

考量配送司機優先順序的 車輛途程問題之求解演算法

期末報告

組別12：孫浩恩、徐新庭

指導老師：姚銘忠 教授



目錄

1. 期中回顧

2. 演算法介紹

3. 結果分析

4. 結論與建議

5. 參考文獻

6. Q&A補充用資料



1.



期中回顧

研究背景與動機

個案公司仍以**人工經驗**排程

建立**有系統性**的排程方法

研究背景與動機



無法求解**過大**的問題

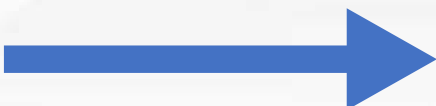
優先順序VRP模型

具有**運算時間過長**之問題

研究目的

 貼近現實狀況  擴大原模型之顧客點數量與分區數

修正司機無法綁定車輛路徑之問題  改善程式

改善程式運算時間過長  建立啟發式演算法

研究方法

3. 改善程式運算時間過長

 建立演算法之流程



產生結果並與Gurobi程式比較



分析比較結果與優化



2.

演算法介紹

演算法簡介

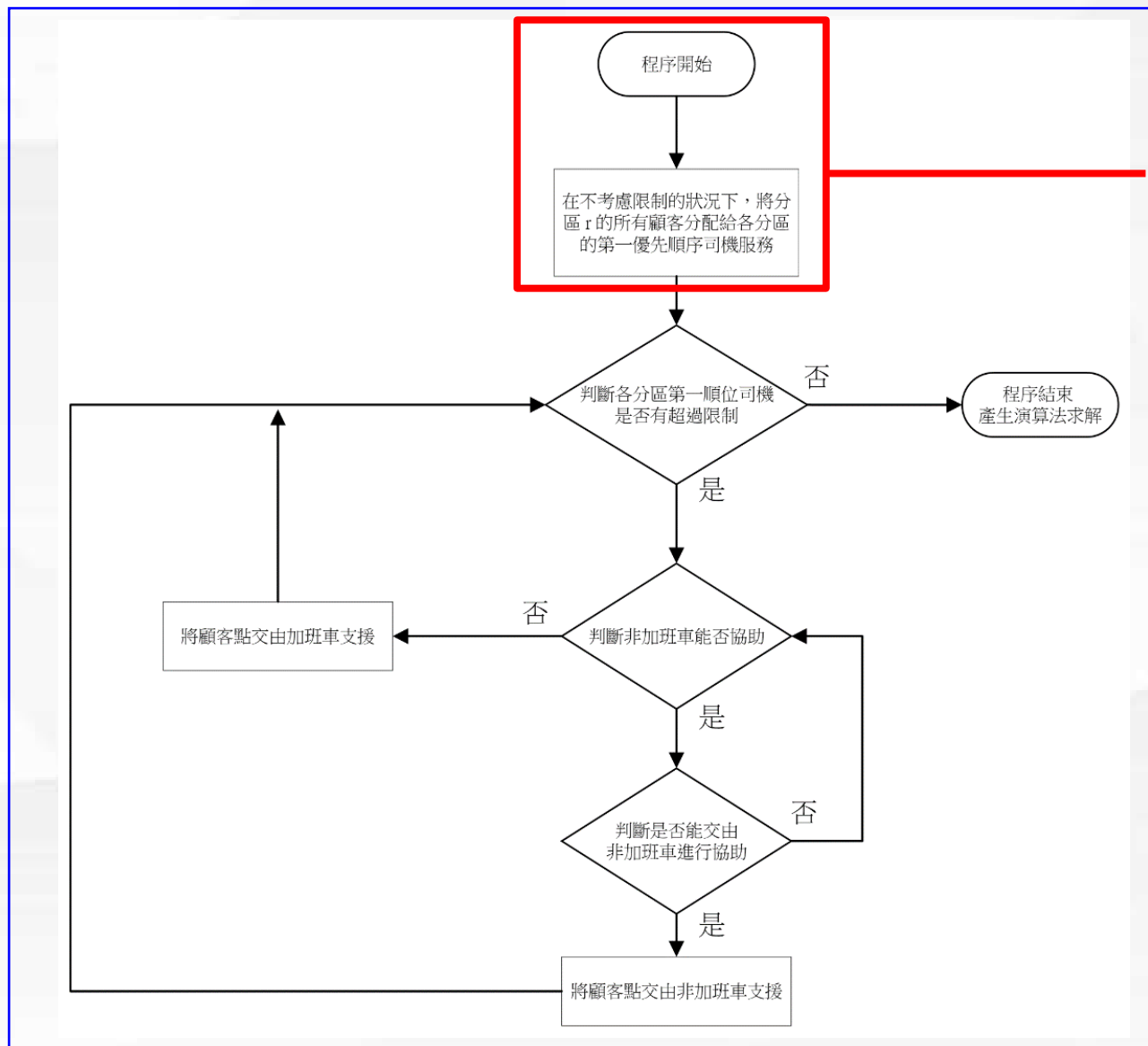
三大部分：

第一部分：初始路徑生成

第二部分：運用非加班車進行路徑修正

第三部分：運用加班車進行路徑修正

演算法流程介紹



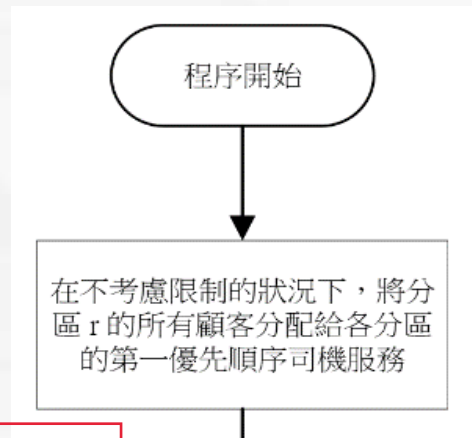
將所有顧客點交給各分區第一順位車輛

➤ 初始路徑生成

在**不考慮**容量和時間的限制下

將**所有**顧客點分配給各分區**第一優先**順位車輛服務

將所有顧客點運用**最短距離**來生成路徑



➤ 初始路徑生成之輸入輸出

主要Input:

車輛 k 的總載貨量 l_k

車輛 k 的總時間 s_k

車輛 k 之實際路徑 η_k

第一優先順序車輛路徑集合 γ_r ;

Output:

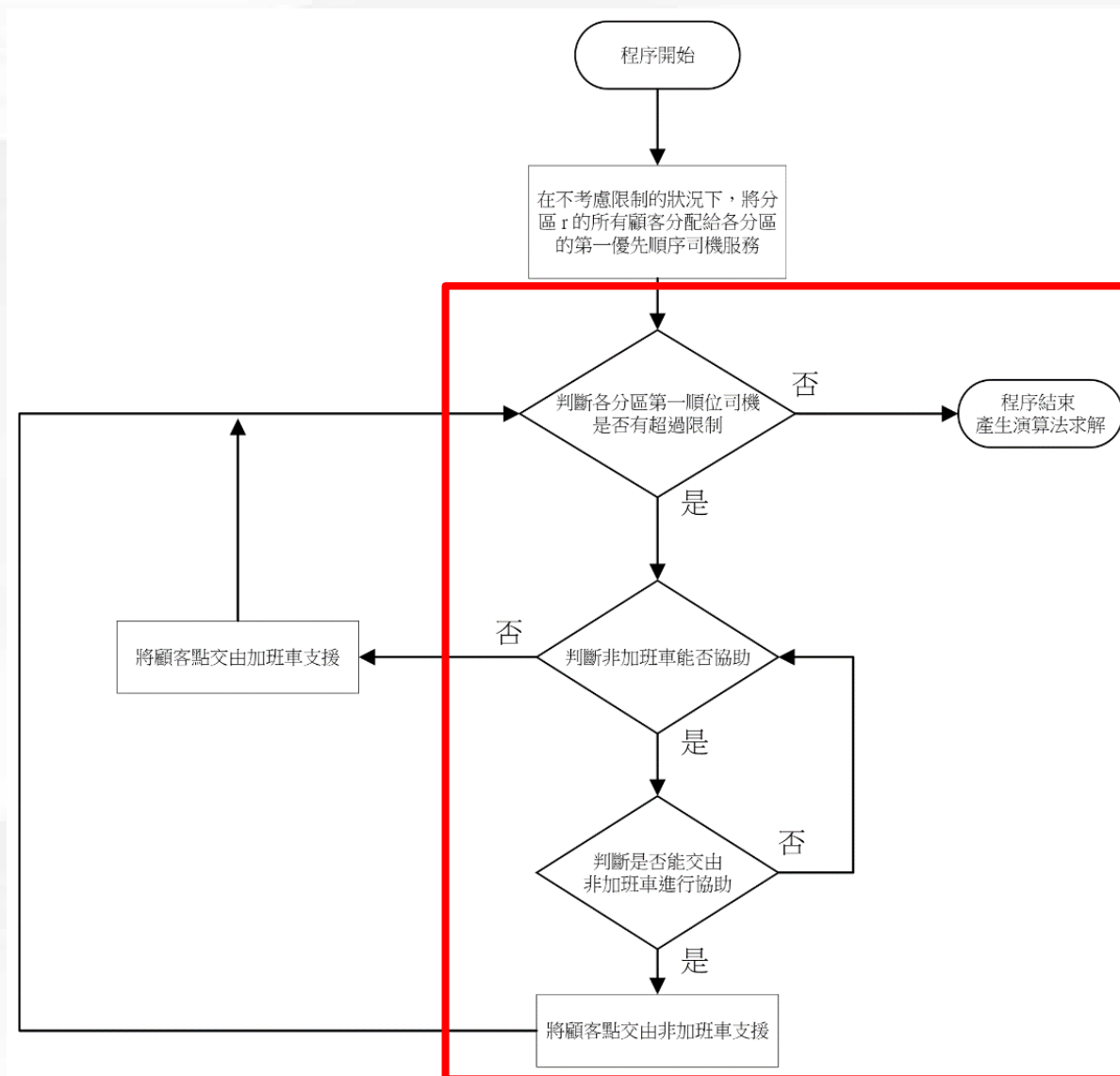
車輛 k 的總載貨量 l_k , $l_k = \sum q_i \ i \in \eta_k$

車輛 k 的總時間 s_k , $s_k = \sum \tau_{ij} + \sum \theta_i \ i \in \eta_k, j \in \eta_k$

車輛 k 之實際路徑 η_k ;

程序開始
↓
在不考慮限制的狀況下，將分區 r 的所有顧客分配給各分區的第一優先順序司機服務
↓

演算法流程介紹



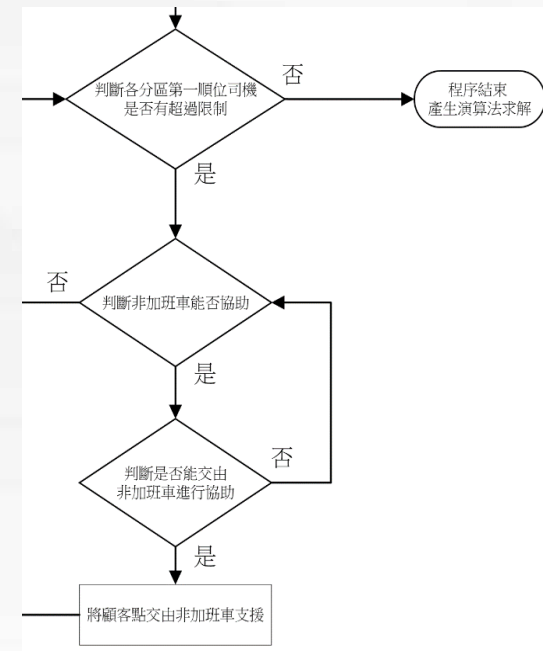
第二部分 運用非加班車修正路徑

運用非加班車進行路徑修正

Step1：檢查每部車輛是否**超過**容量和時間限制

Step2：將**超過**的車輛設定為被**協助車**，並尋找**未超出**限制的**非加班車**，將其設定為**支援車**。若**所有**非加班車都無法支援，則呼叫**加班車程序**(第三部分)

Step3：建立**測試用路徑**，並先測試將距離支援車分區質心**最近**的顧客點交由支援車後是否會超出**容量**限制



運用非加班車進行路徑修正

Step3 : 是否會超出**容量**限制

未超出容量限制

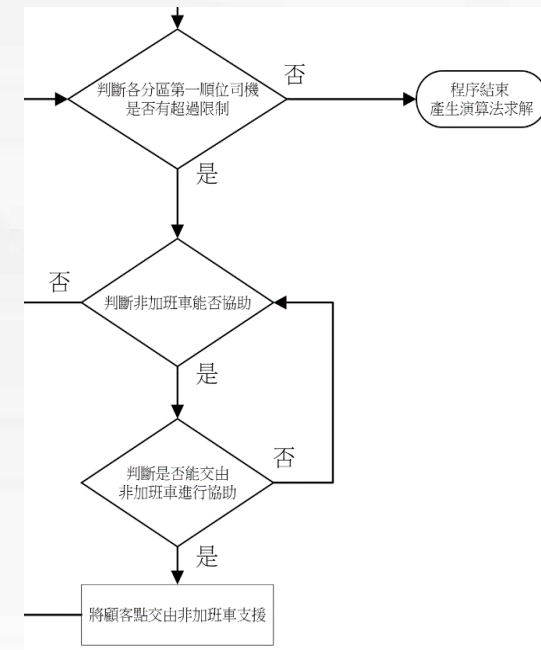


Step4 : 尋找**最佳插入位置**後，
嘗試加入並檢查**時間**限制

超出容量限制



測試其他顧客點，直到
沒有顧客點能交給支援
車，回到Step2



運用非加班車進行路徑修正

Step4 : ~~是否會超出時間限制~~



未超出時間限制

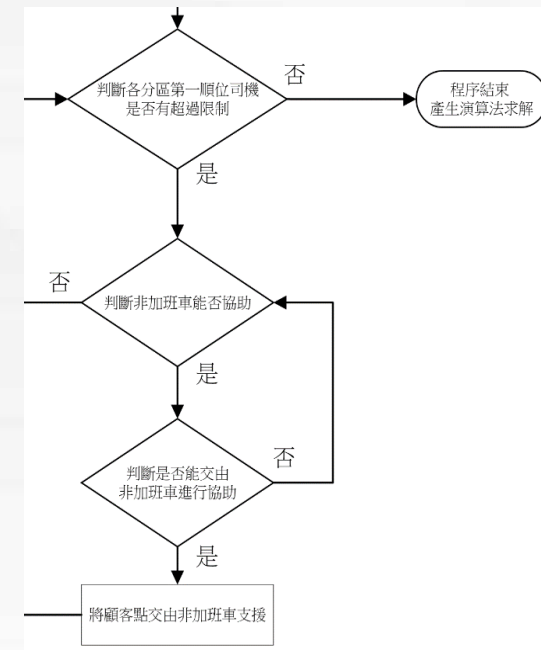


Step5 : 將被協助車實際路徑內的顧客點移出，加到支援車實際路徑內。

超出時間限制



回到Step3
測試其他顧客點



運用非加班車進行路徑修正

為達到「銜接」的效果，建議這裡再將
Step 5列出

Step6 : **檢查**被協助車是否
仍超過容量和時間限制

未超出容量限制

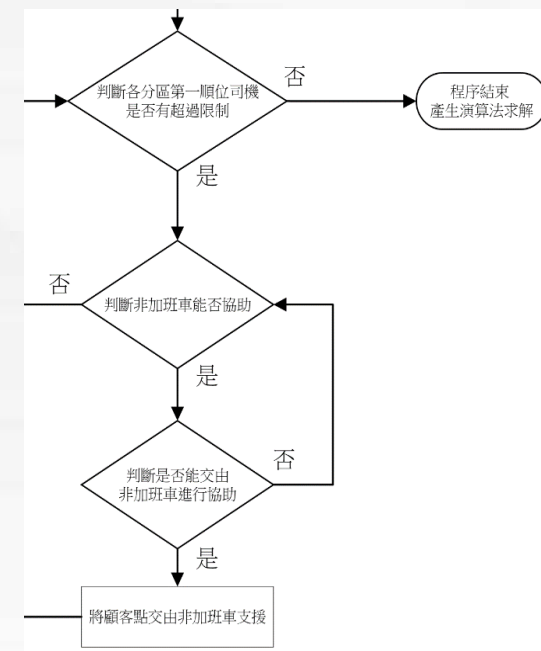


Step7 : 回到Step1，直到所有
車輛**都沒有**超過限制後，
生成**可行解**

超出容量限制

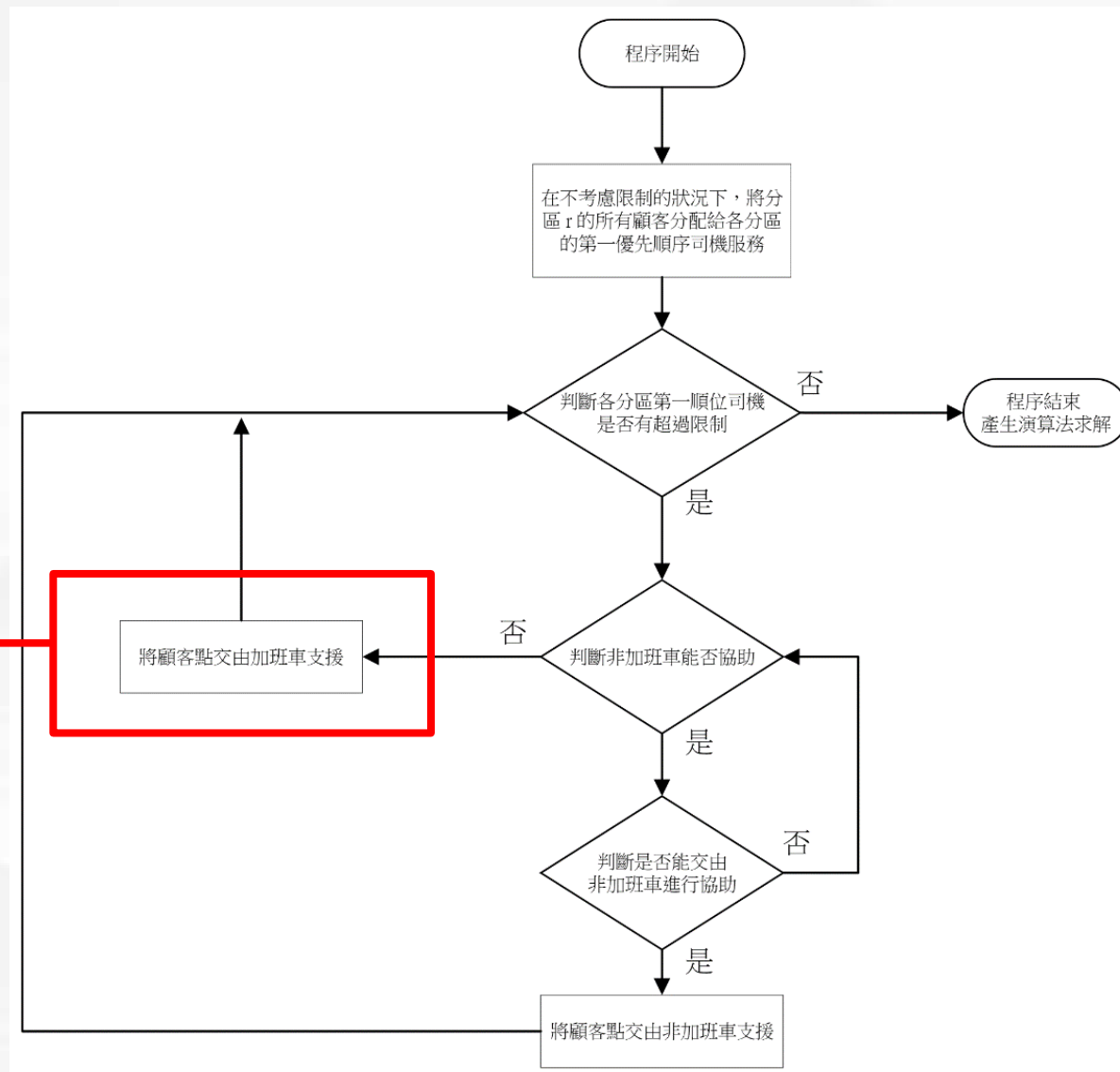


回到Step3
測試其他顧客點



演算法流程介紹

第三部分 運用加班車修正路徑



運用加班車修正路徑

將顧客點交由加班車支援

第三部分為**呼叫程序**

若分區優先順序內**沒有非加班車**能支援，則呼叫此程序

會將被協助車路徑內**需求量最大**的顧客點交由加班車

運用加班車修正路徑

將顧客點交由加班車支援

Step3(加)：建立**測試用路徑**，並先測試將路徑內**需求量最大的**顧客點交由支援車後是否會超出**容量限制**

未超出容量限制



Step4：嘗試加入測試用路徑
尾端並檢查**時間**限制

超出容量限制



測試其他顧客點

運用加班車修正路徑

將顧客點交由加班車支援

Step4 : ~~是否會超出~~時間限制

未超出時間限制



Step5 : 將被協助車實際路徑內的顧客點移出，加到支援車實際路徑內。

超出時間限制



回到Step3(加)
測試其他顧客點

運用加班車修正路徑

為達到「銜接」的效果，建議這裡再將Step 5列出

Step6 : **檢查**被協助車是否
仍超過容量和時間限制

未超出容量限制



Step7 : 回到Step1，直到所有
車輛**都沒有**超過限制後，
生成**可行解**

超出容量限制



回到Step3(加)
測試其他顧客點

將顧客點交由加班車支援



3.

結果分析

環境設置

- 運送時間限制：08:00~12:00、13:00~17:00
- 分區按照顧客點地址的行政區劃分
- 分區=司機數=7
- 每分區的優先順序個數=4
- $\alpha=1000000$
- $\beta=0/100/1000/10000/100000$

分區	(1)	(2)	(3)	(4)
1	D1	D2	D3	D5
2	D2	D1	D5	D3
3	D3	D4	D5	D2
4	D4	D3	D6	D5
5	D5	D3	D6	D2
6	D6	D4	D5	D3
7	D7	D5	D3	D4

1	東勢、豐原、后里
2	大雅、神岡、清水、沙鹿、大甲、潭子
3	北屯區、北區、東區
4	大里、霧峰、太平
5	西屯、西區、南區
6	烏日、南屯、龍井
7	中區

小型題目數據

上午

分區	客戶點 數量	客戶點
1	0	無客戶
2	1	22
3	2	13, 16
4	5	8, 9, 10, 11, 12
5	10	1, 2, 3, 7, 15, 17, 18, 19, 20, 21
6	3	4, 5, 6
7	1	14
總計	22	

下午

分區	客戶點 數量	客戶點
1	1	27
2	3	24, 25, 26
3	4	4, 5, 6, 22
4	11	7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17
5	4	1, 2, 3, 23
6	0	無客戶
7	4	18, 21 19, 20
總計	27	

大型題目數據

上午

分區	客戶點 數量	客戶點
1	7	1, 2, 3, 4, 5, 6, 7
2	6	8, 9, 10, 11, 12, 13
3	6	14, 15, 16, 17, 18, 19
4	6	20, 21, 22, 23, 24, 25
5	7	26, 27, 28, 29, 30, 31, 32
6	8	33, 34, 35, 36, 37, 38, 39, 40
7	4	41, 42, 43, 44
總計	44	

下午

分區	客戶點 數量	客戶點
1	9	45, 46, 47, 48, 49, 50, 51, 52, 53
2	8	37, 38, 39, 40, 41, 42, 43, 44
3	8	29, 30, 31, 32, 33, 34, 35, 36
4	8	21, 22, 23, 24, 25, 26, 27, 28
5	7	14, 15, 16, 17, 18, 19, 20
6	7	7, 8, 9, 10, 11, 12, 13
7	6	1, 2, 3, 4, 5, 6
總計	53	

➤ Gurobi 求解虛擬題目的上下午目標值與執行時間

	上午	下午
總旅行成本	2,182.74	2,271.58
總加班成本	928.58	627.754
總營運成本	3,111.32	2,899.33
優先順序懲罰值	300,200	100,100
第一優先順序獎勵值	39,000,000	50,000,000
目標值	-38,696,700	-49,897,000
GAP	5.43%	4.01%
執行時間(秒)	3,600	3,600

演算法求解虛擬題目的上下午目標值與執行時間

	上午	下午
總旅行成本	2,124.8	2417.78
總加班成本	1,391.52	629.33
總營運成本	3,516.32	3,047.11
優先順序懲罰值	400,100	100,100
第一優先順序獎勵值	36,000,000	49,000,000
目標值	-35,596,400	-48,896,900
GAP	12.75%	5.78%
執行時間(秒)	1.073	0.607

求解小型實際題目的GAP與執行時間比較

	Gurobi程式		求解演算法	
上/下午	GAP	執行時間(秒)	GAP	執行時間(秒)
上午	0.00%	1.52	5.29%	0.297
下午	0.00%	3.59	8.00%	0.364
平均	0.00%	2.56	6.65%	0.330

求解大型虛擬題目的GAP與執行時間比較

	Gurobi程式		求解演算法	
上/下午	GAP	執行時間(秒)	GAP	執行時間(秒)
上午	5.43%	3,600	12.75%	1.073
下午	4.01%	3,600	5.78%	0.607
平均	4.72%	3,600	9.27%	0.840

求解小型實際題目的成本、懲罰值與獎勵值比較

	上午		下午	
	Gurobi	求解演算法	Gurobi	求解演算法
總旅行成本	1,428.97	1,074.49	1,612.50	1,482.93
總加班成本	772.10	390.048	0.00	0.00
總優先順序懲罰值	100,000	100,000	1,100	1,100
第一優先順序獎勵值	19,000,000	18,000,000	25,000,000	23,000,000



求解大型虛擬題目的成本、懲罰值與獎勵值比較

	上午		下午	
	Gurobi	求解演算法	Gurobi	求解演算法
總旅行成本	2,182.74	2,124.8	2,271.58	2417.78
總加班成本	928.58	1,391.52	627.754	629.33
總優先順序懲罰值	300,200	400,100	100,100	100,100
第一優先順序獎勵值	39,000,000	36,000,000	50,000,000	49,000,000



透過Gurobi程式得到的最佳路徑

車輛	服務客戶之路徑	非第一優先 順序之客戶數
車1	0→27→0	0
車2	0→25→24→0	0
車3	0→5→4→22→0 	0
車4	0→13→15→14→11→10→7→ 6 →9→8→12→0	1
車5	0→1→3→2→23→0	0
車6	0→ 16 → 17 →0	2
車7	0→19→20→21→18→0	0
車8(加)	0→ 26 →0	1
車9(加)	無服務	0
總和		4

分區	客戶點
1	27
2	24, 25, 26
3	4, 5, 6, 22
4	7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17
5	1, 2, 3, 23
6	無客戶
7	18, 21 19, 20

透過求解演算法可能得到的路徑

車輛	服務客戶之路徑	非第一優先 順序之客戶數
車1	0→27→0	0
車2	0→26→0	0
車3	0→10→6→5→22→0	1
車4	0→13→11→15→9→16→12→8→14→7→0	0
車5	0→3→2→23→1→24→25→0 	2
車6	0→17→0	1
車7	0→21→18→20→19→0	0
車8(加)	0→4→0	1
車9(加)	無服務	0
總和		5

分區	客戶點
1	27
2	24, 25, 26
3	4, 5, 6, 22
4	7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17
5	1, 2, 3, 23
6	無客戶
7	18, 21 19, 20



4.

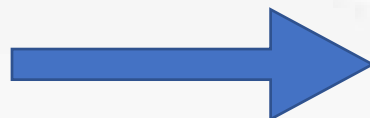
結論與建議



研究建議



缺乏真實公司資料



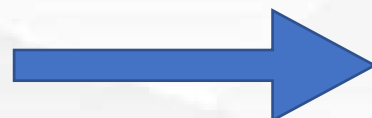
模擬可能與現實不符

演算法還能改進



1. 減少非第一優先順序的客戶數
2. 使用萬用啟發式演算法優化車輛路徑和客戶的組合

上午不能使用加班車



需同時考慮上下午，並將上午由加班車服務的客戶移至下午



5.

參考文獻

參考文獻

Hà, M. H., Phuong, H.N., Nhat, H.T.N., . (2022). "Solving the clustered traveling salesman problem with -relaxed priority rule." International Transactions in Operational Research 29(2): 837-853.

Nucamendi-Guillén, S., Dias, D.F, Olivares-Benitez, E. , Mendoza, A. . (2020). "A Memetic Algorithm for the Cumulative Capacitated Vehicle Routing Problem Including Priority Indexes." Applied Sciences 10(11): 3943.

Panchamgam, K., Xiong, Y., Golden, B., Dussault, B., Wasil, E., .(2013). "The hierarchical traveling salesman problem." Optimization Letters 7(7): 1517-1524.

Ulmer, M.,Nowak, M., Mattfeld, D., Kaminski, B., . (2020). "Binary driver-customer familiarity in service routing." European Journal of Operational Research 286(2): 477-493.


Yang, Z. ,Emmerich, M., Back, T., . (2015). "Ant based solver for dynamic vehicle routing problem with time windows and multiple priorities." IEEE Congress on Evolutionary Computation (CEC), Sendai, JAPAN.

Zhong, H., Hall, R., Dessouky, M.M., . (2007). "Territory planning and vehicle dispatching with driver learning." Transportation Science 41(1): 74-89.

黃品臻、陳沛林， “考量配送司機優先順序的車輛途程問題之研究” ，國立陽明交通大學，學士畢業專題論文，民國111年6月。



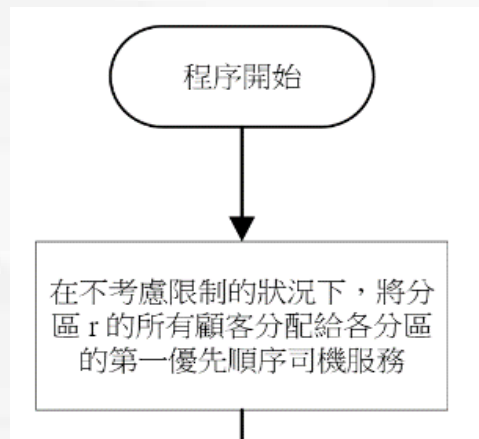
6.



Q&A補充用 資料

完整虛擬碼

▶▶ 初始路徑生成之虛擬碼



1. Input:

路網節點之集合 N 、路網邊之集合 A 、各分區區域代號之集合 R 、所有非加班車與加班車之集合 K 、車輛司機優先順序的集合 P 、位在第 r 區的客戶點集合 G_r 、服務第 r 區的第 p 順位司機參數 d_p^r 、從點 i 到點 j 的旅行時間 τ_{ij} ， $\forall (i, j) \in A$ 、客戶 i 訂購貨品的總材積 q_i ， $\forall i \in N$ 、任一司機在客戶 i 所需停留的服務時間 θ_i ， $\forall i \in N$ 、車輛 k 的總載貨量 l_k 、車輛 k 的總時間 s_k 、車輛 k 之實際路徑 η_k 、第一優先順序車輛路徑集合 γ_r ;

初始路徑生成之虛擬碼

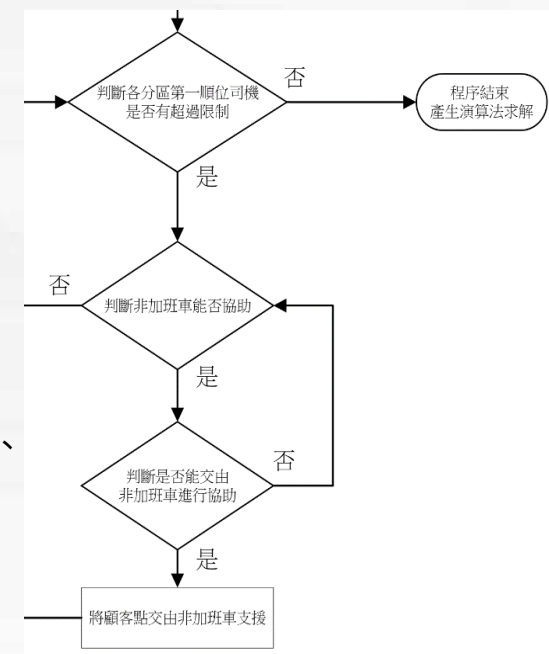
4. $r = 1$
5. Repeat
6. $k = d_1^r$;
7. 將離場站(分區0)在分區 r 內最近的第一個顧客點 x 分配給第一優先順序的車輛路徑集合 γ_r ;
8. $x = \min\{\text{distance}(x_0, N_0)\}, x_0 \text{ in } \gamma_r$;
9. 將離分區 r 質點最近的顧客 x 從集合 γ_r 中取出並加入到車輛 k 路徑 η_k 的尾端;
10. 將顧客點 x 的需求量 q_x 加到 l_k ;
11. 將場站到顧客點 x 的旅行 τ_{0x} 和服務時間 θ_x 加到 s_k ($s_k = \tau_{0x} + \theta_x$);
12. $t = 1$;
13. Repeat
14. 將分區 r 內離第一個顧客點 x 最近的顧客點 y 從車輛路徑集合 γ_r 加入到 η_k 的尾端;
15. 將顧客點 y 的需求量 q_y 加到 l_k ;
16. 將顧客點 x 到顧客點 y 的旅行 τ_{xy} 和服務時間 θ_y 加到 s_k ($s_k = s_k + \tau_{xy} + \theta_y$);
17. $t = t + 1, x = y$;
18. Until $t > G_r$;
19. $r = r + 1$;
20. Until $r > R$;

程序開始

在不考慮限制的狀況下，將分區 r 的所有顧客分配給各分區的第一優先順序司機服務

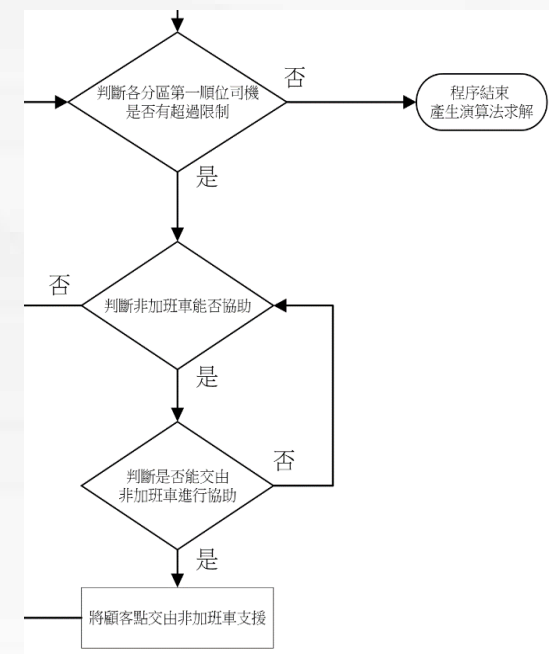
非加班車路徑修正虛擬碼

1. Input : 路網節點之集合 N 、路網邊之集合 A 、各分區區域代號之集合 R 、所有非加班車與加班車之集合 K 、車輛司機優先順序的集合 P 、加班車輛順位之集合 P^+ 、服務第 r 區的第 p 順位司機參數 d_p^r 、每部貨車可容納最大材積 c 、車輛可允許容積的最大使用率 μ 、從點 i 到點 j 的旅行時間 τ_{ij} 、 $\forall (i,j) \in A$ 、客戶 i 訂購貨品的總材積 q_i 、 $\forall i \in N$ 、任一司機在客戶 i 所需停留的服務時間 θ_i 、 $\forall i \in N$ 、非加班車最大工時 t 、車輛 k 的總載貨量 l_k 、車輛 k 的總時間 s_k 、車輛 k 之實際路徑 η_k 、車輛 k 之測試用路徑 H_k ;
2. //從第一分區開始檢查
3. $r=1$;
4. Repeat
5. //若車輛超過限制
6. if($l_k > c * \mu$ or $s_k > t$)
7. $p=2$;
8. Repeat
9. 選取分區 r 中離第 p 優先順序司機車輛 k' 。 $k'=d_p^r$;
10. if($l_{k'} > c * \mu$ or $s_{k'} > t$)
11. $p=p+1$;
12. goto step 49;//跳到第49行繼續選取其他順位的車輛的迴圈



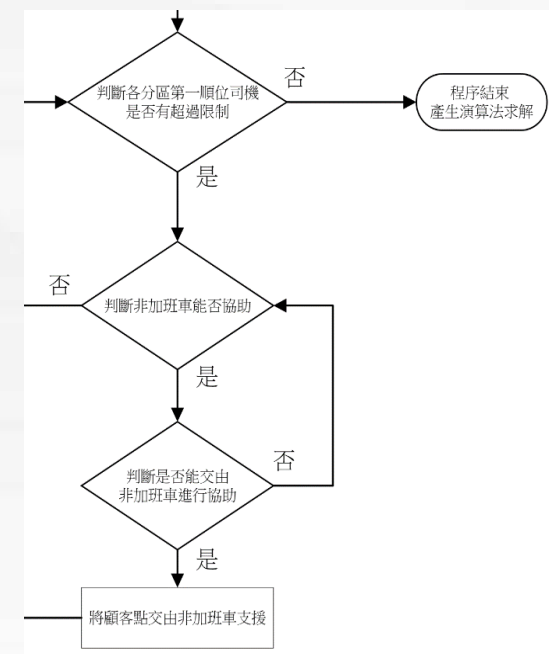
非加班車路徑修正虛擬碼

```
13. Else(  
14.    $\varepsilon=1$ ;  
15.   Repeat  
16.     車輛 k 路徑  $\eta_k$  中選取離分區  $k'$  質心第  $\varepsilon$  近的顧客點 x;  
17.      $H_{k'}$ (測試用路徑) =  $\eta_{k'}$ ;  
18.     將顧客點需求量  $q_x$  加入測試用路徑  $H_{k'}$  內並計算  $H_{k'}$  路徑內的  $l_{k'}$ ;  $l_{k'} = \sum q_i \ i \in H_{k'}$ ;  
19.     if( $l_{k'} > c * \mu$ )  
20.       goto step 46; // 跳到第46行繼續選取其他顧客點的迴圈  
21.     End if  
22.     // 開始尋找最適合插入的位置  
23.     t=1;  $y_{best}=M$ ;  
24.     Repeat  
25.       選取嘗試路徑  $H_{k'}$  中顧客點;  
26.        $y=v_t^{k'}$  以及顧客點;  
27.        $y'=v_{t+1}^{k'}$  並計算;  
28.        $y^*=|yx|+|xy'|-|yy'|$ ;  
29.     if( $y^* < y_{best}$ )  
30.        $y_{best}=y^*$ ;  
31.     Else  $y_{best} = y_{best}$ ;
```



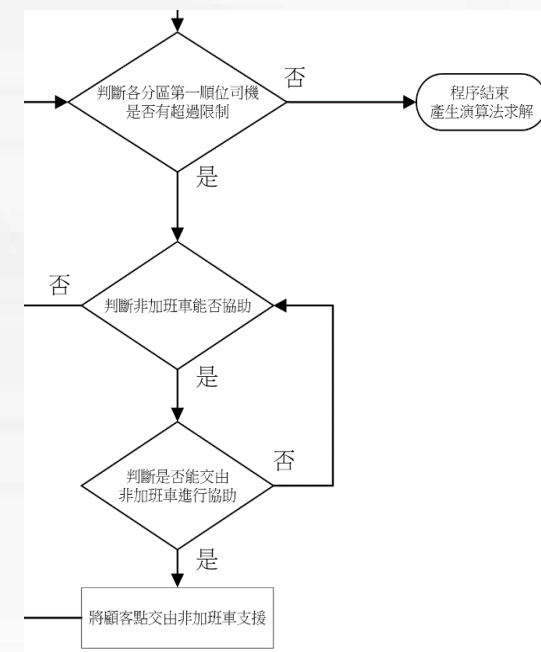
非加班車路徑修正虛擬碼

```
32.      t=t+1;
33.      Until t >  $\sum v_i^{k'} \ i \in H_{k'}$ ;
34.      將顧客點 x 加入到  $H_{k'}$  的  $y_{best}$  的位置，並計算  $H_{k'}$  路徑的總耗時 ( $s_{k'} = \sum \tau_{ij} + \sum \theta_i \ i \in H_{k'}, j \in H_{k'}$ );
35.      if(  $s_{k'} > t$  )
36.          goto step 46; // 跳到第46行繼續選取其他顧客點的迴圈
37.      Else{
38.          將顧客點 x 在  $\eta_k$  路徑中所在的第 m 個位置從  $\eta_k$  移除後連結第 m-1 個和第 m+1 個的顧客點將 x 加入到  $\eta_{k'}$  裡的  $y_{best}$  位置;
39.          將  $l_k$  減去  $q_x$ ;
40.          重新計算  $\eta_k$  路徑裡的  $s_k$ ;
41.          if(  $l_k > c * \mu$  or  $s_k > t$  )
42.               $\varepsilon = 1$ ;
43.          goto step 68; // 跳到第47行繼續選取其他顧客點的迴圈
44.          Else goto step 55; // 跳到第55行繼續下一個分區
45.      }
46.       $\varepsilon = \varepsilon + 1$ ; // 尋找其他顧客點
47.      Until  $\varepsilon \leq \sum v_i^k, i \in N$ ; // 直到確認完所有顧客點
48.  )
```



非加班車路徑修正虛擬碼

```
49.  p=p+1;//尋找後續順位的非加班車
50.  Until  $p > P/P^+$ ;//直到後續順位的非加班車用完
51.  if( $l_k > c * \mu$  or  $s_k > t$ )
52.    加班車程序//第三PART;
53.  End if
54. End if
55.  r=r+1;//選取下個分區
56. Until  $r > R$ ;//直到確認完所有分區
57. Output: 車輛k的總載貨量 $l_k$ 、車輛k的總時間 $s_k$ 、車輛k之實際路徑 $\eta_k$ 
```



加班車路徑修正虛擬碼

1. Input: 路網節點之集合 N 、路網邊之集合 A 、各分區區域代號之集合 R 、所有非加班車與加班車之集合 K 、車輛司機優先順序的集合 P 、加班車輛順位之集合 P^+ 、服務第 r 區的第 p 順位司機參數 d_p^r 、每部貨車可容納最大材積 c 、車輛可允許容積的最大使用率 μ 、從點 i 到點 j 的旅行時間 τ_{ij} ， $\forall (i, j) \in A$ 、客戶 i 訂購貨品的總材積 q_i ， $i \in N$ 、任一司機在客戶 i 所需停留的服務時間 θ_i ， $\forall i \in N$ 、非加班車最大工時 t 、車輛 k 的總載貨量 l_k 、車輛 k 的總時間 s_k 、車輛 k 之實際路徑 η_k 、車輛 o 之測試用路徑 H_o ;
2. //加班車程序
3. $v=1$; $o=1$;
4. Repeat
5. Repeat
6. 車輛 k 路徑 η_k 中選取需求量第 v 大的顧客點 x ;
7. H_o (測試用路徑) $= \eta_o$;
8. 將顧客點 x 加到測試用路徑 H_o 內並計算 H_o 路徑內的 l_o ， $l_o = \sum q_i ; i \in H_o$;
9. If($l_o > c * \mu$)
10. goto step 26; //跳到第26行繼續尋找其他顧客點的迴圈
11. Else{
12. 計算 H_o 路徑的總耗時 $s_o = \sum \tau_{ij} + \sum \theta_i, i \in H_o, j \in H_o$;
13. If($s_o > t$)
14. goto step 26; //跳到第26行繼續尋找其他顧客點的迴圈

加班車路徑修正虛擬碼

```

15. Else{
16.   將顧客點  $x$  在  $\eta_k$  路徑中所在的第  $m$  個位置從  $\eta_k$  移除後連結第  $m-1$  個和第  $m+1$  個的顧客點;
17.   將  $x$  加入到  $\eta_k$  的尾端;
18.   將  $l_k$  減去  $q_x$ ;
19.   重新計算  $\eta_k$  路徑裡的  $s_k$ ;
20.   If( $l_k > c * \mu$  or  $s_k > t$ )
21.      $v=1$ ;
22.     goto step 27; //跳到第27行繼續尋找其他顧客點的迴圈
23.   End if
24. }
25. }
26.  $v=v+1$ ; //尋找其他顧客點
27. Until  $v \leq \sum v_i^k, i \in N$  //直到確認完所有顧客點
28.  $o=o+1$ ; //尋找其他加班車
29. Until  $o > P^+$ ; 直到所有加班車用完
30. Output : 車輛  $k$  的總載貨量  $l_k$ 、車輛  $k$  的總時間  $s_k$ 、車輛  $k$  之實際路徑  $\eta_k$ 

```