元智大學工業工程與管理學系畢業專題

以模擬退火演算法求解週期性旅行推銷員問題

學生：呂柏緯、王長堯　　指導教授：丁慶榮 教授

研究動機與目的

由於自動販賣機在校園與商場中日益普及，我們希望能透過週期性的補貨模式，找出更有效率的多日配送方式。透過觀察實際販賣機的補貨週期，我們蒐集並整理相關資料，並參考過去研究以探討現行方法的不足。預期的成果為建立能有效規劃週期性補貨的多日路線模型，並以模擬退火演算法進行求解，使系統能在不同週期需求下提供最佳化的補貨路線。

研究流程

問題定義與文獻回顧 ➡ 模型建構 ➡ 演算法設計與測試 ➡ 案例分析與結果驗證

問題定義

1. 決定每位顧客的拜訪頻率型態（例如每日、隔日、每週一次等）。
2. 決定在各週期內拜訪的順序與路線規劃。
3. 使整體總行程距離或成本最小化，同時滿足拜訪頻率限制。
4. 將週期性拜訪需求轉換為可編碼的解空間。
5. 結合模擬退火演算法進行全域搜尋，以求得近似最優的週期性路線。

研究方法

● 問題假設

1. 單車輛服務。
2. 所有顧客位置與拜訪頻率已知。
3. 每次路線皆從配送中心出發並返回。
4. 每位顧客於週期內須被拜訪指定次數。
5. 配送路線完整，已知旅行時間。

● 集合

H：顧客可能的拜訪頻率集合

N：客戶集合

T：規劃期內的天集合

V：點集合（0為倉庫），$V = N \cap \{0\}$

● 數學模型

$$Min = \sum_{t\in T}\sum_{i\in V}\sum_{j\in V} c_{ij}x_{ijt}$$

s.t.

$$\sum_{j\in N} x_{0jt} = 1 \qquad \forall t \in T$$

$$\sum_{j\in N} x_{j0t} = 1 \qquad \forall t \in T$$

$$\sum_{j\in V} x_{ijt} = y_{it} \qquad \forall i \in N, t \in T$$

$$\sum_{j\in V} x_{jit} = y_{it} \qquad \forall i \in N, t \in T$$

$$\sum_{t\in T} y_{it} = f_i \qquad \forall i \in N$$

$$u_{it} - u_{jt} + |N|\, x_{ijt} \le |N| - 1 \quad \forall i,j \in V,\ t \in T$$

$$x_{ijt} \in \{0,1\}$$

$$y_{it} \in \{0,1\}$$

● 參數

c_{ij}：節線(i,j)的運輸成本

f_i：客戶 i 在整個規劃期內必須被服務的次數

● 決策變數

x_{ijt} { 1 如果在第 t 天使用節線(i,j) ; 0 其他 }

y_{it} { 1 如果在第 t 天拜訪節點 i ; 0 其他 }

u_{it}：節點 i 在第 t 天拜訪順序

解編碼方式

● 解編碼方式

採用下方之編碼方式，將 4 天週期的拜訪需求以 7 種模式表示：1 天（每日）、隔日（1,3）、隔日（2,4）、第 1 天、第 2 天、第 3 天、第 4 天。

依照圖解碼可得：

1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	1	1	2	2	3	2	3	5	7	4	6	5

指派

第一天:{1,2,4,5,6,8,12}
第二天:{1,2,3,4,7,9,10,14}
第三天:{1,2,3,4,5,6,8,13}
第四天:{1,2,3,4,7,9,11}

途程

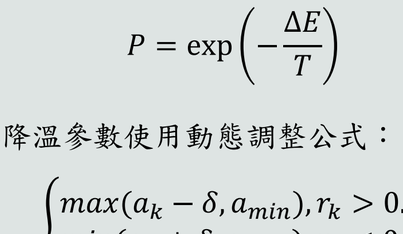
0 → 8 → 12 → 2 → 5 → 4 → 3 → 1 → 6 → 0
0 → 14 → 2 → 7 → 10 → 9 → 4 → 3 → 1 → 0
0 → 8 → 2 → 5 → 4 → 13 → 3 → 1 → 6 → 0
0 → 2 → 7 → 9 → 4 → 3 → 1 → 11 → 0

說明如下：

客戶 1–4：每日拜訪
客戶 5、6、8：{1,3} 拜訪
客戶 7、9：{2,4} 拜訪
客戶 12：第 1 天拜訪
客戶 10：第 2 天拜訪
客戶 13、11：第 3、4 天拜訪
依此形成每日路線，如上方所示：

模擬退火演算法

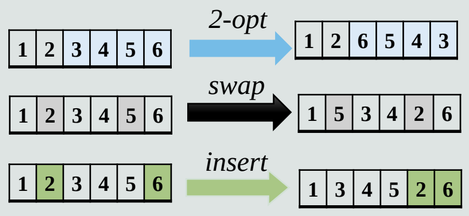
模擬退火以隨機搜尋方式在多峰函數中逐步下降並收斂至較佳解之示意圖：



演算法架構

1.初始解生成
2.鄰域解產生
3.解的評估
4.是否更佳／機率接受
5.更新至目前最佳解
6.降溫與重置
7.終止條件判斷

其中透過三種手法，改善新解：

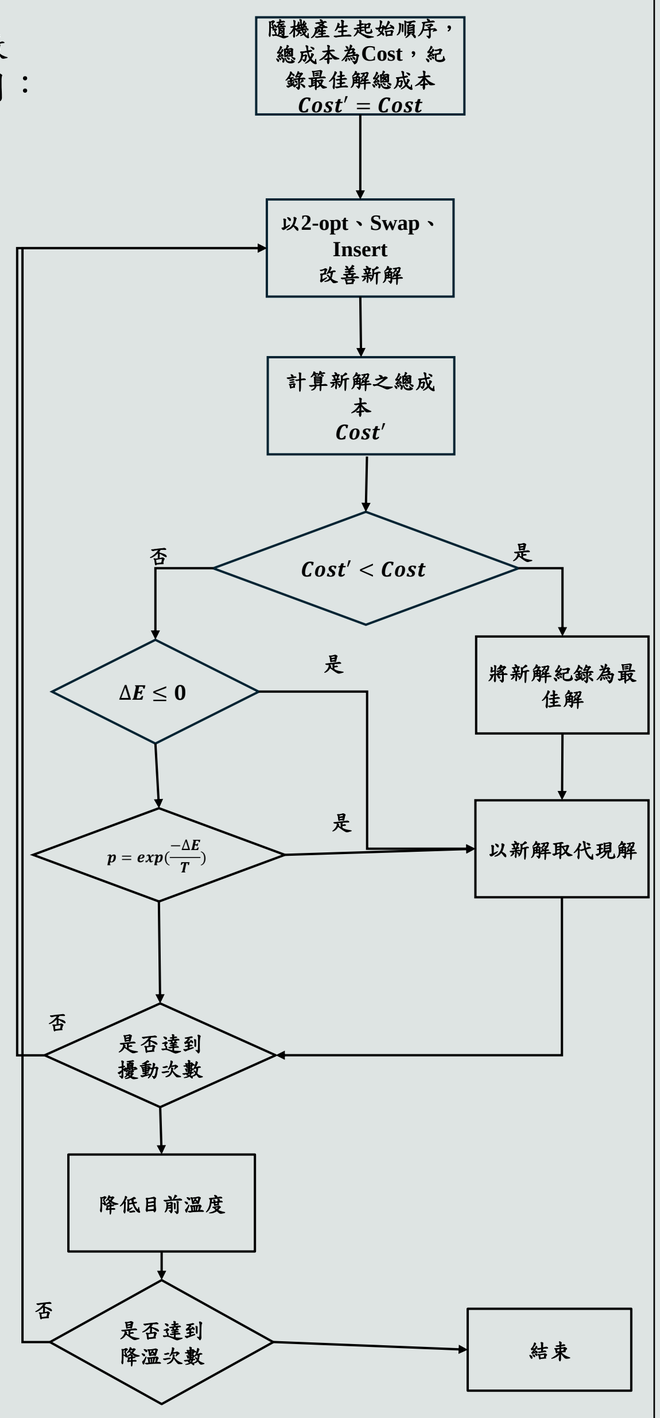


且接受機率為波茲曼函數：

$$P = \exp\left(-\frac{\Delta E}{T}\right)$$

其降溫參數使用動態調整公式：

$$a_k = \begin{cases} max(a_k - \delta, a_{min}), r_k > 0.6 \\ min(a_k + \delta, a_{max}), r_k < 0.2 \\ a_k, 0.2 \le r_k \le 0.6 \end{cases}$$



例題測試

● 硬體測試環境

CPU： Intel® Core™ i5-12500H（12 Cores）
RAM： 16 GB
GPU： Intel Iris Xe Graphics
OS： Windows 11 Home
Device Model： ASUS VivoBook K3502ZA

● 軟體測試環境

Python： Python 3.12.0
IDE： Visual Studio Code
Optimizer： Gurobi Optimizer
OS Packages： 必要 Python 套件

Visual Studio Code　python　GUROBI OPTIMIZATION

評估演算法準確率之指標Gap計算方式如下：

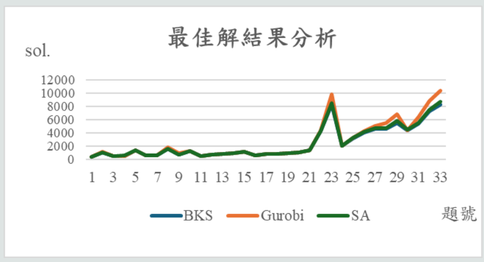
$$Gap = \frac{Sol. - BKS}{BKS} \times 100\%$$

例題結果

題號	n	T	BKS	Gurobi	SA
1	50	2	432.1	432.09	432.48
2	50	5	1105.81	1111.5	1106.15
3	50	5	466.71	466.7	466.74
4	75	2	549.05	549.01	564.73
5	75	5	1382.33	1375.09	1388.69
6	75	10	643.5	652.97	643.35
7	100	2	643.8	643.71	654.5
8	100	5	1613.42	1806.21	1613.24
9	100	8	721.4	896.55	726.34
10	100	5	1237.77	1259.01	1236.29
11	65	4	490.97	490.88	490.97
12	87	4	664.1	664.1	664.09
13	109	4	830.8	830.82	830.79
14	131	4	994.6	994.52	994.6
15	153	4	1157.07	1157.94	1157.07
16	48	4	660.12	660.12	662.28

題號	n	T	BKS	Gurobi	SA
17	66	4	776.43	776.38	777.4
18	84	4	873.73	873.66	873.08
19	102	4	958.51	959.28	960.38
20	120	4	1033.58	1043.58	1033.65
21	77	4	1375.07	1375.1	1375.07
22	154	4	4312.31	4364.26	4312.3
23	231	4	8308.51	9857.31	8495.67
24	48	4	2064.84	2060.81	2064.84
25	96	4	3207.44	3272.58	3227.02
26	144	4	4030.54	4292.94	4162.24
27	192	4	4558.94	5074.83	4670.54
28	240	4	4628.89	5514.44	4687.12
29	288	4	5534.94	6841.29	5877.25
30	72	6	4435.39	4384.85	4448.41
31	144	6	5376.11	6396.92	5437.69
32	216	6	7282.39	8855.79	7470.46
33	288	6	8280.07	10347.26	8657.44

結論及未來研究方向



1.模擬退火法於多數測試實例中，可在極短時間內取得與 BKS 接近之高品質解。
2.相較 Gurobi，SA 在中大型問題上展現較佳的計算效率與穩定性。結果顯示 SA 能有效平衡解品質與計算時間，適合大規模問題應用。
3.未來可結合其他鄰域結構或混合式方法以進一步提升解品質。