	**831.24**	0.23	**831.24**	0.21
	MD7_40	**3588.28**	**3588.28**	**3588.28**	645.52	**3588.28**	0.12	**3588.28**	0.41	**3588.28**	0.36
	MD8_48	**5040.00**	**5040.00**	5060.00	646.91	**5040.00**	0.14	**5040.00**	0.62	**5040.00**	0.60
	MD9_48	**2044.20**	**2044.20**	2074.12	645.16	2080.07	0.14	**2044.20**	0.72	**2044.20**	0.63
	MD10_64[f]	**2684.88**	**2684.88**	2691.69	648.84	2706.51	0.23	**2684.89**	1.40	**2684.89**	1.43
	MD11_80	**13200.00**	**13200.00**	13220.00	662.44	**13200.00**	0.33	**13200.00**	2.51	**13200.00**	2.39
	MD12_80	**7150.58**	**7150.58**	7182.93	659.81	**7150.58**	0.33	**7150.58**	2.39	**7150.58**	2.45
	MD13_96	**10042.40**	**10042.40**	10111.80	667.02	**10042.40**	0.48	**10042.40**	4.73	**10042.40**	4.53
	MD14_120	**10711.06**	**10711.06**	10845.90	1644.98	10720.04	0.70	**10711.07**	11.59	**10711.07**	11.91
	MD15_144[f]	**15004.20**	**15004.20**	15180.70	1640.47	15049.01	0.98	**15004.22**	23.40	**15004.22**	24.89
	MD16_144	**3631.30**	**3631.30**	3755.70	1625.83	3661.89	0.98	**3631.30**	18.38	**3631.30**	17.65
	MD17_160[f]	**26362.46**	**26362.46**	26628.40	1654.72	**26362.36**	1.29	**26362.36**	24.24	**26362.36**	26.56
	MD18_160[f]	**14200.90**	**14200.90**	14477.80	1639.72	14271.20	1.36	**14200.92**	44.96	**14200.92**	48.74
	MD19_192[f]	**19964.86**	**19964.86**	20432.20	1649.31	19998.45	1.92	**19964.87**	81.31	**19964.87**	83.27
	MD20_240[f]	**39484.23**	**39484.23**	40202.50	1691.84	39528.99	3.17	**39484.21**	121.82	**39484.21**	133.57
	MD21_288[f]	**11645.50**	**11645.50**	12014.60	1656.14	11691.69	4.51	**11645.47**	304.40	**11645.47**	325.74
	Average	8999.92	8999.92	9129.06	933.03	9016.38	0.81	8999.92	30.65	8999.92	32.64
	Avg. gap (%)[g]	–	0.00%	0.93%		0.30%		0.00%		0.00%	

粗體表示現行最佳解; [a] Gulczynski *et al*. [29]; [b] P4, 3.0 GHz; [c] i7, 3.9 GHz; [d] 20 次執行的最佳; [e] 20 次執行的平均

[f] 進位的誤差造成 BKS 成本與本研究之結果不同; [g] 各例題成本與 BKS 誤差之平均值

表 7.2　SDVRP-MDA 標竿題庫 Set G 之 $p = 0.2$ (21 題) 測試結果

				EMIP-MDA +ERTR[a]		SRC		SRC+VND		SRC+RVND	
p	Set G instance	BKS	estimated solutions	cost	time[b] (s)	cost	time[c] (s)	cost	time[c] (s)	cost[d]	time[e,c] (s)
0.2	MD1_8	**228.28**	**228.28**	**228.28**	1.06	**228.28**	0.06	**228.28**	0.08	**228.2843**	0.06
	MD2_16	**720.00**	**720.00**	**720.00**	291.81	**720.00**	0.06	**720.00**	0.09	**720.00**	0.08
	MD3_16	**430.58**	**430.58**	**430.58**	5.52	**430.58**	0.05	**430.58**	0.09	**430.58**	0.08
	MD4_24	**631.05**	**631.05**	**631.05**	227.62	640.01	0.05	**631.05**	0.11	**631.05**	0.09
	MD5_32	**1402.40**	**1402.40**	**1402.40**	644.03	**1402.40**	0.12	**1402.40**	0.27	**1402.40**	0.22
	MD6_32	**831.24**	**831.24**	**831.24**	644.95	**831.24**	0.09	**831.24**	0.27	**831.24**	0.21
	MD7_40	**3588.28**	**3588.28**	**3588.28**	645.52	**3588.28**	0.12	**3588.28**	0.42	**3588.28**	0.37
	MD8_48	**5040.00**	**5040.00**	5060.00	646.91	**5040.00**	0.14	**5040.00**	0.70	**5040.00**	0.60
	MD9_48	**2044.20**	**2044.20**	2063.50	645.16	2080.07	0.17	**2044.20**	0.67	**2044.203**	0.61
	MD10_64[f]	**2684.88**	**2684.88**	2704.89	648.84	2706.51	0.23	**2684.89**	1.33	**2684.89**	1.18
	MD11_80	**13200.00**	**13200.00**	13280.00	662.44	**13200.00**	0.36	**13200.00**	2.46	**13200.00**	2.27
	MD12_80	**7150.58**	**7150.58**	7182.93	659.81	**7150.58**	0.36	**7150.58**	2.67	**7150.58**	2.62
	MD13_96	**10042.40**	**10042.40**	10130.60	667.02	**10042.40**	0.51	**10042.40**	4.29	**10042.40**	4.06
	MD14_120	**10711.06**	**10711.06**	10733.10	1644.98	10720.04	0.76	**10711.07**	11.25	**10711.07**	11.57
	MD15_144[f]	**15004.20**	**15004.20**	15116.40	1640.47	15049.01	1.05	**15004.22**	21.68	**15004.22**	23.73
	MD16_144	**3631.30**	**3631.30**	3865.24	1625.83	3661.89	1.03	**3631.30**	15.30	**3631.31**	16.58
	MD17_160[f]	**26362.46**	**26362.46**	26519.50	1654.72	**26362.36**	1.28	**26362.36**	23.67	**26362.36**	23.11
	MD18_160[f]	**14200.90**	**14200.90**	14559.20	1639.72	14271.20	1.29	**14200.92**	39.62	**14200.92**	38.53
	MD19_192[f]	**19964.86**	**19964.86**	20300.40	1649.31	19998.45	1.98	**19964.87**	72.59	**19964.87**	79.69
	MD20_240[f]	**39484.23**	**39484.23**	40102.30	1691.84	39528.99	3.20	**39484.21**	119.15	**39484.21**	126.01
	MD21_288[f]	**11645.50**	**11645.50**	12438.60	1656.14	11691.69	4.60	**11645.47**	235.95	**11645.47**	285.74
	Average	8999.92	8999.92	9137.55	933.03	9016.38	0.83	8999.92	26.32	8999.92	29.40
	Avg. gap (%)[g]	–	0.00%	1.17%		0.30%		0.00%		0.00%	

粗體表示現行最佳解; [a] Gulczynski *et al*. [29]; [b] P4, 3.0 GHz; [c] i7, 3.9 GHz; [d] 20 次執行的最佳; [e] 20 次執行的平均

[f] 進位的誤差造成 BKS 成本與本研究之結果不同; [g] 各例題成本與 BKS 誤差之平均值

表 7.3　SDVRP-MDA 標竿題庫 Set G 之 $p = 0.3$ (21 題) 測試結果

				EMIP-MDA +ERTR[a]		SRC		SRC+VND		SRC+RVND	
p	Set G instance	BKS	estimated solutions	cost	time[b] (s)	cost	time[c] (s)	cost	time[c] (s)	cost[d]	time[e,c] (s)
0.3	MD1_8	**228.28**	**228.28**	**228.28**	1.06	**228.28**	0.03	**228.28**	0.08	**228.28**	0.06
	MD2_16	**720.00**	**720.00**	**720.00**	291.81	**720.00**	0.05	**720.00**	0.12	**720.00**	0.08
	MD3_16	**430.58**	**430.58**	**430.58**	5.52	**430.58**	0.06	**430.58**	0.09	**430.58**	0.08
	MD4_24	**631.05**	**631.05**	**631.05**	227.62	640.01	0.08	**631.05**	0.12	**631.05**	0.09
	MD5_32	**1402.40**	**1402.40**	**1402.40**	644.03	**1402.40**	0.08	**1402.40**	0.20	**1402.40**	0.20
	MD6_32	**831.24**	**831.24**	**831.24**	644.95	**831.24**	0.11	**831.24**	0.28	**831.24**	0.21
	MD7_40	**3588.28**	**3588.28**	**3588.28**	645.52	**3588.28**	0.12	**3588.28**	0.42	**3588.28**	0.36
	MD8_48	**5040.00**	**5040.00**	**5040.00**	646.91	**5040.00**	0.14	**5040.00**	0.69	**5040.00**	0.58
	MD9_48	**2044.20**	**2044.20**	2063.50	645.16	2080.07	0.16	**2044.20**	0.75	**2044.20**	0.69
	MD10_64[f]	**2684.88**	**2684.88**	2710.64	648.84	2706.51	0.23	**2684.89**	1.22	**2684.88**	1.15
	MD11_80	**13200.00**	**13200.00**	13334.10	662.44	**13200.00**	0.34	**13200.00**	2.34	**13200.00**	2.19
	MD12_80	**7150.58**	**7150.58**	7170.58	659.81	**7150.58**	0.34	**7150.58**	2.26	**7150.58**	2.39
	MD13_96	**10042.40**	**10042.40**	10112.40	667.02	**10042.40**	0.48	**10042.40**	5.30	**10042.40**	5.50
	MD14_120	**10711.06**	**10711.06**	10836.30	1644.98	10720.04	0.76	**10711.07**	9.22	**10711.07**	9.34
	MD15_144[f]	**15004.20**	**15004.20**	15172.10	1640.47	15078.66	1.08	**15004.22**	23.18	**15004.22**	22.59
	MD16_144	**3631.30**	**3631.30**	3962.67	1625.83	3661.89	1.05	**3631.30**	13.23	**3631.30**	15.34
	MD17_160[f]	**26362.46**	**26362.46**	26646.50	1654.72	**26362.36**	1.33	**26362.36**	21.58	**26362.36**	22.50
	MD18_160[f]	**14200.90**	**14200.90**	14420.20	1639.72	14271.20	1.26	**14200.92**	31.42	**14200.92**	31.18
	MD19_192[f]	**19964.86**	**19964.86**	20355.70	1649.31	20018.45	2.00	**19964.87**	60.59	**19964.87**	61.95
	MD20_240[f]	**39484.23**	**39484.23**	40018.30	1691.84	39558.64	3.24	**39484.21**	120.76	**39484.21**	125.38
	MD21_288[f]	**11645.50**	**11645.50**	12652.90	1656.14	11691.69	4.54	**11645.47**	234.36	**11645.47**	262.91
	Average	8999.92	8999.92	9158.46	933.03	9020.16	0.83	8999.92	25.15	8999.92	26.89
	Avg. gap (%)[g]	–	0.00%	1.42%		0.32%		0.00%		0.00%	

粗體表示現行最佳解; [a] Gulczynski *et al.* [29]; [b] P4, 3.0 GHz; [c] i7, 3.9 GHz; [d] 20 次執行的最佳; [e] 20 次執行的平均
[f] 進位的誤差造成 BKS 成本與本研究之結果不同; [g] 各例題成本與 BKS 誤差之平均值

表 7.4　SDVRP-MDA 標竿題庫 Set G 之 $p = 0.4$ (21 題) 測試結果

				EMIP-MDA +ERTR[a]		SRC		SRC+VND		SRC+RVND	
p	Set G instance	BKS	estimated solutions	cost	time[b] (s)	cost	time[c] (s)	cost	time[c] (s)	cost[d]	time[e,c] (s)
0.4	MD1_8	**228.28**	**228.28**	**228.28**	1.06	**228.28**	0.05	**228.28**	0.05	**228.28**	0.06
	MD2_16	**720.00**	**720.00**	**720.00**	291.81	753.14	0.03	**720.00**	0.06	**720.00**	0.06
	MD3_16	**430.58**	**430.58**	**430.58**	5.52	**430.58**	0.03	**430.58**	0.08	**430.58**	0.07
	MD4_24	**631.05**	**631.05**	**631.05**	227.62	640.01	0.08	**631.05**	0.09	**631.05**	0.08
	MD5_32	**1402.40**	**1402.40**	1414.75	644.03	**1402.40**	0.09	**1402.40**	0.25	**1402.40**	0.23
	MD6_32	**830.26**	831.24	**830.26**	644.95	831.24	0.06	831.24	0.20	831.24	0.19
	MD7_40	**3588.28**	**3588.28**	**3588.28**	645.52	3810.26	0.08	**3588.28**	0.47	**3588.28**	0.43
	MD8_48	**5040.00**	**5040.00**	**5040.00**	646.91	5363.84	0.17	**5040.00**	0.80	**5040.00**	0.72
	MD9_48	**2044.20**	**2044.20**	2059.03	645.16	2080.07	0.14	**2044.20**	0.51	**2044.20**	0.50
	MD10_64[f]	**2684.88**	**2684.88**	2708.80	648.84	2684.89	0.20	**2684.88**	1.15	**2684.88**	1.19
	MD11_80	**13200.00**	**13200.00**	13240.00	662.44	13796.98	0.31	**13200.00**	3.01	**13200.00**	2.85
	MD12_80	**7150.58**	**7150.58**	7260.01	659.81	7695.17	0.36	**7150.58**	3.10	**7150.58**	3.28
	MD13_96	**10042.40**	**10042.40**	10233.50	667.02	10592.22	0.53	**10042.40**	5.69	**10042.40**	5.97
	MD14_120	**10711.06**	**10711.06**	10865.10	1644.98	11057.13	0.81	**10711.07**	9.06	**10711.07**	10.22
	MD15_144[f]	**15004.20**	**15004.20**	15202.80	1640.47	15796.02	1.15	**15004.22**	18.83	**15004.22**	20.95
	MD16_144	**3445.50**	3631.30	**3445.50**	1625.83	3661.89	1.06	**3443.69**	12.15	**3443.69**	14.11
	MD17_160[f]	**26362.46**	**26362.46**	26904.70	1654.72	27456.64	1.28	**26362.35**	27.77	**26362.36**	29.11
	MD18_160[f]	**14200.90**	**14200.90**	14447.60	1639.72	14724.31	1.28	**14200.92**	24.82	**14200.92**	27.79
	MD19_192[f]	**19964.86**	**19964.86**	20608.90	1649.31	20826.25	2.03	20032.12	52.14	20009.71	57.26
	MD20_240[f]	**39484.23**	**39484.23**	40551.40	1691.84	41220.15	3.32	**39484.21**	102.38	**39484.21**	119.05
	MD21_288[f]	11645.50	11645.50	11909.10	1656.14	11701.90	4.88	**11457.88**	181.80	11476.57	223.53
	Average	8991.03	8999.92	9158.08	933.03	9369.21	0.86	8985.25	21.16	8985.08	24.65
	Avg. gap (%)[g]		0.26%	1.00%		3.33%		-0.06%		-0.06%	

粗體表示現行最佳解，底線表示改善 BKS; [a] Gulczynski *et al.* [29]; [b] P4, 3.0 GHz; [c] i7, 3.9 GHz; [d] 20 次執行的最佳
[e] 20 次執行的平均; [f] 進位的誤差造成 BKS 成本與本研究之結果不同; [g] 各例題成本與 BKS 誤差之平均值

表 7.5　SDVRP-MDA 標竿題庫 Set G (84 題) 測試結果彙整表

Algorithm	Best run				Average	
	gap (%)	BKS found	BKS improved	total time (s)	gap (%)	time (s)
EMIP-MDA+ERTR [29][a]	1.13%	28	–	933.03	–	–
SRC[b]	1.06%	36	0	0.83	–	–
SRC+VND[b]	-0.014%	81	1	25.82	–	–
SRC+RVND[b,c]	-0.014%	81	1	567.93	-0.014%	28.40
SRC+VND$_{all}$[b]	-0.014%	81	1	3139.58	–	–

[a] P4, 3.0 GHz; [b] i7, 3.9 GHz; [c] 20 次執行

特別一提，本研究在求解 Set G 時發現有若干例題的成本與 BKS 的成本十分接近，經過與 Gulczynski [28] 的博士論文中公布的實際路線結果比對，確定為同一個路線解，誤差產生的是由於數字進位的關係導致。因此在表 7.1 至表 7.4 中各表中第一欄的例題上方標註「進位的誤差造成 BKS 成本與本研究之結果不同」以茲說明。

7.2　SDVRP-MDA 標竿題庫 B 求解績效分析

表 7.6 至表 7.9 列出本研究對 SDVRP-MDA 的 Set B 標竿題庫對應求解 p = 0.1、0.2、0.3 以及 0.4 的結果。Set B 題庫共有 11 題例題，搭配 4 個不同的 p 值，共產生 44 個不同的例題結果。針對這 44 題例題，目前的 BKS 皆為 Gulczynski *et al.* [29] 所求得之結果。

由於 Set B 各個例題對應的 p 值不同時，顧客點的需求並不改變，因此不同 p 值僅影響各顧客 i 的 MDA$_i$ 值。換言之，當 p 值越大，最小配送量的要求越高，問題求解的限制相對越嚴格。因此當路線解結果為較嚴格的 p 值時的可行解時，該解亦為同例題搭配較小 p 值的可行解。本研究以上述性質調整 Set B 各例題對應不同 p 值的 BKS，如 p = 0.1 時 S76D3_75、S76D4_75、S101D3_100 以及 S101D5_100 等 4 題例題的 BKS 即更新為同例題 p = 0.2 的 BKS。

在表 7.6 至表 7.9 中的欄位分配如下，第 1 欄為對應求解的 p 值；第 2 欄為例題名稱；第 3 欄為例題在對應 p 值的 BKS 結果；其餘欄位為比較 4 種演算法的求解結果，包括 EMIP-MDA+ERTR (第 4 欄)、SRC+VND (第 5-6 欄)、SRC+RVND (第 7-8 欄) 以及 SRC+VND$_{all}$ (第 9-10 欄)。此外，由於本研究在求解 p = 0.3 時，發現 S51D5_50 的 SRC+VND$_{all}$ 求解結果以及 S101D2_100 的 SRC+RVND 求解結果優於在 p = 0.2 的求解結果。因此，在表 7.7 中額外增加第 11 欄 (Best solution adjusted) 將實際上該例題目前最佳的 BKS 列出以供參考。表 7.6 至表 7.9 各 p 值求解結果列出之後，亦在末 2 列增加平均值與平均誤差之統計數值以供參考。

表 7.6　SDVRP-MDA 標竿題庫 Set B 之 $p = 0.1$ (11 題) 測試結果

p	Set B Instance	BKS	EMIP-MDA +ERTR[a] cost	SRC+VND cost	SRC+VND time[b] (s)	SRC+RVND cost[c]	SRC+RVND time[d,b] (s)	SRC+VND$_{all}$ cost	SRC+VND$_{all}$ time[b] (s)
0.1	S51D2_50	717.35	717.35	713.30	3.99	711.38	3.42	**709.77**	439.65
	S51D3_50	969.99	969.99	953.17	4.84	951.09	4.63	**949.78**	561.03
	S51D4_50	1588.91	1588.91	**1569.89**	6.68	1579.69	6.63	**1569.89**	775.82
	S51D5_50	1373.98	1373.98	1344.68	5.49	1346.49	5.63	**1335.55**	666.33
	S51D6_50	2225.51	2225.51	2197.64	7.96	**2197.04**	7.83	**2197.04**	916.53
	S76D2_75	1106.86	1106.86	1100.90	15.85	1107.87	14.27	**1095.53**	1782.50
	S76D3_75	1446.48[e]	1457.40	1448.39	20.78	1444.88	19.46	**1440.56**	2362.41
	S76D4_75	2118.86[e]	2123.16	**2095.95**	25.96	2097.47	25.37	**2095.95**	3040.63
	S101D2_100	1398.13	1398.13	1407.35	48.20	1400.45	46.90	**1395.42**	5600.83
	S101D3_100	1929.96[e]	1930.86	1900.60	46.29	1904.84	45.67	**1899.66**	5404.45
	S101D5_100	2862.14[e]	2862.34	2851.33	53.01	2841.10	54.48	**2839.49**	6325.75
	Average	1612.56	1614.05	1598.47	21.73	1598.39	21.30	1593.51	2534.18
	Avg. gap (%)[f]		0.09%	-0.87%		-0.87%		-1.23%	

粗體表示現行最佳解，底線表示改善 BKS

[a] 未提供運算時間; [b] i7-3770, 3.4 GHz; [c] 20 次執行的最佳; [d] 20 次執行的平均

[e] 為 EMIP-MDA+ERTR 求解 $p = 0.2$ 的結果

[f] 各例題成本與 BKS 誤差之平均值

表 7.7　SDVRP-MDA 標竿題庫 Set B 之 $p = 0.2$ (11 題) 測試結果

p	Set B Instance	BKS	EMIP-MDA +ERTR[a] cost	SRC+VND cost	SRC+VND time[b] (s)	SRC+RVND cost[c]	SRC+RVND time[d,b] (s)	SRC+VND$_{all}$ cost	SRC+VND$_{all}$ time[b] (s)	Best solution adjusted
0.2	S51D2_50	717.35	717.35	712.59	3.79	711.68	3.25	**709.77**	424.37	
	S51D3_50	969.99	969.99	959.49	4.26	957.70	4.12	**955.45**	511.86	
	S51D4_50	1593.69	1593.69	1589.32	6.19	1581.59	5.84	**1578.15**	709.12	
	S51D5_50	1377.99	1377.99	1359.22	4.66	1358.96	4.55	1351.12	553.29	**1349.55**[e,f]
	S51D6_50	2285.37	2285.37	**2240.14**	5.07	2240.64	5.19	**2240.14**	619.05	
	S76D2_75	1116.64	1116.64	1116.07	13.53	1106.86	12.80	**1104.27**	1621.75	
	S76D3_75	1446.48	1446.48	1450.04	17.07	1449.89	17.10	**1445.63**	2095.96	
	S76D4_75	2118.86	2118.86	2112.62	22.89	2104.00	23.30	**2103.78**	2837.47	
	S101D2_100	1398.87	1398.87	1401.57	44.32	1397.00	46.15	1396.78	5544.72	**1396.17**[f]
	S101D3_100	1929.96	1929.96	1917.56	43.31	1910.09	44.63	**1904.09**	5319.77	
	S101D5_100	2862.14	2862.14	2866.29	42.96	**2861.88**	44.85	**2861.88**	5379.26	
	Average	1619.76	1619.76	1611.36	18.91	1607.30	19.25	1604.64	2328.78	
	Avg. gap (%)[g]		0.00%	-0.52%		-0.79%		-0.99%		

粗體表示現行最佳解，底線表示改善 BKS

[a] 未提供運算時間; [b] i7-3770, 3.4 GHz; [c] 20 次執行的最佳; [d] 20 次執行的平均

[e] 為 SRC+VND$_{all}$ 求解 $p = 0.3$ 的結果; [f] 為 SRC+RVND 求解 $p = 0.3$ 的結果

[g] 各例題成本與 BKS 誤差之平均值

表 7.8　SDVRP-MDA 標竿題庫 Set B 之 $p = 0.3$ (11 題) 測試結果

p	Set B Instance	BKS	EMIP-MDA +ERTR[a] cost	SRC+VND cost	SRC+VND time[b] (s)	SRC+RVND cost[c]	SRC+RVND time[d,b] (s)	SRC+VND$_{all}$ cost	SRC+VND$_{all}$ time[b] (s)
0.3	S51D2_50	717.35	717.35	716.17	3.09	709.63	2.86	**711.56**	370.07
	S51D3_50	969.99	969.99	966.68	4.32	**959.16**	4.07	**959.16**	508.91
	S51D4_50	1597.89	1597.89	1597.10	4.43	1597.50	4.20	**1595.40**	528.17
	S51D5_50	1383.71	1383.71	1356.64	4.02	**1349.55**	3.98	**1349.55**	488.59
	S51D6_50	**2301.51**	**2301.51**	2307.14	2.84	2307.14	2.85	2307.14	357.24
	S76D2_75	1116.64	1116.64	1118.41	14.20	1113.04	13.54	**1108.20**	1676.14
	S76D3_75	1453.17[e]	1453.25	**1446.81**	15.23	1453.11	15.55	**1446.81**	1912.85
	S76D4_75	2151.49	2151.49	**2123.06**	17.88	2127.00	18.49	**2123.06**	2255.93
	S101D2_100	1398.88[e]	1401.85	1407.19	40.76	**1396.17**	42.86	1396.99	5108.96
	S101D3_100	1939.96	1939.96	1921.27	37.44	1921.61	38.85	**1918.41**	4678.31
	S101D5_100	2901.76	2901.76	2887.90	38.45	2882.29	38.93	**2873.36**	4691.72
	Average	1630.21	1630.49	1622.58	16.61	1619.66	16.92	1617.24	2052.44
	Avg. gap (%)[f]		0.02%	-0.43%		-0.70%		-0.82%	

粗體表示現行最佳解，底線表示改善 BKS

[a] 未提供運算時間; [b] i7-3770, 3.4 GHz; [c] 20 次執行的最佳; [d] 20 次執行的平均

[e] 為 EMIP-MDA+ERTR 求解 $p = 0.4$ 的結果

[f] 各例題成本與 BKS 誤差之平均值

表 7.9　SDVRP-MDA 標竿題庫 Set B 之 $p = 0.4$ (11 題) 測試結果

p	Set B Instance	BKS	EMIP-MDA +ERTR[a] cost	SRC+VND cost	SRC+VND time[b] (s)	SRC+RVND cost[c]	SRC+RVND time[d,b] (s)	SRC+VND$_{all}$ cost	SRC+VND$_{all}$ time[b] (s)
0.4	S51D2_50	717.35	717.35	717.31	3.10	716.33	2.89	**713.60**	367.89
	S51D3_50	978.41	978.41	962.21	3.70	962.01	3.54	**959.16**	445.00
	S51D4_50	**1612.30**	**1612.30**	1619.87	3.40	1619.12	3.25	1613.06	405.11
	S51D5_50	1389.32	1389.32	1373.82	3.32	1360.26	3.22	**1360.01**	399.73
	S51D6_50	2402.35	2402.35	**2395.33**	1.37	**2395.33**	1.45	**2395.33**	174.85
	S76D2_75	1116.64	1116.64	1113.78	13.51	**1112.58**	13.43	**1112.58**	1656.92
	S76D3_75	1453.17	1453.17	1453.43	14.21	1453.20	13.96	**1446.88**	1712.80
	S76D4_75	2167.27	2167.27	2143.90	16.50	2139.97	16.52	**2129.97**	2035.99
	S101D2_100	1398.88	1398.88	1403.70	39.42	1399.64	40.46	**1397.75**	4862.50
	S101D3_100	1942.94	1942.94	1935.08	31.09	**1922.10**	32.21	**1922.10**	3938.04
	S101D5_100	2904.61	2904.61	2898.98	32.15	**2879.33**	31.39	2890.76	3950.67
	Average	1643.93	1643.93	1637.95	14.71	1633.35	14.76	1631.02	1813.59
	Avg. gap (%)[e]		0.00%	-0.38%		-0.66%		-0.82%	

粗體表示現行最佳解，底線表示改善 BKS

[a] 未提供運算時間; [b] i7-3770, 3.4 GHz; [c] 20 次執行的最佳; [d] 20 次執行的平均;

[e] 各例題成本與 BKS 誤差之平均值

表 7.10 為 SDVRP-MDA 標竿題庫 Set B 各演算法求解結果的彙整表，欄位配置與表 7.5 相同。由表 7.10 可以明顯地發現 SRC+VND、SRC+RVND 以及 SRC+VND_{all} 的求解績效皆優於 EMIP-MDA+ERTR。在 44 題例題中，SRC+VND 可求得 33 題改善 BKS 的結果，誤差為-0.55%；SRC+RVND 可求得 36 題，誤差為-0.76%，SRC+VND_{all} 可求得 42 題改善 BKS 的結果。其中，SRC+VND_{all} 唯二無法改善的例題為 $p=0.3$ 的 S51D6_50 以及 $p=0.4$ 的 S51D4_50。EMIP-MDA+ERTR 求解 Set B 標竿題庫的運算時間並未在其研究中提供，因此無法在求解效率上與之比較。但基於 SDVRP 的 Set C 與 Set B 標竿題庫的結果以及 SDVRP-MDA 的 Set G 標竿題庫的求解結果推導，在 SDVRP-MDA 的 Set B 中， SRC+VND 與 SRC+RVND 的求解效率應仍優於 EMIP-MDA+ERTR。

此外，不同於 EMIP-MDA+ERTR 在求解時間上不受 p 值的影響，本研究提出之演算法的求解時間上皆隨 p 值提高有下降的趨勢。具體而言，在表 7.6 中 SRC+VND 求解 $p=0.1$、0.2、0.3 以及 0.4 的運算時間分別為 21.7、18.9、16.6、14.7 秒；SRC+RVND 求解 $p=0.1$、0.2、0.3 以及 0.4 的運算時間分別為 21.3、19.3、16.9、14.8 秒。這趨勢表示對於限制越嚴格 (p 值越大) 的例題，求解的效率越快。推測是由於限制越嚴的問題，其可行解區域越小，求解的搜尋時間亦隨之減少。

表 7.10　SDVRP-MDA 標竿題庫 Set B (44 題) 測試結果彙整表

Algorithm	Best run				Average	
	gap (%)	BKS found	BKS improved	total time (s)	gap (%)	time (s)
EMIP-MDA+ERTR [29][a]	0.03%	38	–	–	–	–
SRC+VND[b]	-0.55%	0	33	17.99	–	–
SRC+RVND[b,c]	-0.76%	0	36	361.17	-0.40%	18.06
SRC+VND_{all}[b]	-0.96%	0	42	2182.25	–	–

[a] P4, 3.0 GHz; [b] i7, 3.9 GHz; [c] 20 次執行

7.3　SDVRP-MDA 標竿題庫 G 與 B 綜合求解績效比較分析

本研究針對 SDVRP-MDA 標竿題庫 Set G 與 Set B 配合最小配送比例 $p=0.1$、0.2、0.3 以及 0.4 進行求解測試，共測試 128 題標竿例題。將這 128 題 SDVRP-MDA 標竿例題的求解結果進行彙整統計並於表 7.11 呈現。

表 7.11　SDVRP-MDA 標竿測試結果 (128 題) 績效彙整比較表

Algorithm	Best run				Average	
	gap (%)	BKS found	BKS improved	total time (s)	gap (%)	time (s)
EMIP-MDA+ERTR [29][a]	0.75%	66	–	–	–	–
SRC+VND[b]	-0.20%	81	34	23.13	–	–
SRC+RVND[b,c]	-0.27%	81	37	496.86	-0.15%	24.85
SRC+VND_{all}[b]	-0.34%	81	43	2810.50	–	–

[a] P4, 3.0 GHz; [b] i7, 3.9 GHz; [c] 20 次執行

表 7.11 中欄位的配置與表 7.5 以及表 7.10 相同。由表 7.11 可以觀察出，本研究提

出之 SRC+IMP 架構在求解精準度上有優越之表現。平均誤差方面，以多重起點 SRC 的架構不論是搭配固定鄰域變動順序的 VND (-0.20%)、隨機 RVND (-0.27%) 或是 VND_{all} (-0.34%)，皆能夠求得優於 BKS 的求解績效。其中以 SRC+VND_{all} 的求解結果最佳，並能夠在 128 題標竿例題中求得 81 題以及改善 43 題 BKS。僅在 $p = 0.3$ 的 S51D6_50 與 $p = 0.4$ 的 S51D4_50、MD6_32 以及 MD19_192 等四題例題無法求得小於等於 BKS 的結果，但這四題例題在誤差方面亦皆小於 0.30%。

本研究挑選 Set B 中 S76D4_75 ($p = 0.2$) 的求解結果於附錄表 B.1 呈現，該結果突破了目前已知最佳解。在這個例題結果中，被分送次數最多的節點為顧客 11 以及 58，皆為 3 次。此外，路線 1 為服務最多分送顧客的路線，該路線與其他路線共同服務了四個顧客 (36、37、47 以及 48)。其餘例題的本研究最佳求解結果則存放於網址 http://goo.gl/qBdHGP 以供參考。

求解的效率方面，由於 Gulczynski *et al.* [29] 在 EMIP-MDA+ERTR 求解 Set B 的結果中並沒有提供時間，因此無法與之進行比較。SRC+VND 與 SRC+RVND 兩種策略的單題平均求解時間十分接近，分別是 23.13 秒以及 24.85 秒。SRC+VND_{all} 因為時間需要求解 120 種不同的 VND 鄰域排列組合，耗費的時間最長，大約需要 2810.50 秒。

第八章、SRC+IMP 於 SDVRP-LND 測試結果與績效分析

8.1 SDVRP-LND 標竿例題之 *kmax* 參數設定與已知最佳解建立

SDVRP-LND 為本研究首次提出之問題類型，目前並無標竿題庫，亦無已知最佳解可供演算法求解績效的測試比較。故本研究選擇 SDVRP 以及 SDVRP-MDA 的共用題庫 Set B 作為 SDVRP-LND 的標竿題庫進行測試。*kmax* 值的設定部分，本研究首先將 5.2 節本研究針對 Set B 標竿題庫所求得的最佳 SDVRP 結果進行分析，探討各結果實際的最大配送次數，結果如表 8.1。由表 8.1 中可以發現，11 題例題中共有 6 題的 $kmax = 3$，其餘 5 題為 $kmax = 2$。此部分的觀察結果與物流業者認為至多配送 3 次的建議相同。因此在後續，SDVRP-LND 的測試方面，本研究在最大配送次數的設定採用 $kmax = 3$ 與 2，共測試 22 (11 × 2) 個標竿例題。

表 8.1 本研究 SDVRP 最佳結果與其對應之 *kmax* 值

Set B Instance	SDVRP	
	cost	*kmax*
S51D2_50	709.66	2
S51D3_50	951.09	3
S51D4_50	1567.66	3
S51D5_50	1336.49	3
S51D6_50	2186.41	2
S76D2_75	1098.33	2
S76D3_75	1433.61	3
S76D4_75	2091.09	3
S101D2_100	1394.46	2
S101D3_100	1901.27	2
S101D5_100	2821.41	3

8.2 SDVRP-LND 標竿題庫 B 求解績效分析

本研究利用求解 SDVRP 標竿題庫 Set B 的求解結果進行彙整以萃取出目前 SDVRP-LND 標竿題庫的 BKS 結果。萃取的方式是將各例題以 SRC+RVND 以及 $\text{SRC+VND}_{\text{all}}$ (包含 SRC+VND) 進行求解，並對所有求解的結果中選擇對應 $kmax = 2$ 以及 $kmax = 3$ 最好的結果作為 BKS。

表 8.2 以及表 8.3 為本研究提出之演算法對應 $kmax = 2$ 以及 3 求解 SDVRP-LND 的標竿題庫 Set B 的求解結果。表 8.2 以及表 8.3 中第 1 欄為最大配送次數 *kmax* 值；第 2 欄為例題名稱；第 3 欄為該例題在對應 *kmax* 值的 BKS 結果；其餘欄位為比較本研究 3 種演算法的求解結果，包括 SRC+VND (第 4-5 欄)、SRC+RVND (第 6-7 欄) 以及 $\text{SRC+VND}_{\text{all}}$ (第 8-9 欄)。加入本研究求之解結果若有新的 BKS 結果產生則備註於第 10 欄。由於 $kmax = 2$ 求解結果亦為 $kmax = 3$ 限制下的可行解，因此表 8.3 中則添加第 10

欄，以紀錄同例題 *kmax* = 2 的求解結果若優於 *kmax* = 3 的情況。例如，目前 S76D2_75、S101D2_100 以及 S101D5_100 在 *kmax* = 3 的 BKS 應分別調整為 *kmax* = 2 時 SRC+VND$_{all}$ 求得的 1098.20、1389.08 以及 2819.51。表 8.2 以及表 8.3 求解結果列出之後，亦增加平均值與平均誤差之統計數值於表 8.2 以及表 8.3 以供參考。

表 8.2　SDVRP-LND 標竿題庫 Set B 之 *kmax* = 2 (11 題)測試結果

kmax	Set B Instance	BKS	SRC+VND		SRC+RVND		SRC+VND$_{all}$	
			cost	time[a] (s)	cost[b]	time[a,c] (s)	cost	time[a] (s)
2	S51D2_50	709.66	712.97	3.74	710.59	3.49	**709.39**	445.54
	S51D3_50	**951.66**	953.06	4.34	952.33	4.14	952.33	521.31
	S51D4_50	1588.64	1578.34	6.07	1582.98	5.80	**1573.70**	726.97
	S51D5_50	1341.09	1341.71	5.71	**1340.04**	5.34	**1340.04**	673.38
	S51D6_50	**2186.41**	2230.35	6.22	2230.36	5.91	2230.35	743.23
	S76D2_75	1098.33	1105.93	14.82	**1098.20**	13.82	**1098.20**	1742.18
	S76D3_75	1446.12	1448.83	18.35	1446.50	17.78	**1441.73**	2218.21
	S76D4_75	2111.91	**2095.65**	24.32	2101.66	23.56	**2095.65**	2956.81
	S101D2_100	1394.46	1400.45	45.43	1396.60	49.02	**1389.08**	5980.96
	S101D3_100	**1901.27**	1911.51	42.15	1904.92	42.15	1904.92	5243.93
	S101D5_100	2851.68	2846.41	49.17	**2819.51**	45.96	**2819.51**	5734.79
	Average	1598.29	1602.29	20.03	1598.52	19.72	1595.90	2453.39
	Avg. gap (%)[d]		0.27%		0.05%		-0.13%	

粗體表示現行最佳解，底線表示改善 BKS

[a] i7-3770, 3.4 GHz; [b] 20 次執行的最佳; [c] 20 次執行的平均;

[d] 各例題成本與 BKS 誤差之平均值;

表 8.3　SDVRP-LND 標竿題庫 Set B 之 *kmax* = 3 (11 題)測試結果

kmax	Set B Instance	BKS	SRC+VND		SRC+RVND		SRC+VND$_{all}$		Best solution adjusted
			cost	time[a] (s)	cost[b]	time[a,c] (s)	cost	time[a] (s)	
3	S51D2_50	709.66[d]	714.60	3.81	709.66	3.58	**709.39**	462.76	
	S51D3_50	**951.09**	953.00	4.46	**951.09**	4.49	**951.09**	557.91	
	S51D4_50	**1567.66**	1574.00	6.82	**1567.66**	6.42	1572.94	813.02	
	S51D5_50	**1336.49**	1339.68	6.35	1339.68	6.02	1339.68	757.66	
	S51D6_50	**2186.41**[d]	2194.78	9.05	2193.61	8.40	2193.61	1053.74	
	S76D2_75	1098.33[d]	1098.33	15.26	1099.84	14.56	1098.33	1855.19	**1098.20**[e]
	S76D3_75	**1433.61**	1439.68	20.03	**1433.61**	19.35	**1433.61**	2410.34	
	S76D4_75	**2091.09**	2091.92	28.14	2091.97	27.26	**2091.09**	3432.71	
	S101D2_100	1394.46[d]	1401.89	46.33	1394.45	50.90	1394.45	6168.80	**1389.08**[e]
	S101D3_100	**1901.27**[d]	1909.40	47.60	1904.85	49.02	1904.23	6009.09	
	S101D5_100	2821.41	2821.41	56.71	2826.84	54.70	2821.41	6822.31	**2819.51**[e]
	Average	1590.13	1594.43	22.23	1592.11	22.24	1591.80	2758.50	
	Avg. gap (%)[f]		0.30%		0.10%		0.09%		

粗體表示現行最佳解，底線表示改善 BKS

[a] i7-3770, 3.4 GHz; [b] 20 次執行的最佳; [c] 20 次執行的平均;

[d] 為 *kmax* = 2 的 BKS; [e] 為 *kmax* = 2 的 SRC+VND$_{all}$ 的結果; [f] 各例題成本與 BKS 誤差之平均值;

表 8.4 為表 8.2 與表 8.3 求解結果的彙整，表 8.4 欄位配置與先前表 7.5 相同。由表 8.4 可以觀察到，SRC+VND、SRC+RVND 以及 SRC+VND$_{all}$ 等三個演算法的最佳求解結果與 BKS 的誤差皆十分接近於 0.00%。其中，又以 SRC+RVND 以及 SRC+VND$_{all}$ 的最佳求解績效較佳，分別求得到-0.07%以及-0.02%，並於 22 題例題中可改善 5 題以及 6 題 BKS。求解效率方面，確定性求解的 SRC+VND 求得最佳結果的總時間為 21.13 秒；SRC+RVND 因為測試 20 次的關係，最佳結果的產生耗費 419.69 秒，平均一次測試的時間則接近 SRC+VND，為 20.98 秒；SRC+VND$_{all}$ 的求解結果最佳，但運算的時間需要 2605.95 秒。

由 SDVRP-LND 標竿題庫的求解績效觀察，不論是 SRC+VND 或是 SRC+RVND 的解題架構皆能夠求得相當接近 BKS 的結果。求解績效的良好亦反應出本研究方法一般化求解的能力以及彈性。由於目前用於比較績效的 BKS 結果是由 SDVRP 以及 SDVRP-MDA 的所有求解過程中產生的結果所彙整萃取而得，因此應是屬於品質相當優良的 BKS，亦較難突破。SRC+VND 以及 SRC+RVND 分別能在 22 題標竿題庫中求得突破 3 題以及 5 題的結果，實屬表現相當優異。

本研究挑選 Set B 中 S76D2_75 (*kmax* = 2) 的最佳結果於附錄表 C.1 進行呈現，S76D2_75 (*kmax* = 2) 的最佳結果除了改善已知最佳解外，其亦優於 S76D2_75 (*kmax* = 3) 的 BKS，因此該結果亦為 *kmax* = 2 以及 *kmax* = 3 的目前最佳結果。SDVRP-LND 的其餘例題的最佳路線結果則存放於網址 http://goo.gl/qBdHGP 以供參考。

表 8.4　SDVRP-LND 標竿題庫 Set B (22 題) 測試結果彙整表

Algorithm	Best run				Average	
	gap (%)	BKS found	BKS improved	total time (s)	gap (%)	time (s)
SRC+VND[a]	0.29%	2	3	21.13	–	–
SRC+RVND[a,b]	0.07%	5	5	419.69	0.40%	20.98
SRC+VND$_{all}$[a]	-0.02%	6	9	2605.95	–	–

[a] i7, 3.9 GHz; [b] 20 次執行

第九章、綜合比較與探討

本研究針對 SDVRP、SDVRP-MDA 以及 SDVRP-LND 的求解測試於第六到八章進行比較分析。所有標竿例題的最佳測試結果可至網址 http://goo.gl/qBdHGP 進行下載。

本章節將對於本研究所提出之多重起點啟發式演算法進行綜合性的結果比較。包含 SRC 多重起始解是否有效提高求解搜尋的多樣性、*SNEC* 對於求解績效的影響、VND 以及 RVND 兩種不同的變動鄰域策略的求解績效比較以及增加配送次數限制的 SDVRP-LND 對於 VRP 以及 SDVRP 在成本上的差異進行分析與探討。

9.1 SRC 多重起點起始解多樣性說明

SRC 起始解構建模組能夠利用不同的α值以及不同的策略 (SRC_0 或 SRC_1) 產生多重起始點。SRC 構建起始解時，首先利用插入現行路線與新增路線兩個成本的權重公式挑選欲排程的顧客點。再針對該顧客以 *NIRA* 決定需求最後配置的方式。*NIRA* 是一套具有適應性的插入排程演算法。首先，先將顧客需求以最佳的方案進行安排。若發生路線超載，則挑選該路線中距離場站較近的顧客需求進行分割或是移出，以維持可行性。

SRC 在整個演算法中扮演的是一個廣度搜尋的重要角色，因此該模組產生的起始解多樣性十分重要。本研究遂利用 TSPLIB 標竿題庫中的例題 eil30 對 SRC 進行一個初步的測試。我們以$\Delta\alpha = 0.25$ 進行測試求解，因此整個 SRC 將產生 5 個 (1/0.25 + 1) 不同的α值，搭配 2 個起始解策略後共可求得 10 個起始解結果，圖 9.為這 10 個結果的路線圖。

由圖中可以觀察到，SRC_0 策略所求得的起始解結果皆為 3 條路線，而 SRC_1 策略的結果皆為 4 條路線。因此，SRC_0 的起始解成本皆低於 SRC_1。在路線結構上來觀察，不論是 SRC_0 或是 SRC_1 皆能夠產生具有分割特性的起始解。此外，受到 SRC_1 最遠點單獨路線的策略影響，使得兩種不同策略的路線解產生不同的結構。再由不同的α值的路線結果觀察，雖然是同一個起始解策略下，但不同的α值也的確造成路線結構的變化。由上述觀察得知，SRC 多重起始解模組在起始解的多樣性上確實有其效果。

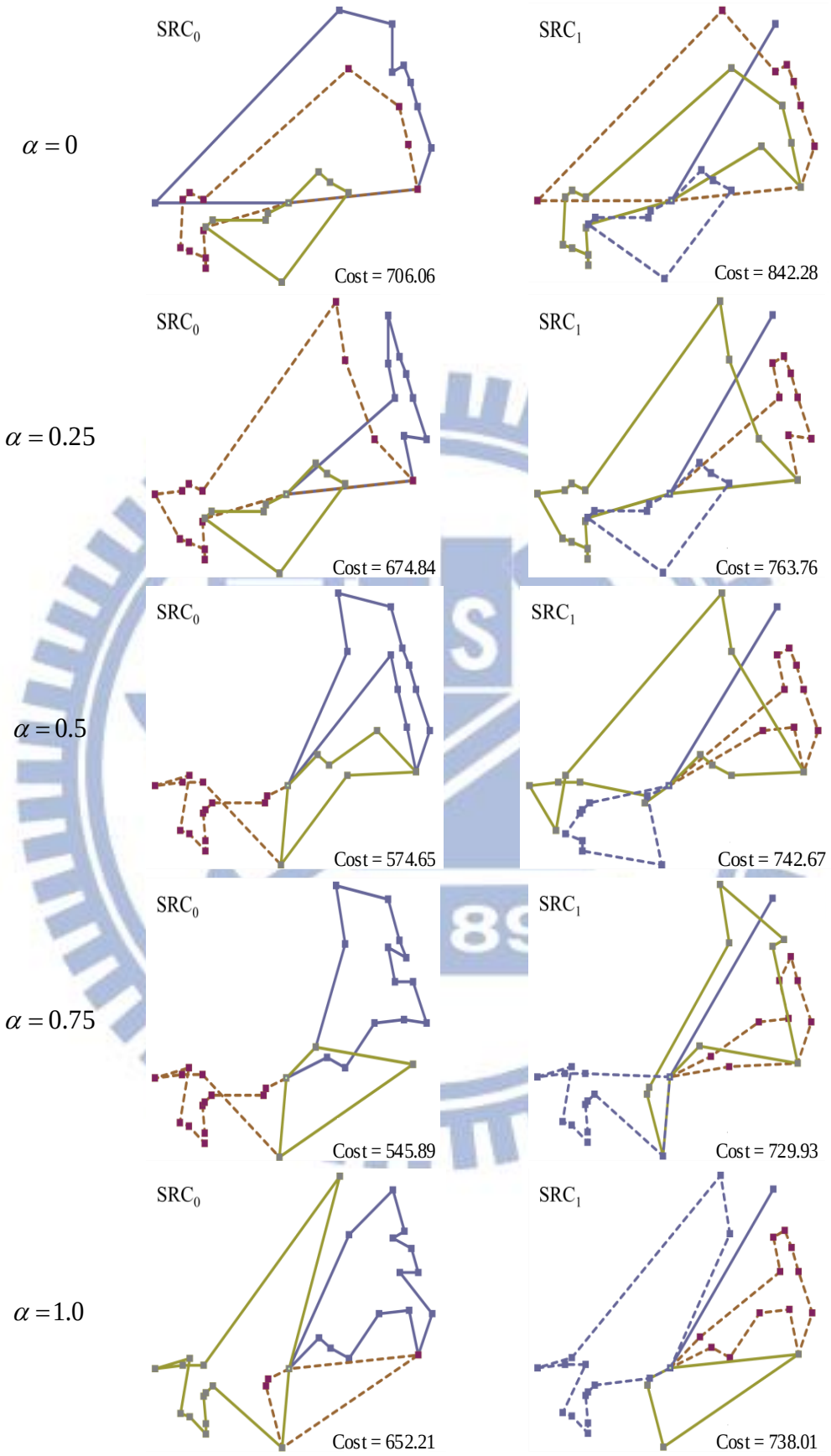

圖 9.1　SRC 多重起始解模組多樣性

9.2 有無 *SNEC* 鄰域搜尋法對求解之影響比較

本研究設計一個簡單的測試，以標竿題庫 Set B 作為求解例題，比較有無 *SNEC* 的情況下比較求解績效，求解架構為 SRC ($\Delta\alpha = 0.01$) + RVND。以 with_*SNEC* 表示 RVND 中包含 *SNEC*，以 without_*SNEC* 表示 RVND 中無 *SNEC*，並分別測試 20 次，求解結果如表 9.1。表 9.1 中第 1 欄為例題名稱；第 2 欄為該例題 SDVRP 的 BKS；第 3-5 欄為 without_*SNEC* 的求解績效，分別包含最佳結果 (best cost)、平均結果 (Avg. cost) 以及平均時間 (time (s))；第 6-8 欄為 with_*SNEC* 的求解績效，分別包含最佳結果 (best cost)、平均結果 (Avg. cost) 以及平均時間 (time (s))。表 9.1 中最下方兩列分別統計各欄位的平均值 (Average) 以及求解結果與 BKS 的平均誤差 (Avg. gap (%))。由表 9.1 可以發現 *SNEC* 對於求解效果上有明顯的幫助，最佳結果誤差為 0.84%，相較無 *SNEC* 的 2.75% 提升約 1.91%的求解精準度。平均求解效率為 11.35 秒，雖較無 *SNEC* 的 6.18 秒增加約 5 秒鐘的時間，但求解效率依舊快速。由此可以判斷 *SNEC* 是為一個有效求解分割配送車輛路線問題的鄰域搜尋方法。

表 9.1 有無 *SNEC* 的求解績效比較

		without_*SNEC*			with_*SNEC*		
instances[a]	BKS	best cost[a]	Avg. cost[b]	time[b,c] (s)	best cost[a]	Avg. cost	time[b,c] (s)
S51D2_50	**708.42**[d♦]	718.02	726.28	2.21	710.68	713.71	2.38
S51D3_50	**947.97**[d]	964.90	972.13	2.71	951.09	956.11	2.75
S51D4_50	**1560.88**[d]	1610.37	1624.69	2.36	1572.59	1574.62	3.64
S51D5_50	**1333.67**[d]	1361.84	1373.77	2.81	1337.67	1342.36	3.39
S51D6_50	**2169.10**[e]	2209.85	2226.01	2.64	2188.08	2189.48	4.33
S76D2_75	**1087.40**[e]	1122.24	1124.39	5.64	1099.84	1103.14	8.67
S76D3_75	**1427.81**[e]	1473.66	1481.23	6.51	1437.62	1447.05	11.05
S76D4_75	**2079.76**[e]	2142.65	2147.43	7.06	2091.92	2094.19	13.83
S101D2_100	**1378.43**[e]	1418.53	1426.97	13.05	1397.40	1400.03	27.48
S101D3_100	**1874.81**[e]	1941.42	1948.98	12.54	1906.71	1908.38	24.88
S101D5_100	**2791.22**[e]	2902.79	2904.43	10.41	2824.73	2826.99	22.42
Average	1578.13	1624.21	1632.39	6.18	1592.57	1596.00	11.35
Avg. gap (%)[f]	–	2.75%	3.29%		0.84%	1.11%	

粗體字表示目前已知最佳解; ♦表示該結果已由 Archetti *et al*. [5] 證明為最佳精確解

[a] 20 次執行的最佳; [b] 20 次執行的平均; [c] i7, 3.9 GHz.; [d] 最佳解來源為 Archetti *et al*. [5];

[e] 最佳解來源為 Silva *et al*. [44]; [f] 各例題成本與 BKS 誤差之平均值

9.3 SRC+VND 與 SRC+RVND 求解績效比較

在前述章節中，本研究分別利用 SRC+VND、SRC+RVND 與 SRC+VND$_{all}$ 三套演算法分別測試 SDVRP (32 題)、SDVRP-MDA (128 題)以及 SDVRP-LND (22 題) 三個問題的標竿題庫，總共測試 182 題標竿例題，並將前述所有測試結果進行綜合彙整表如表 9.2 所示。

表 9.2　不同變動鄰域策略對 SDVRPs 求解績效彙整表

Algorithm	Best run				Average	
	gap (%)	BKS found	BKS improved	total time (s)	gap (%)	time (s)
SRC+VND[a]	-0.03%	96	37	22.39	–	–
SRC+RVND[a,b]	-0.12%	99	42	476.96	0.02%	23.85
SRC+VND$_{all}$[a]	-0.18%	100	52	2711.31	–	–

[a] i7, 3.9 GHz; [b] 20 次執行

基本上，演算法的好壞應由其求解效果以及效率進行判斷。由表 9.2 觀察，在求解精準度的表現上是 SRC+VND$_{all}$ ≻ SRC+RVND ≻ SRC+VND，而求解效率方面排序則相反是 SRC+VND ≻ SRC+RVND ≻ SRC+VND$_{all}$。如此的表現是符合演算的邏輯，求解的所花費的時間越長，能夠進行搜尋的機會就越多，求得更好的結果的機會亦相對提高。圖 9.2 為 SRC+VND、SRC+RVND 與 SRC+VND$_{all}$ 三套演算法最佳結果的求解績效的折線分析圖。從圖中可以觀察到，以 SRC+VND 的結果為基礎來看，SRC+RVND 因為 20 次執行的關係，大約是多耗費 20 倍的運算時間，績效上可以提升將近 0.1%的改善幅度；SRC+VND$_{all}$ 則因為需要運算 120 總不同的 VND 排序的關係，大約是多耗費 120 倍的運算時間，可以較 SRC+VND 提升約 0.15%的改善幅度。由以上數據可以說明，SRC+RVND 是一套兼顧求解效果與效率的演算法。此外，本研究最好的 VND 排序其實需要透過部分題庫經過 VND$_{all}$ 的測試結果，以測試題組「平均」績效最佳者進行推薦，但若以單一例題而言，則無法保證是最好的鄰域排序方式。綜合上述觀察，在實際應用上，本研究建議的演算法架構為 SRC+RVND，使用者可以依照求解時間允許的程度，決定 SRC+RVND 的求解次數。一般而言，具有隨機性的演算法增加求解樣本數對於搜尋的廣度上具有相當的幫助。

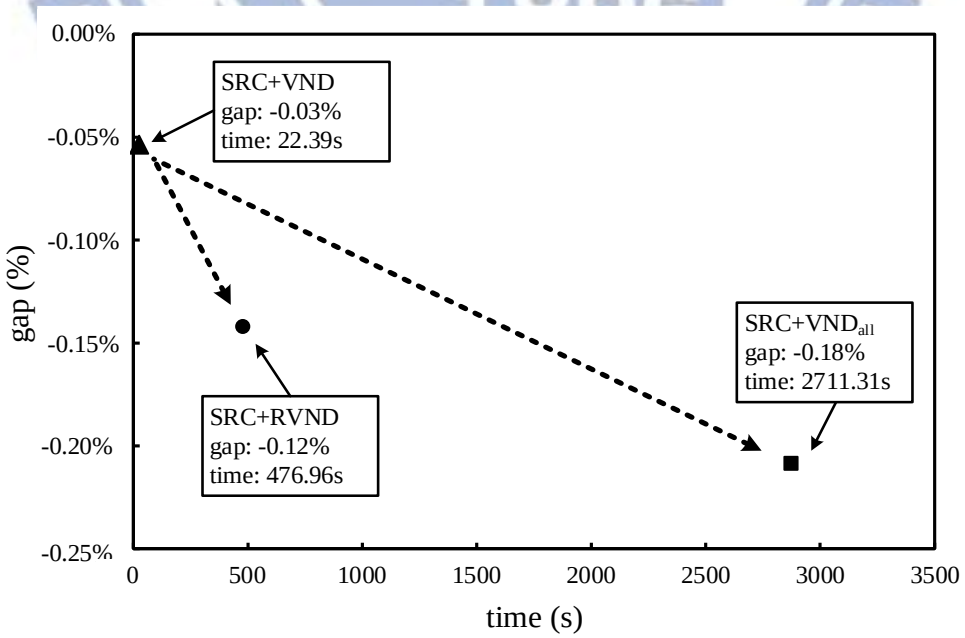

圖 9.2　本研究整體求解績效分析圖

9.4 有無次數限制之分割配送車輛路線問題潛在效益之評估

本節將討論傳統 VRP 問題、具配送次數限制的 SDVRP-LND 問題以及 SDVRP 問題間成本上的比較。成本的比較上，本研究以共通標竿題庫 Set B 作為分析的目標題庫。分別針對 Set B 中各例題萃取出本研究演算法所求得的 VRP (以 SRC+VND、SRC+RVND 以及 $SRC+VND_{all}$ 求解 $kmax=1$ 的 SDVRP-LND) 、SDVRP-LND 以及 SDVRP 的最佳成本，並計算與 VRP 結果的誤差後彙整於表 9.3。表 9.3 中，第 1 欄為例題名稱；第 2 欄為 VRP 最佳結果；第 3-4 以及 5-6 欄則分別為 SDVRP-LND 對應 $kmax = 2$ 以及 3 的最佳結果 (cost) 以及節省幅度 (saving)。第 7-8 欄則為本研究求解 SDVRP 的最佳結果以及節省幅度。表 9.3 最下面一列列出各欄位數值的平均值。節省幅度算法是透過以下公式計算 (以 SDVRP-LND 舉例)：

$$\frac{z(\text{VRP})-z(\text{SDVRP}-\text{LND})}{z(\text{VRP})}\times 100\% \tag{16}$$

表 9.3 中，由左至右，隨著問題的分割限制越來越放鬆，最佳結果的平均成本有遞減的趨勢。由節省幅度的部分觀察，當問題放鬆到 $kmax=2$ 的 SDVRP-LND 時，相較於 $kmax = 1$ 的 VRP 約在成本可節省 3.85%，$kmax = 3$ 時節省為 4.11%，到最限制最鬆的 SDVRP 則可節省 4.17%。

表 9.3 有無配送次數限制對成本節省之影響分析表

Set B	VRP ($kmax = 1$)	SDVRP-LND ($kmax = 2$)		SDVRP-LND ($kmax = 3$)		SDVRP ($kmax = \infty$)	
Instance	cost	cost	saving (%)	cost	saving (%)	cost	saving (%)
S51D2_50	717.19	709.39	1.09%	709.39	1.09%	709.39	1.09%
S51D3_50	973.28	952.33	2.15%	951.09	2.28%	951.09	2.28%
S51D4_50	1676.42	1573.70	6.13%	1567.66	6.49%	1567.66	6.49%
S51D5_50	1435.64	1340.04	6.66%	1339.68	6.68%	1336.49	6.91%
S51D6_50	2402.35	2230.35	7.16%	2193.61	8.69%	2186.41	8.99%
S76D2_75	1111.71	1098.20	1.22%	1098.2	1.22%	1098.20	1.22%
S76D3_75	1462.81	1441.73	1.44%	1433.61	2.00%	1433.61	2.00%
S76D4_75	2186.32	2095.65	4.15%	2091.09	4.36%	2091.09	4.36%
S101D2_100	1407.88	1389.08	1.34%	1389.08	1.34%	1389.08	1.34%
S101D3_100	1948.58	1904.92	2.24%	1904.23	2.28%	1901.27	2.43%
S101D5_100	3091.92	2819.51	8.81%	2819.51	8.81%	2819.51	8.81%
Average	1674.01	1595.90	3.85%	1590.65	4.11%	1589.44	4.17%

經過實際觀察可以發現，雖然 SDVRP-LND 對於配送的最大次數進行限制，但可節省的幅度 ($kmax = 2$ 的 3.85%以及 $kmax = 3$ 的 4.11%) 僅略低於完全不受限制的 SDVRP (4.17%)。其中，當 SDVRP-LND 的 $kmax = 3$ 時，11 題例題中僅有 3 題的求解成本劣於不受限制的 SDVRP，其餘 8 題皆可以達到 SDVRP 的求解結果，與 SDVRP 的節省程度僅差距 0.06%。此點說明 SDVRP-LND 在成本上的節省依然顯著，配送次數的限制亦符合實務應用的期待，故 SDVRP-LND 非常適合作為後續車輛路線問題的研究或是實務應用。

第十章、結論與建議

分割配送車輛路線問題 SDVRP 於 1989 年提出至今已有將近約三十年的時間，其特點在於允許同一個需求可由多部車輛共同進行服務。SDVRP 的潛在效益方面，對於整體物流成本中的距離成本或是車輛使用數上至多可以節省 50%。但在實務應用上，允許分割配送的物流方式會多次收送貨的情況而造成其他作業上的成本。有鑑於此，如何使 SDVRP 兼顧成本效率與服務品質是一重要議題。近年已有文獻 (Gulczynski *et al.* [29]) 將最小配送量列入考慮，提出最小配送量限制之分割配送車輛路線問題 (SDVRP with Minimum Delivery Amounts, SDVRR-MDA)。本研究則首次提出以配送次數作為限制的「具配送次數限制的分割配送車輛路線問題」(SDVRP with Limited Number of Deliveries, SDVRP-LND)，以期分割配送能有更高的應用價值。

本研究以 SDVRP、SDVRP-MDA 以及 SDVRP-LND 為主題，在考慮方法求解效果與效率兼顧的目標下，提出多重起點搭配鄰域搜尋的 SRC+IMP 演算法架構。SRC+IMP 的求解設計考慮分割配送相關問題的特性，是一套能夠一體適用於不同分割配送車輛路線問的的求解方法。其中，起始解產生模組 SRC 可透過α參數的變動產生多個不同的分割配送起始解作為搜尋起點；鄰域改善模組 IMP 中則新提出一套針對問題特性所設計的 *SNEC* 鄰域搜尋法，並搭配其他傳統搜尋法，以 VND 或是 RVND 的變動鄰域策略進行搜尋改善。本研究利用 SDVRP、SDVRP-MDA 以及 SDVRP-LND 的標竿題庫進行求解測試並分析，主要貢獻如下:

1. 提出衍伸問題 SDVRP-LND 以配合實務應用需求，並對該問題的數學規劃模式、整數解性質以及與 VRP 相對產生的節省部分進行初步研究。

2. 演算法設計方面具有以下優點：(1) 單一參數設計，僅需要設定$\Delta\alpha$便可執行求解。(2) 依分割需求的問題特性設計，能夠有效加大可行鄰域搜尋的範圍。(3) 以多重起點加強廣度搜尋，並利用 *SNEC* 適應性的交換方式提高深度搜尋能力。(4) 僅以下降的改善方式進行簡單求解，提高演算法被應用的可行性。

3. 求解績效方面，本研究 SRC+IMP 的求解架構依據變動鄰域順率的不同可分為 SRC+VND 以及 SRC+RVND 兩種執行方式。在測試 SDVRP、SDVRP-MDA 以及 SDVRP-LND 三套的標竿題庫共 182 題標竿例題的結果發現 SRC+RVND 求解績效優於 SRC+VND。其結果在 SDVRP 方面，對 32 個標竿題求解的平均誤差為 0.36%，其績效僅次於 ILS (Silva *et al.*[44])；在 SDVRP-MDA 方面，對 128 個標竿題求解的平均誤差為-0.27%，且求得 81 題並突破 36 題已知最佳解；在 SDVRP-LND 方面，22 題標竿例題中則求得 5 題已知最佳解並突破 5 題，平均求解誤差為 0.07%。

4. 彙整 SRC 搭配 VND、RVND 的求解結果進行分析。透過求解結果的績效優劣，對於變動鄰域下降策略給予建議。結果發現搭配 RVND 的變動鄰域策略雖然在運算總時間上高於 VND，但其在求解品質上的有其實際助益。

5. 分析 SDVRP- LND 對應不同 *kmax* 值時，其節省程度的變化。發現適當的對 *kmax* 值進行限制並不會造成大幅的損失。特別是當 *kmax*＝3 時，SDVRP-LND 的成本節省效益為 4.11%，其與無次數限制的 SDVRP 相差僅 0.06%；這顯示以最多配送三次執行 SDVRP-LND 可對物流界提供相當的實用價值。建議後續研究分割配送相關問題時，應將 SDVRP-LND 配送次數的限制考慮。

本研究所提出之 SRC+IMP 是一套針對分割需求車輛路線問題所設計的一般化演算法。與現行的相關演算法比較，不論在求解效果或效率上，皆十分具有競爭力。由於演算法架構簡單，因此可輕易的與其他巨集進行結合。建議後續研究可嘗試將不同的巨集策略加入求解分割配送相關車輛路線問題，應能有精確度更高的求解效果。

多重起點 SRC 以及針對問題設計的交換法 *SNEC* 是本研究演算法中的兩個主要特色。兩者分別在搜尋的廣度以及深度上加強了演算法求解的能力。建議後續其他車輛路線問題的相關研究亦能採用此種針對問題特性設計方法論的概念加入方法論的設計中，應更容易有突破性的發現。

參考文獻

1 7-Eleven. (2017). Takeout service order. from https://www.7-11.com.tw/service/takeout.asp

2 Ahuja, R., Magnanti, T. and Orlin, J. (1993). *Network Flows: Theory, Algorithms, and Applications*: Prentice Hall.

3 Aleman, R. E. and Hill, R. R. "A Tabu Search with Vocabulary Building Approach for the Vehicle Routing Problem with Split Demands." *International Journal of Metaheuristics,* Vol. 1, No. 1, pp. 55-80, 2010.

4 Aleman, R. E., Zhang, X. and Hill, R. R. "An Adaptive Memory Algorithm for the Split Delivery Vehicle Routing problem." *Journal of Heuristics,* Vol. 16, No. 3, pp. 441-473, 2010.

5 Archetti, C., Bianchessi, N. and Speranza, M. G. "Branch-and-cut Algorithms for the Split Delivery Vehicle Routing Problem." *European Journal of Operational Research,* Vol. 238, No. 3, pp. 685-698, 2014.

6 Archetti, C., Bianchessi, N. and Speranza, M. G. "A Column Generation Approach for the Split Delivery Vehicle Routing Problem." *Networks,* Vol. 58, No. 4, pp. 241-254, 2011.

7 Archetti, C., Mansini, R. and Speranza, M. G. "Complexity and Reducibility of the Skip Delivery Problem." *Transportation Science,* Vol. 39, No. 2, pp. 182-187, 2005.

8 Archetti, C., Savelsbergh, M. W. P. and Speranza, M. G. "To Split or not to Split: That is the Question." *Transportation Research Part E: Logistics and Transportation Review,* Vol. 44, No. 1, pp. 114-123, 2008.

9 Archetti, C., Savelsbergh, M. W. P. and Speranza, M. G. "Worst-Case Analysis for Split Delivery Vehicle Routing Problems." *Transportation Science,* Vol. 40, No. 2, pp. 226-234, 2006.

10 Archetti, C. and Speranza, M. G. "Vehicle Routing Problems with Split Deliveries." *International Transactions in Operational Research,* Vol. 19, No. 1-2, pp. 3-22, 2012.

11 Archetti, C., Speranza, M. G. and Hertz, A. "A Tabu Search Algorithm for the Split Delivery Vehicle Routing Problem." *Transportation Science,* Vol. 40, No. 1, pp. 64-73, 2006.

12 Archetti, C., Speranza, M. G. and Savelsbergh, M. W. P. "An Optimization-Based Heuristic for the Split Delivery Vehicle Routing Problem." *Transportation Science,* Vol. 42, No. 1, pp. 22-31, 2008.

13 Belenguer, J. M., Martinez, M. C. and Mota, E. "A Lower Bound for the Split Delivery Vehicle Routing Problem." *Operations Research,* Vol. 48, No. 5, pp. 801-810, 2000.

14 Berbotto, L., García, S. and Nogales, F. J. “A Randomized Granular Tabu Search Heuristic for the Split Delivery Vehicle Routing Problem.” *Annals of Operations Research,* Vol. 222, No. 1, pp. 153-173, 2014.

15 Boudia, M., Prins, C. and Reghioui, M. (2007). An Effective Memetic Algorithm with Population Management for the Split Delivery Vehicle Routing Problem. In T. Bartz-Beielstein, M. J. Blesa Aguilera, C. Blum, B. Naujoks, A. Roli, G. Rudolph, & M. Sampels (Eds.), *Hybrid Metaheuristics: 4th International Workshop, HM 2007, Dortmund, Germany, October 8-9, 2007. Proceedings* (pp. 16-30). Berlin, Heidelberg: Springer Berlin Heidelberg.

16 Bräysy, O., Hasle, G. and Dullaert, W. “A Multi-Start Local Search Algorithm for the Vehicle Routing Problem with Time Windows.” *European Journal of Operational Research,* Vol. 159, No. 3, pp. 586-605, 2004.

17 Chen, S., Golden, B. and Wasil, E. “The Split Delivery Vehicle Routing Problem: Applications, Algorithms, Test Problems, and Computational results.” *Networks,* Vol. 49, No. 4, pp. 318-329, 2007.

18 Clarke, G. and Wright, J. W. “Scheduling of Vehicles from a Central Depot to a Number of Delivery Points.” *Operations Research,* Vol. 12, No. 4, pp. 568-581, 1964.

19 Cordeau, J.-F. and Laporte, G. (2005). Tabu Search Heuristics for the Vehicle Routing Problem. In R. Sharda, S. Voß, C. Rego, & B. Alidaee (Eds.), *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search* (pp. 145-163). Boston, MA: Springer US.

20 Croes, G. A. “A Method for Solving Traveling-Salesman Problems.” *Operations Research,* Vol. 6, No. 6, pp. 791-812, 1958.

21 Dantzig, G. B. and Ramser, J. H. “The Truck Dispatching Problem.” *Management Science,* Vol. 6, No. 1, pp. 80-91, 1959.

22 Derigs, U., Li, B. and Vogel, U. “Local Search-based Metaheuristics for the Split Delivery Vehicle Routing problem.” *Journal of the Operational Research Society,* Vol. 61, No. 9, pp. 1356-1364, 2010.

23 Dror, M. and Trudeau, P. “Savings by Split Delivery Routing.” *Transportation Science,* Vol. 23, No. 2, pp. 141-145, 1989.

24 Dror, M. and Trudeau, P. “Split Delivery Routing.” *Naval Research Logistics,* Vol. 37, No. 3, pp. 383-402, 1990.

25 Gendreau, M., Hertz, A. and Laporte, G. “New Insertion and Postoptimization Procedures for the Traveling Salesman Problem.” *Operations Research,* Vol. 40, No. 6, pp. 1086-1094, 1992.

26 Glover, F. (1992). New Ejection Chain and Alternating Path Methods for Traveling Salesman Problems. In O. Balci, R. Sharda, & S. A. Zenios (Eds.), *Computer Science and Operations Research* (pp. 491-509). Amsterdam: Pergamon.

27 Groër, C., Golden, B. and Wasil, E. “A Parallel Algorithm for the Vehicle Routing Problem.” *INFORMS Journal on Computing,* Vol. 23, No. 2, pp. 315-330, 2011.

28 Gulczynski, D. (2010). *Integer Programming-Based Heuristics for Vehicle Routing Problems.* (Ph.D.), University of Maryland.

29 Gulczynski, D., Golden, B. and Wasil, E. “The Split Delivery Vehicle Routing Problem with Minimum Delivery Amounts.” *Transportation Research Part E: Logistics and Transportation Review,* Vol. 46, No. 5, pp. 612-626, 2010.

30 Han, A. F.-W. and Chu, Y.-C. “A Multi-start Heuristic Approach for the Split-delivery Vehicle Routing Problem with Minimum Delivery Amounts.” *Transportation Research Part E: Logistics and Transportation Review,* Vol. 88, No., pp. 11-31, 2016.

31 Ho, S. C. and Haugland, D. “A Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows and Split Deliveries.” *Computers & Operations Research,* Vol. 31, No. 12, pp. 1947-1964, 2004.

32 Jin, M., Liu, K. and Bowden, R. O. “A Two-stage Algorithm with Valid Inequalities for the Split Delivery Vehicle Routing Problem.” *International Journal of Production Economics,* Vol. 105, No. 1, pp. 228-242, 2007.

33 Jin, M., Liu, K. and Eksioglu, B. “A Column Generation Approach for the Split Delivery Vehicle Routing Problem.” *Operations Research Letters,* Vol. 36, No. 2, pp. 265-270, 2008.

34 López-Sánchez, A. D., Hernández-Díaz, A. G., Vigo, D., Caballero, R. and Molina, J. “A Multi-Start Algorithm for a Balanced Real-World Open Vehicle Routing Problem.” *European Journal of Operational Research,* Vol. 238, No. 1, pp. 104-113, 2014.

35 Lin, S. “Computer Solutions of the Traveling Salesman Problem.” *Bell System Technical Journal,* Vol. 44, No. 10, pp. 2245-2269, 1965.

36 Lin, S. and Kernighan, B. W. “An Effective Heuristic Algorithm for the Traveling-Salesman Problem.” *Operations Research,* Vol. 21, No. 2, pp. 498-516, 1973.

37 Mladenović, N. and Hansen, P. “Variable Neighborhood Search.” *Computers & Operations Research,* Vol. 24, No. 11, pp. 1097-1100, 1997.

38 Mota, E., Campos, V. and Corberán, Á. (2007). A New Metaheuristic for the Vehicle Routing Problem with Split Demands. In C. Cotta & J. van Hemert (Eds.), *Evolutionary Computation in Combinatorial Optimization: 7th European Conference, EvoCOP 2007, Valencia, Spain, April 11-13, 2007. Proceedings* (pp. 121-129). Berlin, Heidelberg: Springer Berlin Heidelberg.

39 Or, I. (1976). *Traveling Salesman-type Combinatorial Problems and Their Relation to the Logistics of Regional Blood Banking.* (Ph.D. Dissertation), Northwestern University.

40 Osman, I. H. “Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem.” *Annals of Operations Research,* Vol. 41, No. 4, pp. 421-451, 1993.

41 Potvin, J.-Y. and Rousseau, J.-M. “An Exchange Heuristic for Routeing Problems with Time Windows.” *The Journal of the Operational Research Society,* Vol. 46, No. 12, pp. 1433-1446, 1995.

42 Rego, C. “Node-ejection Chains for the Vehicle Routing Problem: Sequential and Parallel Algorithms.” *Parallel Computing,* Vol. 27, No. 3, pp. 201-222, 2001.

43 Rosenkrantz, D. J., Stearns, R. E. and Lewis, P. M. (1974, 14-16 Oct. 1974). *Approximate Algorithms for the Traveling Salesperson Problem.* Paper presented at the 15th Annual Symposium on Switching and Automata Theory (swat 1974).

44 Silva, M. M., Subramanian, A. and Ochi, L. S. “An Iterated Local Search Heuristic for the Split Delivery Vehicle Routing Problem.” *Computers & Operations Research,* Vol. 53, No., pp. 234-249, 2015.

45 Solomon, M. M. “Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints.” *Operations Research,* Vol. 35, No. 2, pp. 254-265, 1987.

46 Subramanian, A., Drummond, L. M. A., Bentes, C., Ochi, L. S. and Farias, R. “A Parallel Heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery.” *Computers & Operations Research,* Vol. 37, No. 11, pp. 1899-1911, 2010.

47 Tesco. (2017). Delivery Options. from http://www.tesco.com/groceries/zones/default.aspx?name=delivery-options

48 Xiong, Y., Gulczynski, D., Kleitman, D., Golden, B. and Wasil, E. “A Worst-case Analysis for the Split Delivery Vehicle Routing Problem with Minimum Delivery Amounts.” *Optimization Letters,* Vol. 7, No. 7, pp. 1597-1609, 2013.

49 Yellow, P. C. “A Computational Modification to the Savings Method of Vehicle Scheduling.” *Operational Research Quarterly,* Vol. 21, No. 2, pp. 281-283, 1970.

附錄 A　SDVRP 標竿例題 SD16_144 最佳結果

表 A.1 為本研究對 SDVRP 標竿例題 SD16_144 的最佳求解結果。本例題的最佳解結果具有特色，每條路線皆只服務兩個顧客；顧客被分送最多為 2 次，因此該結果為 *kmax* = 2 的解；此外，每部車輛中至少有一個顧客的需求被分送。

表 A.1　本研究求解 SDVRP 標竿例題 SD16_144 最佳結果

Problem: Problem: SDVRP	Benchmark set: C	Instance: SD16_144
BKS cost: 3381.26	Our best solution: 3381.28	Algorithm: SRC+RVND
Personal computer: 3.4 GHz i7-3770 processor, 16GB of RAM, Windows7		Computational time: 18.43s

Route No.	Customer (demand) serviced in sequence	Route No.	Customer (demand) serviced in sequence	Route No.	Customer (demand) serviced in sequence
1	0, 124(90), 123(10)*, 0	37	0, 114(50)*, 115(50)*, 0	73	0, 11(60), 10(40)*, 0
2	0, 144(90), 143(10)*, 0	38	0, 126(40)*, 125(60), 0	74	0, 13(50)*, 14(50)*, 0
3	0, 74(50)*, 75(50)*, 0	39	0, 134(40)*, 133(60), 0	75	0, 21(10)*, 20(90), 0
4	0, 90(40)*, 89(60), 0	40	0, 74(40)*, 73(60), 0	76	0, 27(60), 26(40)*, 0
5	0, 94(40)*, 93(60), 0	41	0, 140(90), 139(10)*, 0	77	0, 29(10)*, 28(90), 0
6	0, 108(90), 107(10)*, 0	42	0, 127(10)*, 128(90), 0	78	0, 33(10)*, 32(90), 0
7	0, 110(50)*, 111(50)*, 0	43	0, 99(50)*, 98(50)*, 0	79	0, 41(50)*, 42(50)*, 0
8	0, 130(40)*, 129(60), 0	44	0, 117(60), 118(40)*, 0	80	0, 39(60), 38(40)*, 0
9	0, 77(60), 78(40)*, 0	45	0, 135(50)*, 134(50)*, 0	81	0, 53(10)*, 52(90), 0
10	0, 87(50)*, 86(50)*, 0	46	0, 127(50)*, 126(50)*, 0	82	0, 57(10)*, 56(90), 0
11	0, 97(60), 98(40)*, 0	47	0, 75(10)*, 76(90), 0	83	0, 64(90), 65(10)*, 0
12	0, 103(10)*, 104(90), 0	48	0, 96(90), 95(10)*, 0	84	0, 69(10)*, 68(90), 0
13	0, 113(60), 114(40)*, 0	49	0, 111(10)*, 112(90), 0	85	0, 10(50)*, 9(50)*, 0
14	0, 121(60), 122(40)*, 0	50	0, 135(10)*, 136(90), 0	86	0, 12(90), 13(10)*, 0
15	0, 139(50)*, 138(50)*, 0	51	0, 92(90), 91(10)*, 0	87	0, 30(50)*, 29(50)*, 0
16	0, 79(10)*, 80(90), 0	52	0, 110(40)*, 109(60), 0	88	0, 30(40)*, 31(60), 0
17	0, 86(40)*, 85(60), 0	53	0, 119(10)*, 120(90), 0	89	0, 46(50)*, 45(50)*, 0
18	0, 95(50)*, 94(50)*, 0	54	0, 91(50)*, 90(50)*, 0	90	0, 46(40)*, 47(60), 0
19	0, 107(50)*, 106(50)*, 0	55	0, 4(90), 5(10)*, 0	91	0, 66(40)*, 67(60), 0
20	0, 123(50)*, 122(50)*, 0	56	0, 17(50)*, 18(50)*, 0	92	0, 65(50)*, 66(50)*, 0
21	0, 141(60), 142(40)*, 0	57	0, 22(40)*, 23(60), 0	93	0, 2(50)*, 1(50)*, 0
22	0, 82(40)*, 81(60), 0	58	0, 34(50)*, 33(50)*, 0	94	0, 18(40)*, 19(60), 0
23	0, 82(50)*, 83(50)*, 0	59	0, 40(90), 41(10)*, 0	95	0, 36(90), 37(10)*, 0
24	0, 100(90), 99(10)*, 0	60	0, 50(40)*, 51(60), 0	96	0, 54(40)*, 55(60), 0
25	0, 118(50)*, 119(50)*, 0	61	0, 58(40)*, 59(60), 0	97	0, 2(40)*, 3(60), 0
26	0, 138(40)*, 137(60), 0	62	0, 70(50)*, 69(50)*, 0	98	0, 14(40)*, 15(60), 0
27	0, 143(50)*, 142(50)*, 0	63	0, 6(40)*, 7(60), 0	99	0, 22(50)*, 21(50)*, 0
28	0, 88(90), 87(10)*, 0	64	0, 16(90), 17(10)*, 0	100	0, 37(50)*, 38(50)*, 0
29	0, 106(40)*, 105(60), 0	65	0, 24(90), 25(10)*, 0	101	0, 45(10)*, 44(90), 0
30	0, 131(10)*, 132(90), 0	66	0, 34(40)*, 35(60), 0	102	0, 54(50)*, 53(50)*, 0
31	0, 115(10)*, 116(90), 0	67	0, 42(40)*, 43(60), 0	103	0, 62(50)*, 61(50)*, 0
32	0, 131(50)*, 130(50)*, 0	68	0, 50(50)*, 49(50)*, 0	104	0, 1(10)*, 72(90), 0
33	0, 78(50)*, 79(50)*, 0	69	0, 58(50)*, 57(50)*, 0	105	0, 62(40)*, 63(60), 0
34	0, 83(10)*, 84(90), 0	70	0, 70(40)*, 71(60), 0	106	0, 25(50)*, 26(50)*, 0
35	0, 102(40)*, 101(60), 0	71	0, 5(50)*, 6(50)*, 0	107	0, 49(10)*, 48(90), 0
36	0, 102(50)*, 103(50)*, 0	72	0, 9(10)*, 8(90), 0	108	0, 61(10)*, 60(90), 0

*denotes a split delivery

附錄 B　SDVRP-MDA 標竿例題 S76D4_75 (p = 0.2) 最佳結果

表 B.1 為本研究對 SDVRP-MDA 標竿例題 S76D4_75 (p = 0.2) 求解過程中的最佳結果，該結果突破了目前已知最佳解。在這個例題結果中，被分送次數最多的兩個顧客為 11 以及 58，此兩點皆被分送 3 次。此外，亦可發現被分送的顧客數目最多的路線為路線 1，該路線有四個被分送的顧客，即 36、37、47 以及 48。

表 B.1　本研究求解 SDVRP-MDA 標竿例題 S76D4_75 (p = 0.2) 最佳結果

Problem: SDVRP-MDA	Benchmark set: B	Instance: S76D4_75 (p = 0.2)
BKS cost: 2118.86	Our best solution: 2103.78	Algorithm: SRC+VND$_{all}$
Personal computer: 3.9 GHz i7 processor, 16GB of RAM, Windows7		Computational time: 2837.47s

Route No.	Customer (demand) serviced in sequence	Route No.	Customer (demand) serviced in sequence
1	0, 48(7)*, 47(18)*, 36(21)*, 60(98), 37(16)*, 0	20	0, 30(5)*, 21(143), 28(10)*, 0
2	0, 33(14)*, 64(121), 22(25), 0	21	0, 13(37)*, 54(41), 19(82), 0
3	0, 25(39)*, 55(55), 18(26), 50(35), 0	22	0, 48(21)*, 5(36), 29(84), 4(18)*, 0
4	0, 20(37), 70(87), 37(36)*, 0	23	0, 2(20)*, 74(103), 30(19)*, 4(13)*, 0
5	0, 51(14)*, 16(25), 24(44), 44(20), 32(47), 40(9)*, 0	24	0, 8(17)*, 53(143), 0
6	0, 14(29), 59(101), 11(30)*, 0	25	0, 67(143), 0
7	0, 47(68)*, 69(89), 0	26	0, 12(17)*, 39(143), 0
8	0, 23(35), 56(73), 1(52)*, 0	27	0, 6(22)*, 63(131), 0
9	0, 58(27)*, 10(33), 31(100), 0	28	0, 36(21)*, 71(139), 0
10	0, 6(17)*, 43(126), 33(17)*, 0	29	0, 58(12)*, 72(143), 12(5)*, 0
11	0, 11(80)*, 66(72), 38(8)*, 0	30	0, 68(138), 0
12	0, 28(37)*, 61(121), 0	31	0, 8(24)*, 35(133), 0
13	0, 40(24)*, 25(85)*, 9(51), 0	32	0, 34(24), 46(31), 52(49), 27(50), 0
14	0, 3(139), 0	33	0, 4(34)*, 45(126), 0
15	0, 57(42)*, 15(118), 0	34	0, 7(143), 0
16	0, 2(26)*, 62(60), 73(74), 0	35	0, 26(24), 17(135), 0
17	0, 58(10)*, 38(17)*, 65(105), 11(28)*, 0	36	0, 75(124), 0
18	0, 13(65)*, 57(93)*, 0	37	0, 41(96), 42(27), 1(34)*, 0
19	0, 51(18)*, 49(141), 0		

*denotes a split delivery

附錄 C　SDVRP-LND 標竿例題 S76D2_75 (*kmax* = 2) 最佳結果

表 C.1 為本研究對 SDVRP-LND 標竿例題 S76D2_75 (*kmax* = 2) 求解的最佳結果。關於本題的路線結果，在 75 個顧客中共有 10 個顧客點被兩部車輛分送，分別是顧客 4、11、17、26、48、51、52、67、68 以及 73；其餘顧客則無分割配送的情況。路線 12 為服務最多分送顧客的路線，包含顧客 4、52 以及 67。由於本研究對本例題在 *kmax* = 3 未求得較佳的結果，因此下列 (*kmax* = 2) 的結果亦為本例題在 *kmax* = 3 的已知最佳解。

表 C.1　本研究求解 SDVRP-LND 標竿例題 S76D2_75 (*kmax* = 2) 最佳結果

Problem: SDVRP-LND	Benchmark set: B	Instance: S76D2_75 (*kmax* = 2)
BKS cost: 1098.33	Our best solution: 1098.20	Algorithm: SRC+RVND
Personal computer: 3.9 GHz i7 processor, 16GB of RAM, Windows7		Computational time: 13.82s

Route No.	Customer (demand) serviced in sequence
1	0, 48(7)*, 69(22), 71(18), 60(45), 70(46), 20(22), 0
2	0, 43(27), 41(43), 42(46), 64(38), 73(6)*, 0
3	0, 9(44), 25(35), 55(21), 18(23), 50(29), 17(8)*, 0
4	0, 67(6)*, 35(35), 14(18), 59(20), 66(47), 11(34)*, 0
5	0, 48(37)*, 47(25), 36(33), 21(23), 74(40), 0
6	0, 51(8)*, 49(32), 24(24), 56(47), 23(46), 0
7	0, 26(9)*, 58(29), 10(17), 31(39), 39(31), 72(16), 12(19), 0
8	0, 68(10)*, 2(22), 28(26), 61(16), 22(29), 62(40), 73(17)*, 0
9	0, 17(14)*, 40(37), 32(29), 44(41), 3(22), 51(15)*, 0
10	0, 5(47), 37(47), 15(46), 57(20), 0
11	0, 26(26)*, 38(47), 65(19), 11(8)*, 53(26), 7(34), 0
12	0, 67(12)*, 34(46), 52(43)*, 27(43), 4(16)*, 0
13	0, 52(4)*, 13(35), 54(16), 19(47), 8(18), 46(40), 0
14	0, 6(20), 33(47), 1(44), 63(19), 16(20), 0
15	0, 4(2)*, 45(38), 29(43), 30(41), 68(18)*, 75(18), 0

*denotes a split delivery

簡例

中文姓名：朱佑旌

英文姓名：Yu-Ching Chu

出生日期：民國 71 年 7 月 10 日

連絡地址：高雄市楠梓區宏毅二路南三巷 58 號

連絡電話：0919188667

電子信箱：youjingchu.tem97g@g2.nctu.edu.tw

簡歷：

2017 年 6 月　　國立交通大學運輸與物流管理學系博士班畢業 (博士)

2006 年 6 月　　中華大學科技管理研究所甲組畢業 (碩士)

2004 年 6 月　　中華大學運輸與物流管理學系畢業 (學士)