萬用啟發式演算法期末專題報告

- 組別：第 4 組
- 組員：311703003 馮成林、311703007 黃怡棻、0813224 徐新庭
- 期刊論文：Vincent, F. Y., Jewpanya, P., Redi, A. P., & Tsao, Y. C. (2021). Adaptive neighborhood simulated annealing for the heterogeneous fleet vehicle routing problem with multiple cross-docks. *Computers & Operations Research*, *129*, 105205.

一、 緒論

我們期末專題實作的文獻是要解決 Heterogeneous Fleet Vehicle Routing Problem with Multiple Cross-docks（HF-VRPCD）這個問題，主要是從以下兩個問題延伸而來：

1. Vehicle Routing Problem with Pickup and Delivery（VRPPD）

車輛路徑問題（VRP）的環境設定基本上就是設有一個場站（depot）、多部有容量限制的車輛，及多個已知貨物需求量的客戶，車輛必須由場站出發服務各個客戶。而加入提供貨物的客戶，產生取貨的概念，即為收送貨問題（Pickup and delivery），在此問題下車輛會同時有收貨及送貨的服務，且會將貨物從供給的客戶點直接配送至需求的客戶點，中間不經過場站。此問題必須滿足以下限制：

(1) 車輛起迄點皆為同一個場站。

(2) 每個客戶需求都得被服務到。

(3) 一個客戶只會被一輛車所服務。

(4) 需求量不得超過車子容量限制。

2. Vehicle Routing Problem with Cross-Dock（VRPCD）

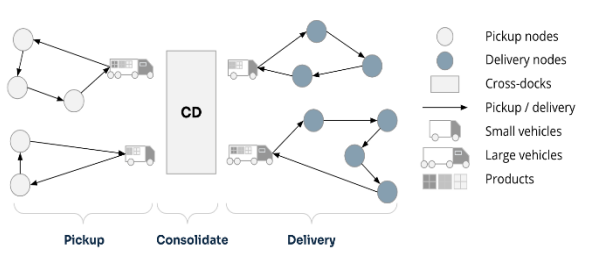
前述的 VRPPD 中有同時考慮收送貨，為了提高整體物流模式的效率，因此發展出越庫作業（Cross-docking）。其有三個主要步驟：

(1) 收貨（Pickup）：車輛從 Cross-dock 出發，將貨物從各個收貨點收至 Cross-dock。

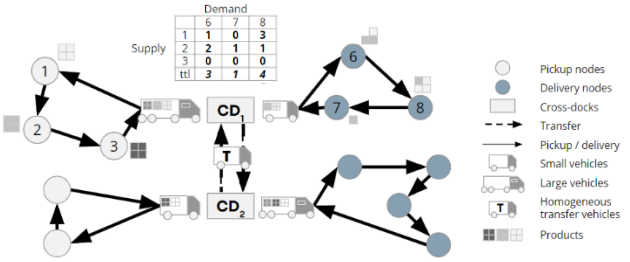
(2) 集貨（Consolidate）：在 Cross-dock 整貨、理貨，並將所有貨物重新分配。

(3) 送貨（Delivery）：將貨物由 Cross-dock 送往各個需求客戶點。

VRPCD 須滿足上述 VRPPD 提到的所有限制式，唯車輛起迄點須在同一 Cross-dock。



圖一 Vehicle Routing Problem with Cross-Dock 示意圖



圖二 本篇文獻的問題環境示意圖

二、 文獻回顧

文獻可以分為兩個大方向進行回顧，分別是考量到問題種類的文獻，以及演算法求解的文獻。參照表一及表二。

表一 各文獻之環境假設

文獻	環境假設	重點描述
問題 1：Vehicle Routing Problem with Cross Docking（VRPCD）		
Lee et al. (2006) / Liao et al. (2010)	單一 Cross-dock	• 抵達 Cross-dock 的時間一致（盡可能同時） • 將貨物進行分類
Guastaroba et al. (2016)	多個 Cross-dock 可以作為起迄點及理貨中心	考量到現實情況及時間、資源等限制所設置的多個 Cross-dock
問題 2：Heterogeneous Fleet Vehicle Routing Problem（HF-VRP）		
Baldacci et al. (2008)	考量異質車隊的變動成本	考量到現有的物流公司多為異質車隊（大、小容量的車）
本篇文獻考慮的問題： Heterogeneous Fleet Vehicle Routing Problem with Multiple Cross-docks（HF-VRPCD）		
本篇文獻	考量到異質車隊及多個 Cross-dock 的車輛路徑問題	

表二 各文獻採用之演算法

文獻	環境假設	演算法
Yu et al. (2016)	開放式的車隊問題（open vehicle routing problem）	利用 Simulated Annealing（SA）求解，發現該演算法求解類似問題較有效率。
Demir et al. (2012) Gunawan et al. (2020) Masson et al. (2013) Sacramento et al. (2019)	車輛路徑問題（vehicle routing problem）	將自適應（adaptive）機制加入 Large Neighborhood Search（LNS）演算法中，發現此機制能夠幫助演算法在良好的效率下得到得出更佳品質的解。
本篇文獻	考量到異質車隊及多個 Cross-dock 的車輛路徑問題（HF-VRPCD）	將自適應機制加入 SA 演算法，提升求解效率，並得出平均而言較好的解。

三、 問題描述

本文獻所提出的問題為考量到多部異質車輛、並且是多個 Cross-dock 的問題，如圖二所示。除了須滿足上述 VRPCD 的所有限制，另有幾點環境設定：

1. 起訖點相同：各收貨路線及送貨路線最後都會回到一開始出發的 Cross-dock。
2. 總體運行時間限制：整體配送作業需要在給定的時限內完成。
3. 供需關係：問題中的每個送貨點對於每一個收貨點都可能會有貨物需求，且其需求量可能都不同，而所有顧客對於某一收貨點的總需求量為該收貨點供給的總量。
4. 轉貨（Transfer）機制：由於每個顧客被分配到的最佳 Cross-dock 在供給、需求可能會無法配對上，因此在各個 Cross-dock 之間加入轉貨機制，可以讓顧客順利且有效率的被服務。
5. Return Time Gap 定義：Return Time Gap 為收貨過程中，「最晚返回 Cross-dock 車輛」之返回時間與「最早返回 Cross-dock 車輛」之返回時間的差值。這是我們自己延伸的設定，原論文的環境並沒有考慮這個要素。由於原論文的文獻回顧中有提到，車輛返回 Cross-dock 的時間應盡可能同時，以利後續的集貨作業，為了反應這個現實情況，我們自行定義了 Return Time Gap，並且希望能夠將這個時間差值盡可能縮短，當 Return Time Gap 越短，表示各車輛回到 Cross-dock 的時間越接近同時，因此我們會將此時間差值乘上一個單位時間的懲罰值，加入目標式做考慮。

本問題所要決定的有（1）每個顧客點是由哪一部車服務；（2）每台車的運行路線為何；（3）每台車的路線是從哪一個 Cross-dock 出發（由於各車輛最後也會回到一開始出發的 Cross-dock，且路線與車輛一一對應，因此以下會以各路線「使用」哪一個 Cross-dock 來描述）。而其所要達到的目標為最小化總運輸成本，包含收貨過程的運輸成本、轉貨過程的運輸成本、送貨過程的運輸成本，以及 Return Time Gap 所造成的懲罰成本。

四、 演算法

本篇文獻提出的演算法是 Adaptive Neighborhood Simulated Annealing（ANSA），以下會先說明演算法中解的編碼方式，而後說明起始解的產生、ANSA 尋找鄰域解（Neighborhood solution）的移步（Move），最後則是 ANSA 的流程。

4.1 解的編碼方式

在這個問題中總共有三種點集合：

D：Cross-dock 集合，$D=\{1,2,...,c\}$，其中 c 為 Cross-dock 數量；

M：收貨點集合，$M=\{c+1,c+2,...,c+p\}$，其中 p 為收貨點數量；

N：送貨點集合，$N=\{c+p+1,c+p+2,...,c+p+d\}$，其中 d 為送貨點數量；

除了點集合外，還有收送貨車輛的集合：

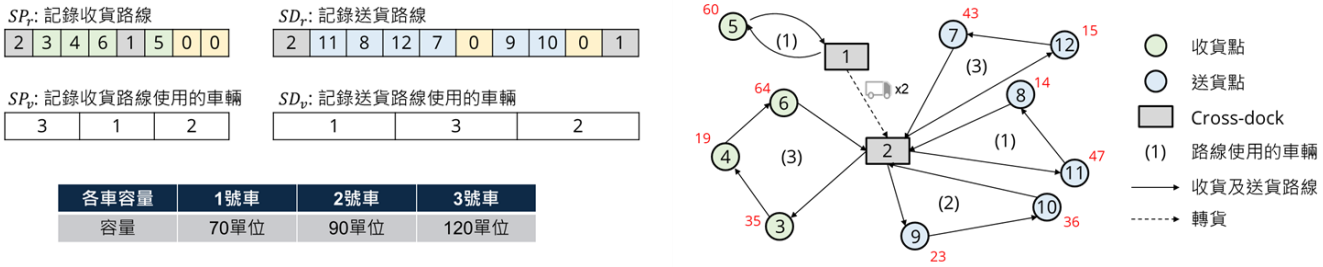
V：收送貨車輛集合，$V=\{1,2,...,v\}$，其中v為收送貨車輛數。

演算法中以SP_r及SD_r兩個串列分別記錄收貨路線及送貨路線，其中SP_r（SD_r）由D、M（N）裡面的點及$v-1$個 0 所構成；並以SP_v及SD_v兩個串列分別記錄個收貨路線及送貨路線所使用的車，皆由V當中的收送貨車輛所構成，SP_v及SD_v依照各收貨路線及送貨路線讀取的順序、由左至右記錄各路線所使用的車輛。圖三為此四個串列對應到的路線圖例子，圖中各收貨點（送貨點）旁邊的紅色數字為該點的供給量（需求量）。

從原文獻計算目標值的流程中，我們整理出了SP_r及SD_r基本讀取規則：串列上任兩相鄰的編號對應到這兩個點形成的節線，其方向為由左邊的點移動到右邊的點，如圖三中SP_r第一個編號對應到 2 號 Cross-dock，第二個編號對應到 3 號收貨點，這兩個編號就對應到 2 號點移動到 3 號點的節線。因此，由左至右讀取SP_r及SD_r，即可將收貨路線及送貨路線分別連接出來。

除了基本讀取規則外，讀取過程中還會碰到三種狀況需進行特殊處理。當碰到這三種狀況時，基本上都是讓路線回到一開始出發的 Cross-dock，以結束當前路線的讀取，並接著開始讀取下一條路線。以下為三種狀況的說明：

1. 當讀取到「移動至 Cross-dock」的節線時，需將當前讀取的路線連接回一開始出發的 Cross-dock。如圖三中SP_r的第五個編號對應到 1 號 Cross-dock，因此第一條讀取的收貨路線需從 6 號點連結回 2 號點，以結束掉該條路線的讀取，而下一個讀取到的節線則依照基本讀取規則處理，即可接續讀取下一條收貨路線。
2. 當下一個點的供給（需求）加進到路線以後，會超過車輛的容量時，就需將當前讀取的路線連接回一開始出發的 Cross-dock，並改由下一個順位的車輛從同樣的 Cross-dock 出發，接續去服務先前未放入路線的點。如圖三的SD_r，若將 12 號點加入到第一條送貨路線，會造成路線的總需求量超過 1 號車的容量（47+14+15 大於 70），因此需從 11 號點連結回 2 號點，以結束掉該條路線的讀取，並改由 3 號車從 2 號 cross-dock 出發，接續服務 12 號點，同時開啟下一條送貨路線的讀取。
3. 當讀取到移動至「0」的節線時，需將當前讀取的路線連接回一開始出發的 cross-dock，且讀取至下一條節線時，需將「0」轉換為前一條路線所使用的 cross-dock。如圖三中SD_r的第六個編號為 0，因此讓第二條送貨路線從 7 號點連接回 2 號點，以結束掉該條路線的讀取；下一條節線是從 0 移動到 9，須將它解讀為從 2 移動到 9，也就是從前一條送貨路線使用的 Cross-dock 出發，由下一個順位的車輛（2 號車）去服務 9 號點，同時開啟下一條送貨路線的讀取。

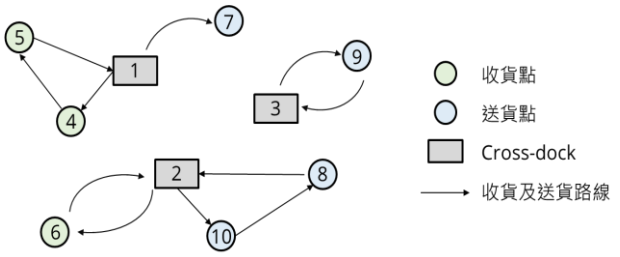


圖三　解的編碼方式之例子

透過基本讀取規則搭配三種特殊狀況的處理，就可以完整讀取出所有收貨路線及送貨路線。SP_r（SD_r）讀取的過程會在所有的收貨點（送貨點）都放到路線當中就停止。讀取完成後，即可計算這個解的收貨（送貨）過程運輸成本，即是將收貨路線（送貨路線）上所有節線的旅行距離乘以對應車輛的單位距離運輸成本，進行加總而得。

接著是轉貨的判斷：得知收貨路線及送貨路線後，我們可以知道各個客戶是由哪個 Cross-dock 派遣車輛去收送貨，因此考慮各個收貨點及送貨點配對，可以整理收貨點及送貨點的 Cross-dock 配對表，如表三所示，其配對來自於圖四的收送貨路線。表三的各個欄位中，最左邊的編號為收貨點被指派到的 Cross-dock；中間的編號則是送貨點被指派到的 Cross-dock，而最右邊的數字則代表對應的送貨點需要對應的收貨點多少單位的貨物；當同一個欄位的兩個 Cross-dock 不同，就需要進行轉貨。舉例而言，4 號收貨點使用 1 號 Cross-dock，但 8 號送貨點使用 2 號 Cross-dock，而 8 號送貨點對於 4 號收貨點的需求量為 47 單位，因此從 1 號 Cross-dock 到 2 號 Cross-dock 的轉貨量就會增加 47 單位。針對表三各個欄位都進行判斷，將各個 Cross-dock 之間的轉貨量進行累加後，即可知道各個 Cross-dock 之間的總轉貨量，由於每部轉貨車輛的容量相同，我們可以進一步計算各個 Cross-dock 之間所需要的轉貨車輛數。舉例而言，假設轉貨車輛的容量是 70 單位，根據表三的資訊進行計算後，可得知從 1 號 Cross-dock 到 2 號 Cross-dock 的總轉貨量是 161 單位（47+33+29+52），因此需要 3 台轉貨車輛（161 除以 70 後無條件進位到整數）。

得知各 Cross-dock 之間需要幾台轉貨車輛，再分別乘上兩 Cross-dock 之間的運輸成本（套用表三的例子，假設從 1 號點到 2 號點的運輸成本為 249 單位，從 1 號點轉貨到 2 號點的轉貨成本就是 747 單位，等於 249 單位乘以 3 台車），全部進行加總後，即為轉貨過程的運輸成本。



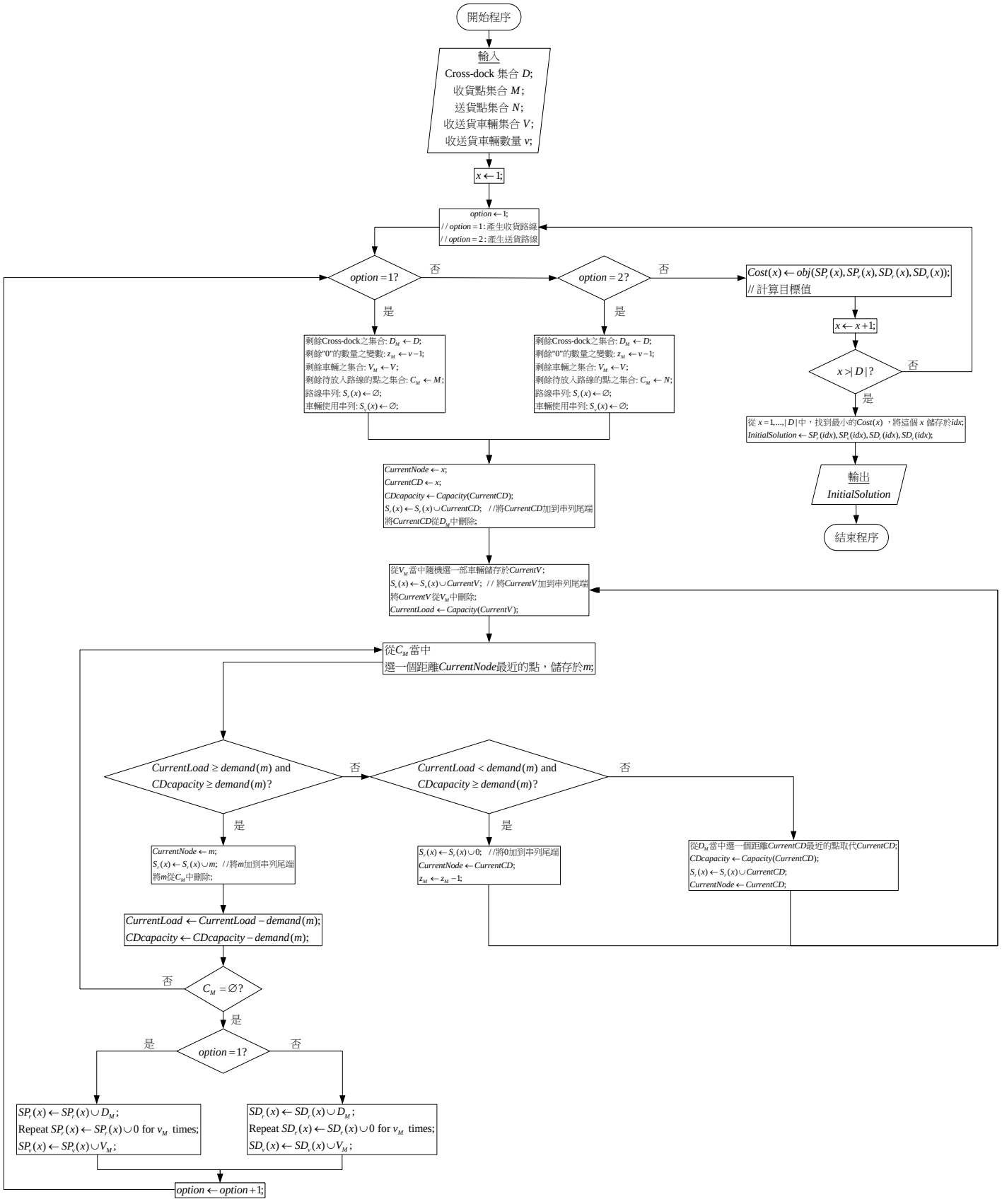
圖四 另一個解的收貨及送貨路線圖

表三 收貨點、送貨點配對表

收貨點＼送貨點	7	8	9	10
4	(1,1,0)	(1,2,47)	(1,3,0)	(1,2,29)
5	(1,1,62)	(1,2,33)	(1,3,0)	(1,2,52)
6	(2,1,0)	(2,2,15)	(2,3,64)	(2,2,16)

4.2 起始解的產生

起始解是以「貪婪(greedy)」的策略產生，每一步都朝當前可見的最低成本前進。起始解產生過程的首要目標是減少轉貨成本，採行的策略是盡可能讓所有的收貨路線以及送貨路線都使用同一個 Cross-dock，或者是距離較近的幾個 Cross-dock，以盡可能減少轉貨過程的運輸成本；次要目標則是減少收送貨過程的運輸成本，路線的產生是以逐點連接成路線的方式進行，每次都尋找距離最近的、且還沒被放到路線的點作為下一個要連接的點，也就是以最小的運輸成本作為下一個點的挑選準則。圖五為起始解產生的流程。


開始程序
輸入
Cross-dock 集合 D;
收貨點集合 M;
送貨點集合 N;
收送貨車輛集合 V;
收送貨車輛數量 v;
x ← 1;
option ← 1;
// option = 1: 產生收貨路線
// option = 2: 產生送貨路線
option = 1?
否
option = 2?
否
是
是
Cost(x) ← obj(SP_r(x), SP_v(x), SD_r(x), SD_v(x));
// 計算目標值
x ← x + 1;
x >| D |?
否
是
剩餘Cross-dock之集合: D_M ← D;
剩餘"0"的數量之變數: z_M ← v − 1;
剩餘車輛之集合: V_M ← V;
剩餘待放入路線的點之集合: C_M ← M;
路線串列: S_r(x) ← ∅;
車輛使用串列: S_v(x) ← ∅;
剩餘Cross-dock之集合: D_M ← D;
剩餘"0"的數量之變數: z_M ← v − 1;
剩餘車輛之集合: V_M ← V;
剩餘待放入路線的點之集合: C_M ← N;
路線串列: S_r(x) ← ∅;
車輛使用串列: S_v(x) ← ∅;
從 x = 1,...,| D |中，找到最小的Cost(x) ，將這個 x 儲存於idx;
InitialSolution ← SP_r(idx), SP_v(idx), SD_r(idx), SD_v(idx);
輸出
InitialSolution
結束程序
CurrentNode ← x;
CurrentCD ← x;
CDcapacity ← Capacity(CurrentCD);
S_r(x) ← S_r(x) ∪ CurrentCD; //將CurrentCD加到串列尾端
將CurrentCD從D_M中刪除;
從V_M當中隨機選一部車輛儲存於CurrentV;
S_v(x) ← S_v(x) ∪ CurrentV; //將CurrentV加到串列尾端
將CurrentV從V_M中刪除;
CurrentLoad ← Capacity(CurrentV);
從C_M當中
選一個距離CurrentNode最近的點，儲存於m;
CurrentLoad ≥ demand(m) and
CDcapacity ≥ demand(m)?
否
CurrentLoad < demand(m) and
CDcapacity ≥ demand(m)?
否
是
是
CurrentNode ← m;
S_r(x) ← S_r(x) ∪ m; //將m加到串列尾端
將m從C_M中刪除;
S_r(x) ← S_r(x) ∪ 0; //將0加到串列尾端
CurrentNode ← CurrentCD;
z_M ← z_M − 1;
從D_M當中選一個距離CurrentCD最近的點取代CurrentCD;
CDcapacity ← Capacity(CurrentCD);
S_r(x) ← S_r(x) ∪ CurrentCD;
CurrentNode ← CurrentCD;
CurrentLoad ← CurrentLoad − demand(m);
CDcapacity ← CDcapacity − demand(m);
否
C_M = ∅?
是
是
option = 1?
否
SP_r(x) ← SP_r(x) ∪ D_M;
Repeat SP_r(x) ← SP_r(x) ∪ 0 for v_M times;
SP_v(x) ← SP_v(x) ∪ V_M;
SD_r(x) ← SD_r(x) ∪ D_M;
Repeat SD_r(x) ← SD_r(x) ∪ 0 for v_M times;
SD_v(x) ← SD_v(x) ∪ V_M;
option ← option + 1;

圖五 起始解產生流程

4.3 尋找鄰域解的移步（Move）

ANSA 使用三種尋找鄰域解的移步方式，並且在每次迭代中依照各方法的選擇機率選取一種方式，然後再以相同的機率從SP_r、SD_r、SP_v、SD_v四個串列中隨機選取一個來產生鄰域解。各個尋找鄰域解的方式都是先根據選擇的串列長度，產生不相同且不超過長度的兩個隨機數，再來以兩個隨機數在串列中的對應位置，來改變串列以產生鄰域解。以下使用長度為 8 的收貨路徑{2, 4, 3, 1, 5, 6, 0, 0}為例:

1. 交換(Swap)：

   假設產生兩個隨機數 4 和 2，在串列中對應位置的值為 1 和 4，接著將兩個位置的值進行交換，得到新的收貨路徑{2, 1, 3, 4, 5, 6, 0, 0}，以此產生新的鄰域解。

2. 倒轉(Reverse)：

   假設產生兩個隨機數 1 和 7，在串列中對應位置的值為 2 和 0，在這兩者之間的所有數字之順序進行倒轉，得到新的收貨路徑{2, 6, 5, 4, 3, 1, 0, 0}，以此產生新的鄰域解。

3. 插入(Insertion)：

   產生兩個隨機數 X 和 Y，將 X 在串列中對應位置上的值插入到 Y 對應位置之前。假設產生兩個隨機數 2 和 5，但抽到的順序有所差別，會有兩種情形如下:

   a. X 為 2 以及 Y 為 5，對應位置的值為 4 跟 5，得到新的收貨路徑{2, 3, 1, 4, 5, 6, 0, 0}。
   b. X 為 5 以及 Y 為 2，對應位置的值為 5 跟 4，得到新的收貨路徑{2, 5, 4, 3, 1, 6, 0, 0}。

4.4 ANSA 流程

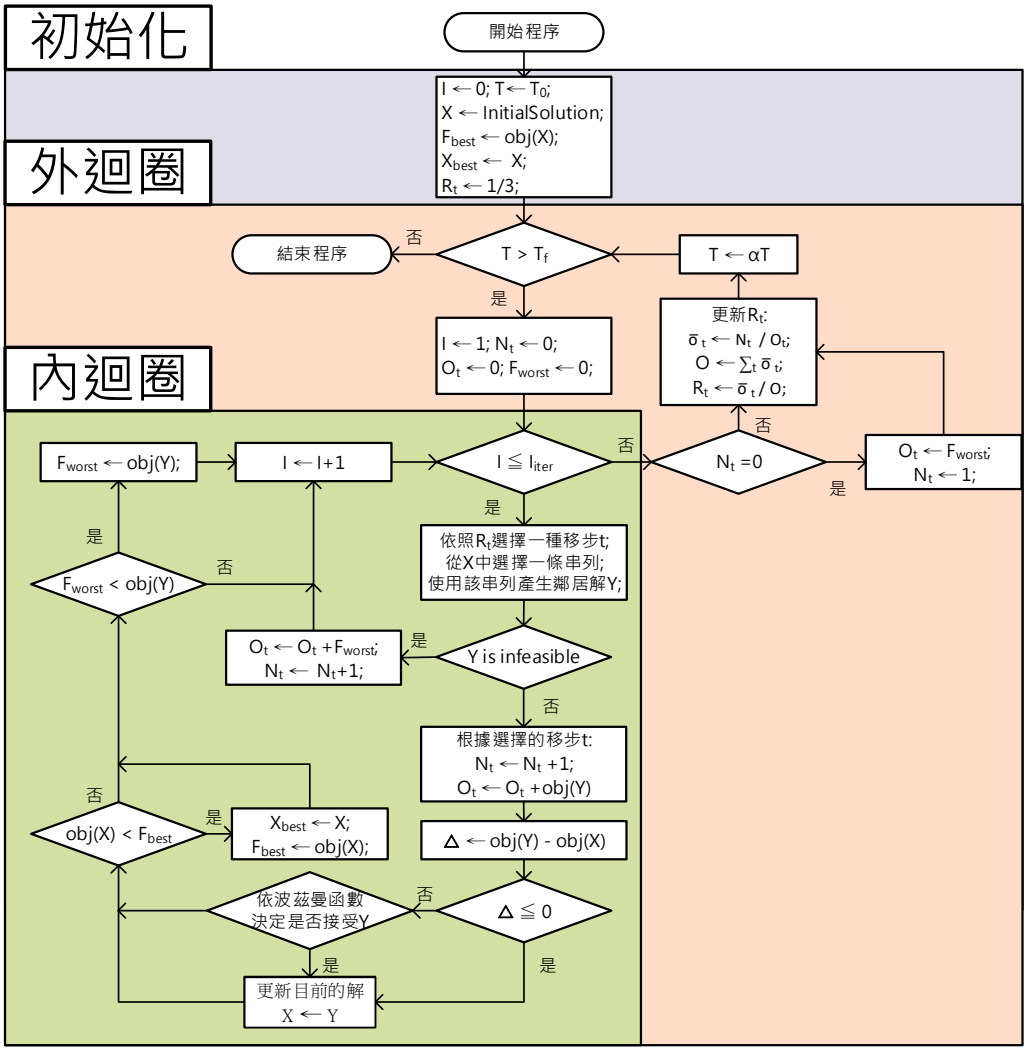
ANSA 共有 5 個輸入的參數，I_{iter}、T_0、T_f、α 和 K，分別代表每次溫度下的迭代次數、初始溫度、最低溫度、降溫速度和波茲曼常數。再來 ANSA 會使用到幾個與 SA 不同的變數，整理於表四。

表四 ANSA 變數

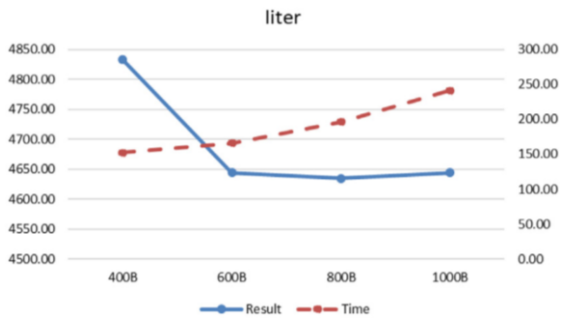
F_{worst}	紀錄目前溫度下最差的解的目標值
R_t	每種移步 t 被選擇到的機率, $t \in$ {交換, 倒轉, 插入}
O_t	移步 t 在目前溫度下的累積目標值
N_t	移步 t 在目前溫度下的累積被選擇到的次數
$\bar{O}_t$	移步 t 在目前溫度下的平均目標值的倒數
O	目前溫度下所有移步的平均目標值加總

ANSA 分為三個部分，分別為初始化、外迴圈和內迴圈。首先初始化的部分將目前溫度T設定為起始溫度T_0、以 4.2 小結提到的流程產生起始解，並將目前的解X和最佳解X_{best}設定為起始解，以及設定選擇機率R_t為 1/3。接著是外迴圈的部分，先將N_t、O_t和F_{worst}設定為 0，然後才執行內迴圈。而內迴圈結束後依照上述的變數值計算和更新R_t，若某一種移步 t 的N_t為 0，則需要先將它的N_t設為 1 以及O_t設為F_{worst}，確保不會出現除以 0 的情形，最後才降低

溫度。當目前溫度T是否低於最低溫度T_f時，跳出外迴圈和結束程序。在內迴圈的部分，首先依照R_t選擇一種移步方式，再從目前解 X 中選擇一個串列來產生鄰域解 Y，接著更新被選擇的次數加上 1 以及累積目標值加上鄰域解的目標值，然後判斷是否要接受 Y 為目前的解 X，若 Y 為不可行解則累積目標值改成加上F_{worst}並直接跳到下一次迭代。除此之外，還需要判斷 Y 是否為這個溫度下最差的解，然後更新F_{worst}，最後若是迭代次數到上限則跳出內迴圈。圖六為 ANSA 的流程圖。



圖六 ANSA 流程圖



圖七 參數設定例子

五、 實驗結果

5.1 參數設定

這篇論文使用 one factor at time(OFAT)的方式設定參數，也就是一次調整一個參數，比較不同數值下的目標值和執行時間，如圖七。而我們的實驗也採用跟原論文一樣的參數值，將I_{iter}設定為 600B、T_0為 3、T_f為 0.01、α 為 0.9 和 K為 1，其中 B 代表例子的總節點數。

5.2 結果分析

我們按照原論文的作法，以 SA 使用不同組合的固定機率去和 ANSA 比較，並且同個例子會執行 10 次，找出最低的目標值和計算平均目標值，表五為不同組合的選擇機率，表六為擷取我們部分例子的比較結果。從表六來看，SA 使用不同機率組合在某些例子有比較低的目標值。若以全部例子的平均來看，在最佳的目標值上，會是第一種組合表現最好，但是若是平均目標值就會是 ANSA 表現最好。由此可以推論出雖然 ANSA 無法保證能得到最好的解，但在無法知道哪種例子適用哪種機率組合，以及無法執行多次的情形下，適合使用 ANSA 來確保最差的情況（Worst case）不會太差。

表五 SA 的各個機率組合

機率組合	交換	倒轉	插入
1	1/3	1/3	1/3
2	1/4	1/4	1/2
3	1/2	1/4	1/4
4	1/4	1/2	1/4

表六 SA 與 ANSA 比較

例子	Best objective value					Average objective value				
	SA				ANSA	SA				ANSA
	1	2	3	4		1	2	3	4	
23	3234.9	3205.3	3349.6	3234.9	3234.9	3497.0	3394.1	3584.9	3501.9	3405.6
24	3027.3	2988.0	2987.2	3055.6	3076.2	3230.9	3232.4	3219.5	3256.4	3276.8
25	2828.3	2875.5	2931.0	2974.9	2859.8	3048.6	3100.4	3098.9	3132.8	3104.5
26	2822.3	2825.5	2825.5	2825.5	2825.5	2882.0	2909.4	2928.2	2959.4	2935.4
27	2962.0	2962.0	2967.0	2946.4	2960.1	3088.5	3053.8	3100.0	3090.2	3117.5
28	2830.3	2936.1	2890.3	2890.3	2890.3	3029.4	3042.2	3049.4	3040.5	3058.1
Total average	3136.8	3141.2	3155.0	3144.6	3151.6	3326.2	3328.2	3327.5	3328.1	3319.8

六、 結論

我們實際使用 C++來實作起始解的產生以及 ANSA 演算，然後按照論文的說明產生 HF-VRPMCD 的例子來求解，以進行論文中提到的實驗。除此之外，我們也針對原論文的問題和演算法進行補充和修正，如下幾點：

1. 在原論文的文獻回顧中有提及，在 Cross-docking 的問題中收貨車輛回到場站的時間應盡可能同時，但是原論文在數學模型和 ANSA 中都沒有相關的限制，因此我們自行延伸了問題，加入 Return Time Gap 的懲罰值於目標值中，並在實作時也有考慮進去。
2. 原論文在讀取解的四個串列時（原論文的 algorithm 2），簡化了轉貨的判斷流程，並沒有針對各個收貨點、送貨點的配對進行判斷，與其描述的問題環境不符，且簡化後的判斷流程也會少判斷到某些 Cross-dock 之間的轉貨，導致轉貨過程的運輸成本被低估，而無法計算出正確的目標值，針對這點我們也進行了修正。
3. 在原論文的 ANSA 流程中，並沒有對出現不可行解的情況提出解決方法，所以我們在原本 ANSA 的基礎上，將不可行解的目標值替換成目前搜尋到最差的目標值，以作為其找到不可行解的懲罰，並且跳過比較目前解的目標值之步驟。在實作中也有處理到這些額外加入的步驟。
4. 原論文計算 ANSA 移步選擇機率的方法會讓得到比較差的解之移步有更高的選擇機率，因此我們將計算$\bar{O}_t$改成平均目標值的倒數，以此解決這個問題。

附錄：文獻回顧中的論文

Lee, Y.H., Jung, J.W., Lee, K.M., 2006. Vehicle routing scheduling for cross-docking in the supply chain. Comput. Ind. Eng. 51 (2), 247–256.

Liao, C.-J., Lin, Y., Shih, S.C., 2010. Vehicle routing with cross-docking in the supply chain. Expert Syst. Appl. 37 (10), 6868–6873.

Guastaroba, G., Speranza, M.G., Vigo, D., 2016. Intermediate facilities in freight transportation planning: a survey. Transp. Sci. 50 (3), 763–789.

Baldacci, R., Battarra, M., Vigo, D., 2008. Routing a heterogeneous fleet of vehicles. In: Golden, B., Raghavan, S., Wasil, E. (Eds.), The Vehicle Routing Problem: Latest Advances and New Challenges. Springer, US, Boston, MA, pp. 3–27.

Yu, V.F., Jewpanya, P., Redi, A.A.N.P., 2016. Open vehicle routing problem with cross-docking. Comput. Ind. Eng. 94, 6–17.

Demir, E., Bektas", T., Laporte, G., 2012. An adaptive large neighborhood search heuristic for the pollution-routing problem. Eur. J. Operational Res. 223 (2), 346–359.

Gunawan, A., Widjaja, A.T., Vansteenwegen, P., Yu, V.F., 2020, July. Adaptive large neighborhood search for vehicle routing problem with cross-docking. In: 2020 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE.

Masson, R., Lehuédé, F., Péton, O., 2013. An adaptive large neighborhood search for the pickup and delivery problem with transfers. Transp. Sci. 47(3), 344–355.

Sacramento, D., Pisinger, D., Ropke, S., 2019. An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. Transp. Res. Part C: Emerg. Technol. 102, 289–315.