

The background features abstract green geometric shapes, including triangles and polygons, some of which are semi-transparent, creating a layered effect. A thin white line also runs diagonally across the lower right portion of the image.

車輛途程問題 (Vehicle Routing Problem; VRP)

Solution encoding/decoding

- Encoding continuous decision variables

3.6	7.2	4.9	1.3	2.9
-----	-----	-----	-----	-----

- Encoding discrete decision variables

1	0	0	1	1
---	---	---	---	---

3	7	4	3	2
---	---	---	---	---

- Encoding permutation solutions

3	1	5	4	2
---	---	---	---	---

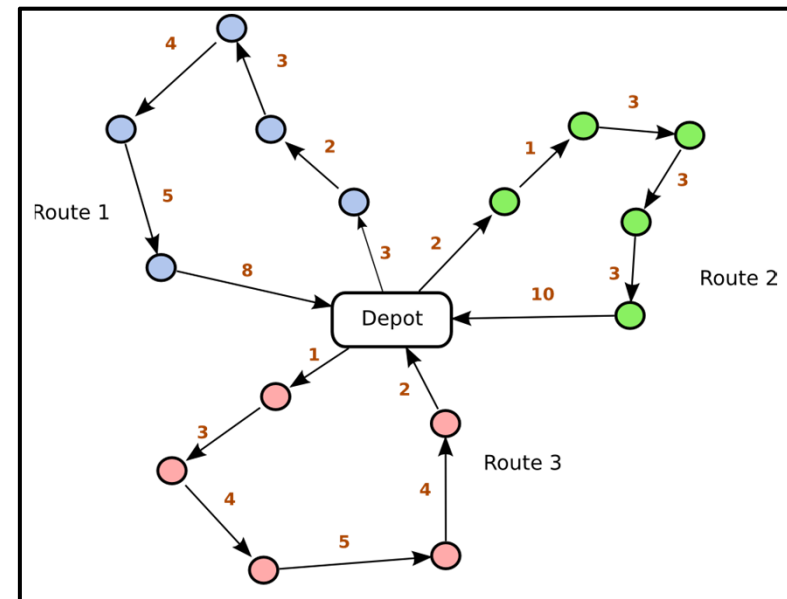
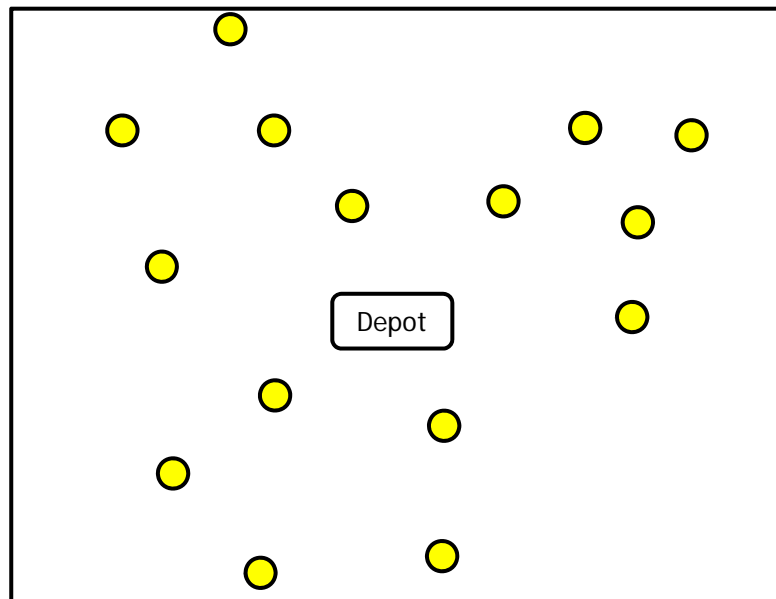
- Mixed encoding

3	1	5	4	2	1	0	0	1	1
---	---	---	---	---	---	---	---	---	---

車輛途程問題(Vehicle Routing Problem, VRP)

● 車輛途程問題

- 使一個車隊組織車輛運輸路線
滿足顧客需求，且不超過車子的容量限制
以達到最低成本、最短路徑等目的。
- 被認為是一個NP-Hard的組合優化問題。



旅行銷售員問題(Traveling Salesman Problem, TSP)

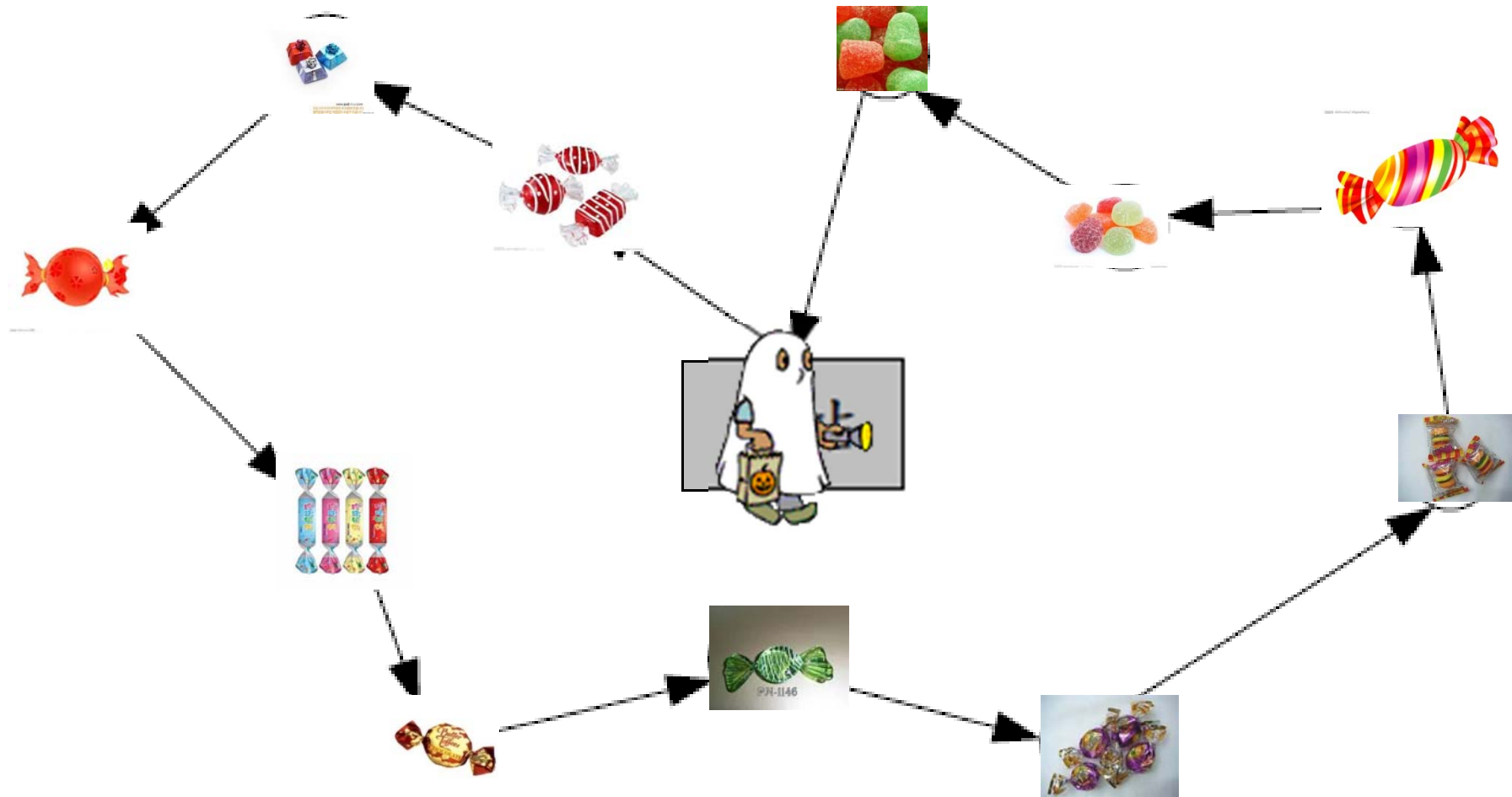
- TSP

- 為車輛以最小運送成本(旅行距離或旅行時間)
由場站出發，服務完所有需求點後回到場站，
路線規劃時不考慮車容量限制，
為單一VRP問題。

- VRP為TSP之延伸，

主要差異為VRP車輛有載量限制且為多車輛問題。

TSP



VRP

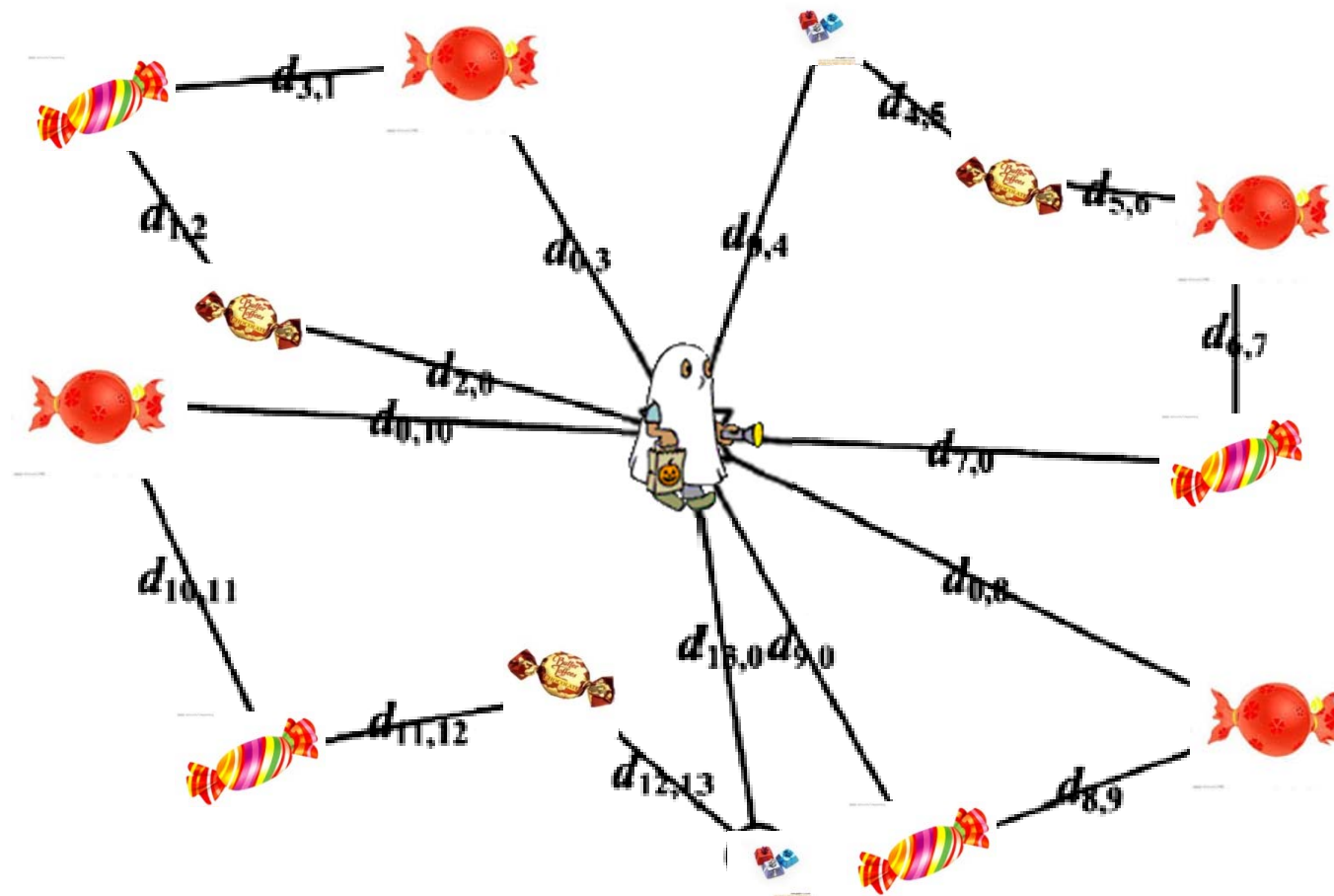


Fig. 1. An example of the VRP.

(Capacitated) Vehicle Routing Problem-限制

目標：

Minimized 路徑長度(成本)

限制：

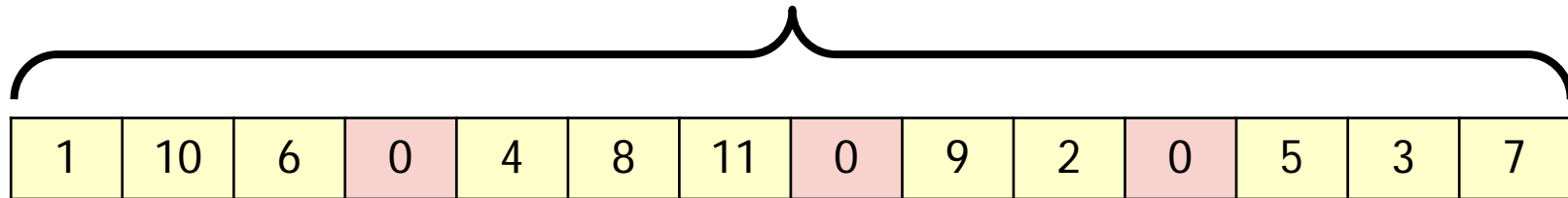
1. 每條路線的起點與終點都在倉庫
2. 每個顧客只能被一輛車經過一次
3. 每輛車最多走一條路線
4. 每個路線的負載不得超過車輛載重 Q
5. 每個路線的顧客服務時間與運送時間的總和不得超過車輛總服務時間 D

文獻探討

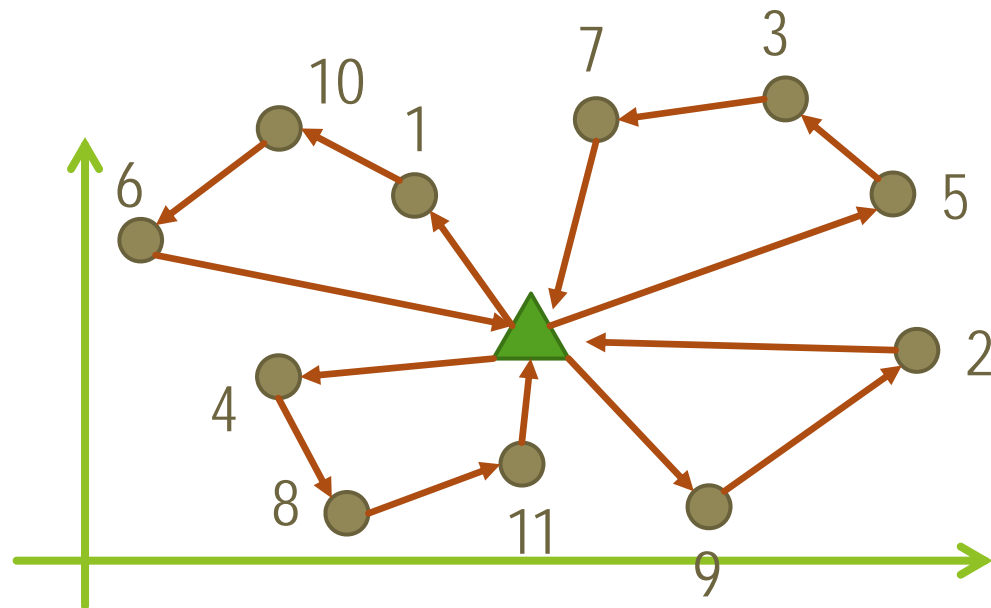
1. CVRP是NP-Hard問題
(Haimovich et al., 1988)
2. 基因演算法
(Backer & Ayechev, 2003; Berger & Barkaoui, 2003)
3. 螞蟻演算法
(Bullnheimer et al., 1999; Doerner et al., 2002)
4. 粒子群演算法
(Chen et al., 2006; Ai & Kachitvichyanukul, 2007)

第一種編碼設計

11個顧客的排列 + 3個0 (分4輛車)

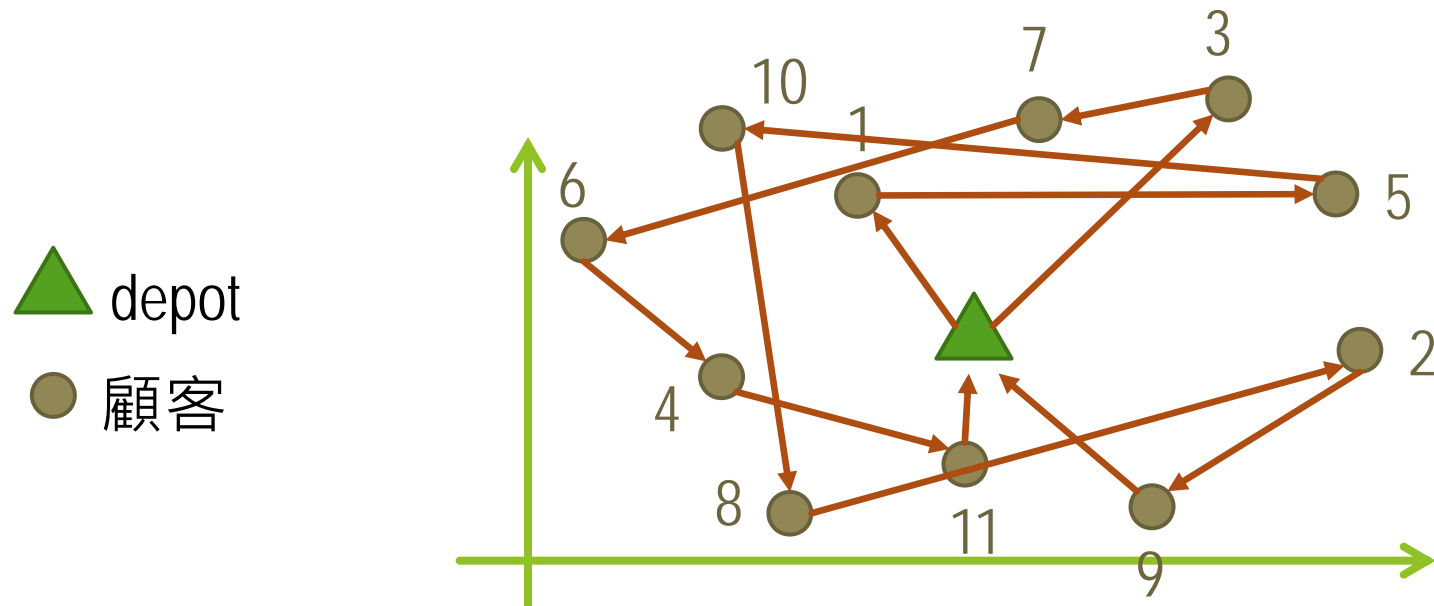
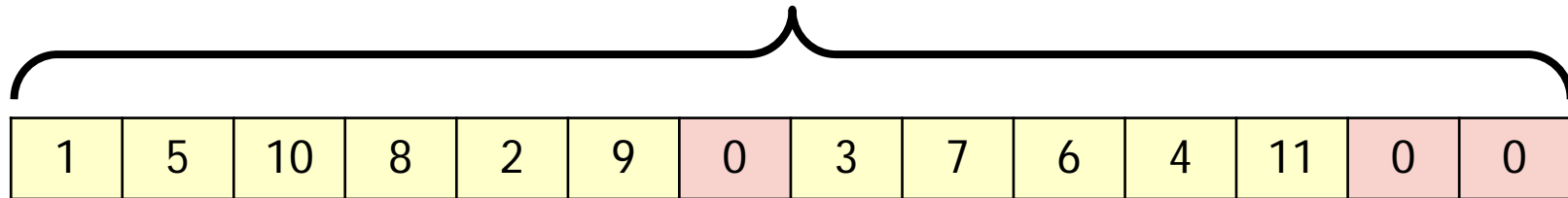


▲ depot
● 顧客



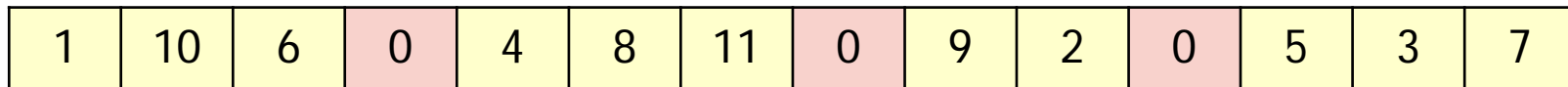
然而，排列不好會造成不好的解...

11個顧客的排列 + 3個0 → 只分2輛車，且繞路到圖的兩端



Python code for the 1st encoding

11個顧客的排列 + 3個0 (分4輛車)



1	10	6	0	4	8	11	0	9	2	0	5	3	7
---	----	---	---	---	---	----	---	---	---	---	---	---	---

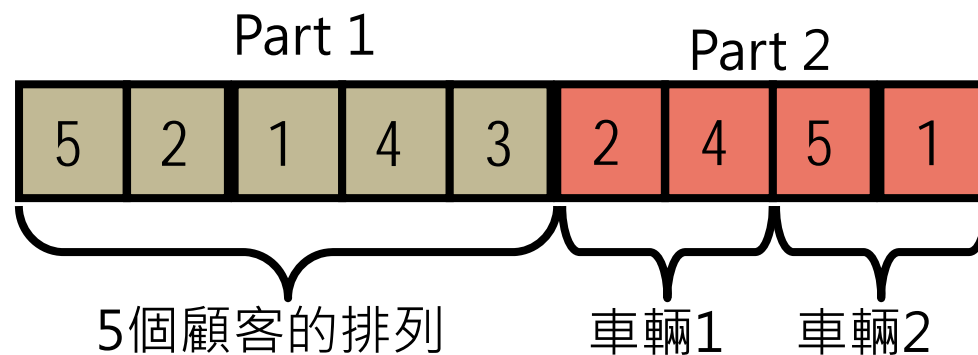
```
55 def initPop():                                # 初始化群體
56     p = []
57
58     for i in range(NUM_CHROME) :
59         # ==== Step 2-1. 產生1~11的隨機排列 ====
60         a = list(np.random.permutation(range(1, NUM_CITY+1)))
61
62         # ==== Step 2-2. 產生用三個0插入到這個1~11的隨機排列 ==
63         for j in range(NUM_VEHICLE-1):
64             a.insert(np.random.randint(len(a)+1), 0)
65         p.append(a)
66
67
68     return p
```

Solution decoding

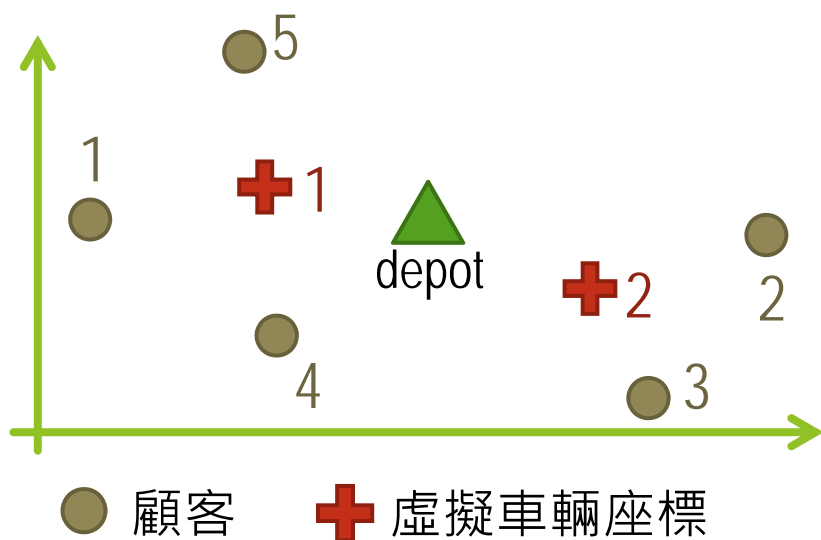
(自己練習看)

```
55 def fitFunc(x):          # 適應度函數      # ==== Step 3. 改適應度函數 ====
56     v = []              # 記所有車的路徑      [ [車1的路], [車2的路], ... ]
57     v_dist = []         # 記所有車的路徑距離 [ 車1的路長度, 車2的路長度, ... ]
58
59     route = []          # 記目前考慮的車的路 [ 經過城市1, 經過城市2, ... ]
60     dist = 0            # 記目前考慮的車的路的長度
61
62     pre_city = 0        # 車子前一個city所在位置為 depot (0)
63
64     for i in range(NUM_BIT):
65         if x[i] == 0 :   # 當 code 是 0 時(表這台車回到dept, 此時要結算這台車的路和長度)
66             v.append(route) # 把目前的路加入到 v
67             v_dist.append(dist + cost[pre_city][0]) # 把目前路長度加上回到dept的長度, 把此長度加到 v_dist
68
69             route = []     # 清空route內經過的城市
70             dist = 0       # 長度設為初始0
71             pre_city = 0   # 車子前一個city所在位置在 depot (0)
72             continue      # 跳執行 for loop 的下一迴圈
73
74             route.append(x[i]) # 把 code 所代表的城市加入到目前車的路route
75             dist += cost[pre_city][x[i]] # 把前一城市至目前 code 所代表城市的距離加到目前車的路長度 dist
76             pre_city = x[i]    # 車子前一個city所在位置變成是 code 所代表城市
77
78     if route != [] :      # 若最後一台車還沒考慮的話
79         v.append(route)   # 把目前的路加入到 v
80         v_dist.append(dist + cost[pre_city][0]) # 把目前路長度加上回到dept的長度, 把此長度加到 v_dist
81
82     return -max(v_dist)   # 因為是最小化問題
```

第二種編碼設計

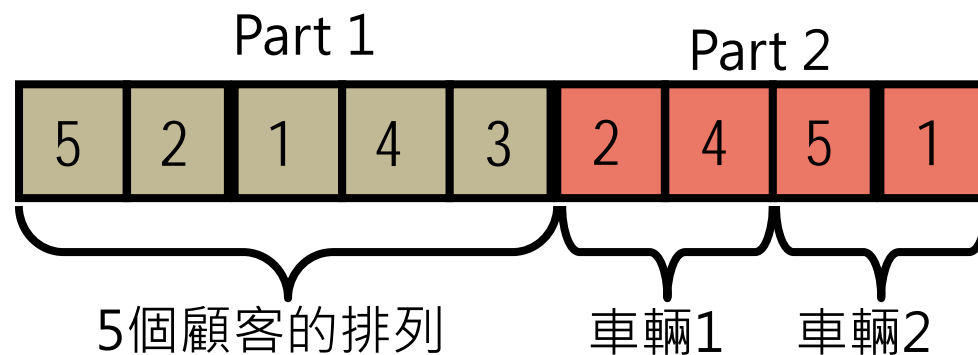


顧客	5	2	1	4	3
車輛1	近	遠	近	近	遠
車輛2	遠	近	遠	遠	近

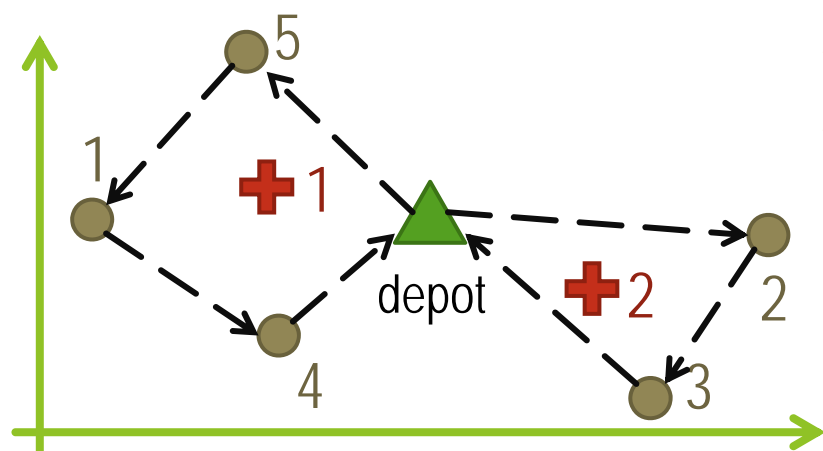


車輛1			
車輛2			

第二種編碼設計



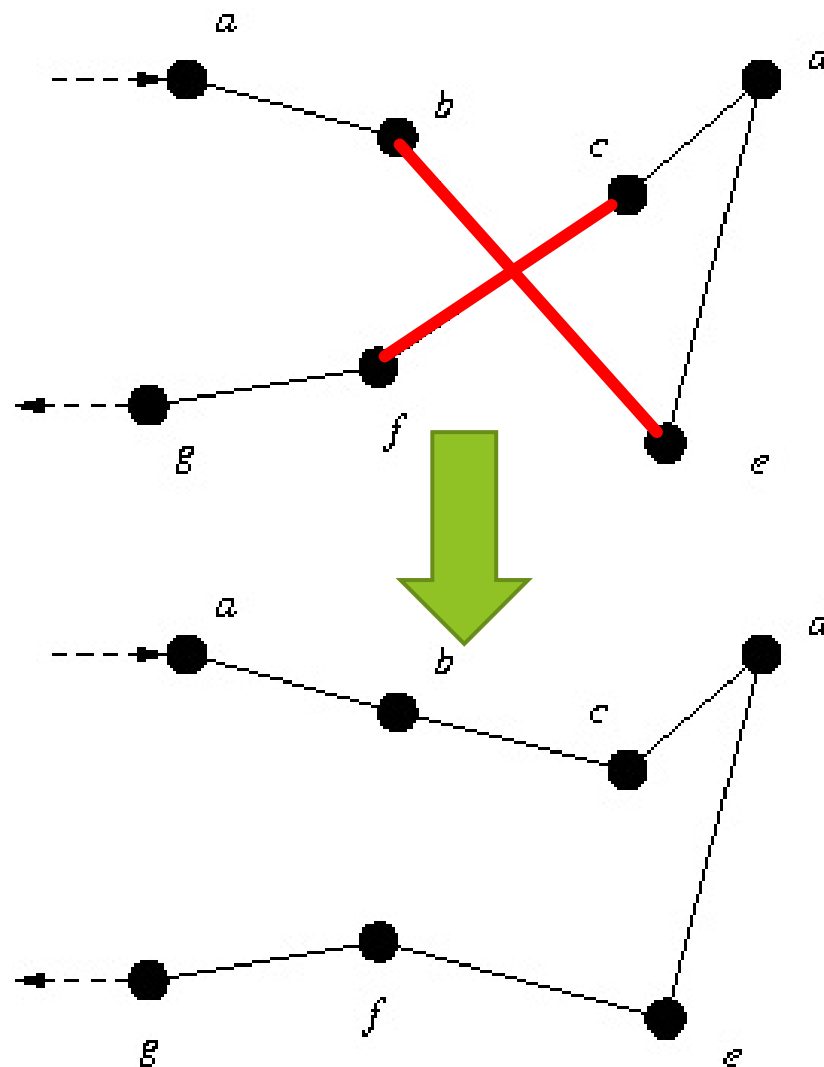
顧客	5	2	1	4	3
車輛1	近	遠	近	近	遠
車輛2	遠	近	遠	遠	近



● 顧客 + 虛擬車輛座標

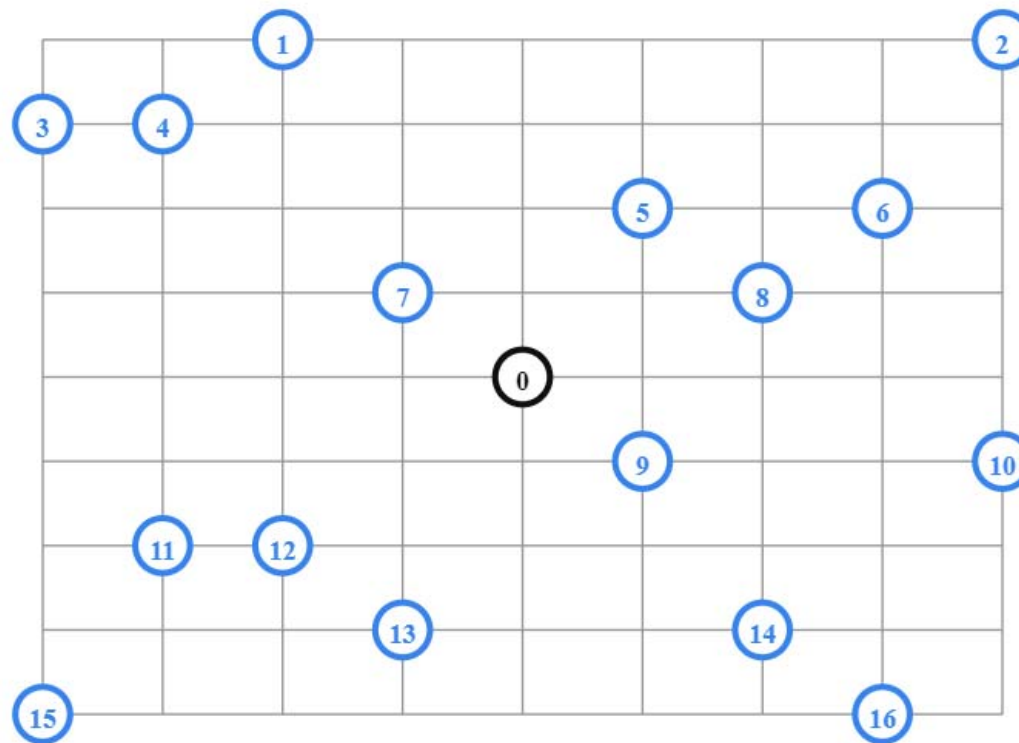
車輛1	5	1	4
車輛2	2	3	

區域搜尋：2-opt exchange



Exercise

- 考慮有一間公司有4輛車，今天要送貨給16個顧客。
- 修改“GA08-VRP-ex.py”程式 (是個不完整的此問題的程式) 找出送貨給這16個顧客的這4輛車的最佳路徑，使得最長的一條車輛路徑能越小越好。



16個顧客的(x, y)座標

- location = [
 - [456, 320], # location 0 - the depot
 - [228, 0], # location 1
 - [912, 0], # location 2
 - [0, 80], # location 3
 - [114, 80], # location 4
 - [570, 160], # location 5
 - [798, 160], # location 6
 - [342, 240], # location 7
 - [684, 240], # location 8
 - [570, 400], # location 9
 - [912, 400], # location 10
 - [114, 480], # location 11
 - [228, 480], # location 12
 - [342, 560], # location 13
 - [684, 560], # location 14
 - [0, 640], # location 15
 - [798, 640] # location 16
-]

16 個顧客地點的距離矩陣

- `dist = [`
- `[0, 548, 776, 696, 582, 274, 502, 194, 308, 194, 536, 502, 388, 354, 468, 776, 662],`
- `[548, 0, 684, 308, 194, 502, 730, 354, 696, 742, 1084, 594, 480, 674, 1016, 868, 1210],`
- `[776, 684, 0, 992, 878, 502, 274, 810, 468, 742, 400, 1278, 1164, 1130, 788, 1552, 754],`
- `[696, 308, 992, 0, 114, 650, 878, 502, 844, 890, 1232, 514, 628, 822, 1164, 560, 1358],`
- `[582, 194, 878, 114, 0, 536, 764, 388, 730, 776, 1118, 400, 514, 708, 1050, 674, 1244],`
- `[274, 502, 502, 650, 536, 0, 228, 308, 194, 240, 582, 776, 662, 628, 514, 1050, 708],`
- `[502, 730, 274, 878, 764, 228, 0, 536, 194, 468, 354, 1004, 890, 856, 514, 1278, 480],`
- `[194, 354, 810, 502, 388, 308, 536, 0, 342, 388, 730, 468, 354, 320, 662, 742, 856],`
- `[308, 696, 468, 844, 730, 194, 194, 342, 0, 274, 388, 810, 696, 662, 320, 1084, 514],`
- `[194, 742, 742, 890, 776, 240, 468, 388, 274, 0, 342, 536, 422, 388, 274, 810, 468],`
- `[536, 1084, 400, 1232, 1118, 582, 354, 730, 388, 342, 0, 878, 764, 730, 388, 1152, 354],`
- `[502, 594, 1278, 514, 400, 776, 1004, 468, 810, 536, 878, 0, 114, 308, 650, 274, 844],`
- `[388, 480, 1164, 628, 514, 662, 890, 354, 696, 422, 764, 114, 0, 194, 536, 388, 730],`
- `[354, 674, 1130, 822, 708, 628, 856, 320, 662, 388, 730, 308, 194, 0, 342, 422, 536],`
- `[468, 1016, 788, 1164, 1050, 514, 514, 662, 320, 274, 388, 650, 536, 342, 0, 764, 194],`
- `[776, 868, 1552, 560, 674, 1050, 1278, 742, 1084, 810, 1152, 274, 388, 422, 764, 0, 798],`
- `[662, 1210, 754, 1358, 1244, 708, 480, 856, 514, 468, 354, 844, 730, 536, 194, 798, 0],`
- `]`

最佳解

Route for vehicle 0:

0 -> 8 -> 6 -> 2 -> 5 -> 0

Distance of route: 1552m

Route for vehicle 1:

0 -> 7 -> 1 -> 4 -> 3 -> 0

Distance of route: 1552m

Route for vehicle 2:

0 -> 9 -> 10 -> 16 -> 14 -> 0

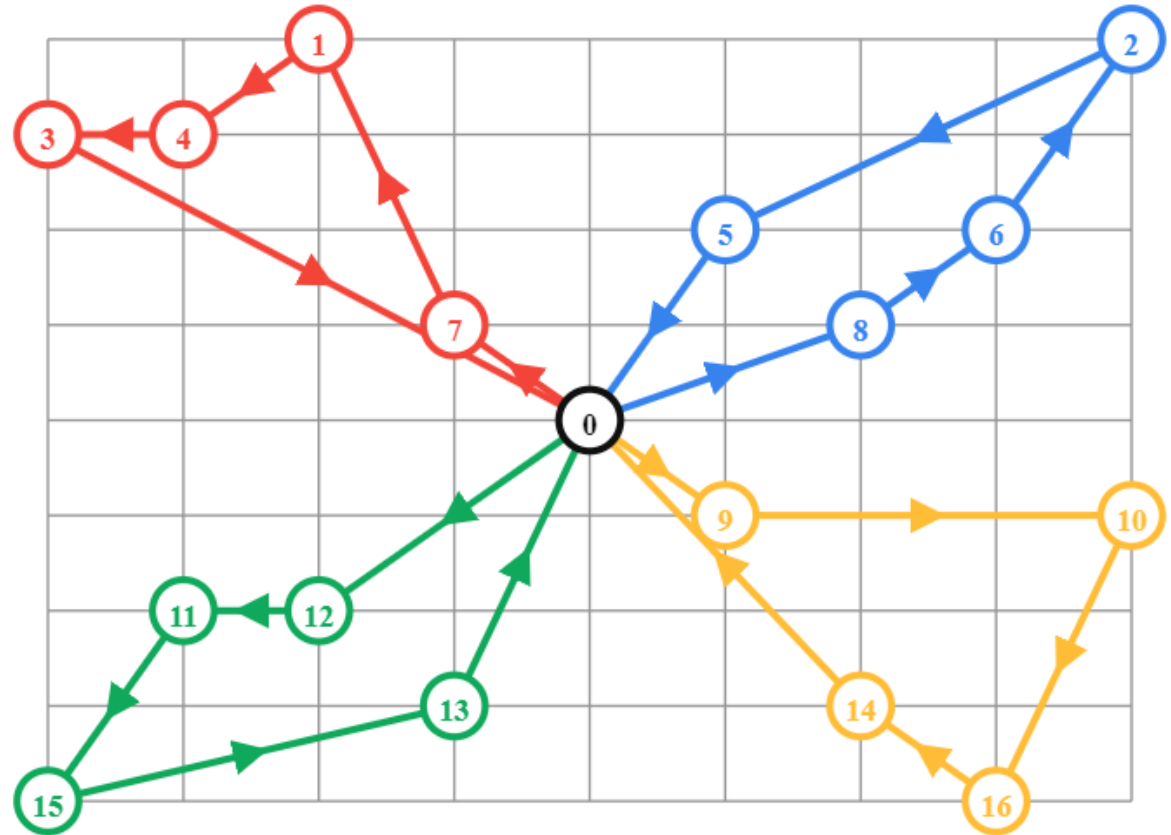
Distance of route: 1552m

Route for vehicle 3:

0 -> 12 -> 11 -> 15 -> 13 -> 0

Distance of route: 1552m

Total distance of all routes: 6208m



- Step 1. 設定參數

- 1) 設定16個顧客; 2) 設定16個顧客的位置和距離矩陣dist; 3) 設定4個車輛

- Step 2. (編碼) 設定初始解

- {1, ..., 16} 的隨機排列(Part 1) 和 4 個(x, y) 的隨機座標(Part 2) (範圍[0,900])

- Step 3. (解碼) 改適應度函數(把下面的程式碼貼到對應的位置，已貼好)

- 輸入參數：x1是{1, ..., 16} 的排列, x2是4 個車輛(x, y)座標

- `v_dist = [0, 0, 0, 0]` #用以紀錄4輛車的路徑長
`pre_city = [0, 0, 0, 0]` #用以紀錄4輛車的目前位置

```
for i in range(NUM_BIT):  
    j = np.argmin([math.hypot(location[x1[i]][0] - x2[2*k],  
location[x1[i]][1] - x2[2*k+1]) for k in range(NUM_VEHICLE)])  
    v_dist[j] += dist[pre_city[j]][x1[i]]  
    pre_city[j] = x1[i]
```

```
for i in range(NUM_VEHICLE):  
    v_dist[i] += dist[pre_city[i]][0]
```

```
return -max(v_dist) # 因為是最小化問題
```

- Step 4. 調演算法參數使得60迴圈以前就可以找出最佳解1552
(NUM_CHROME, Pc, Pm)