柔性期末結報

2019/01/13

工工碩一

R08546016

楊育禎

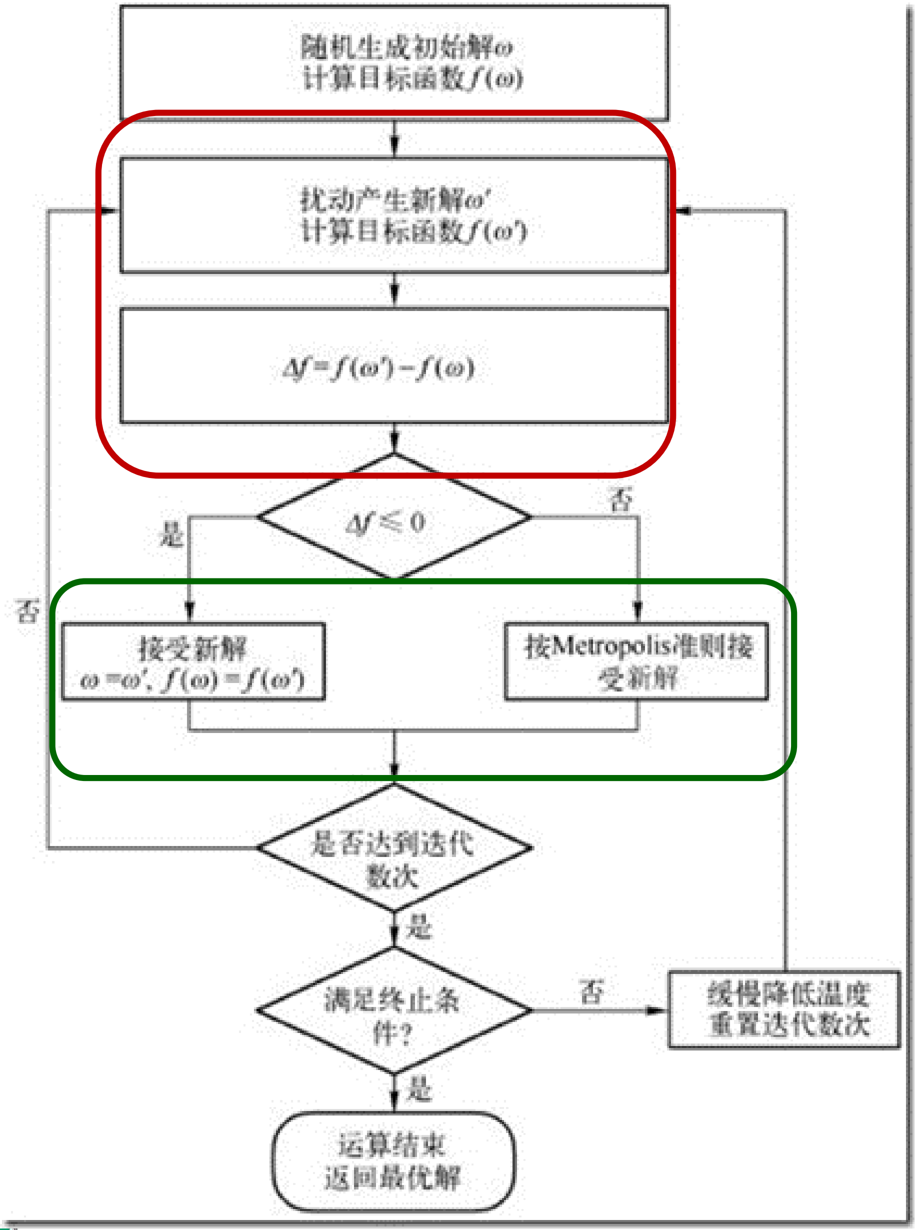
模擬退火演算法(SA)

- 模擬退火來自冶金學的專有名詞退火。
- 模擬退火算法所得解依**概率收斂**到全局最優解。
- 缺點:
  - 如果降溫過程夠緩慢，多得到的解的性能會比較好，但與此相對的是收斂速度太慢
  - 如果降溫過程過快，很可能得不到全局最優解。
- 概率收斂
  - 設$(X_n)_{n\geq 1}$是一個隨機變量序列，X是一個隨機變量，如果對於任意的正實數$\varepsilon > 0$都有:

$$\lim_{n\to\infty} P(|X - X_n| \geq \varepsilon) = 0$$

那麼則稱 $(X_n)_{n\geq 1}$ 一概率收斂到X。

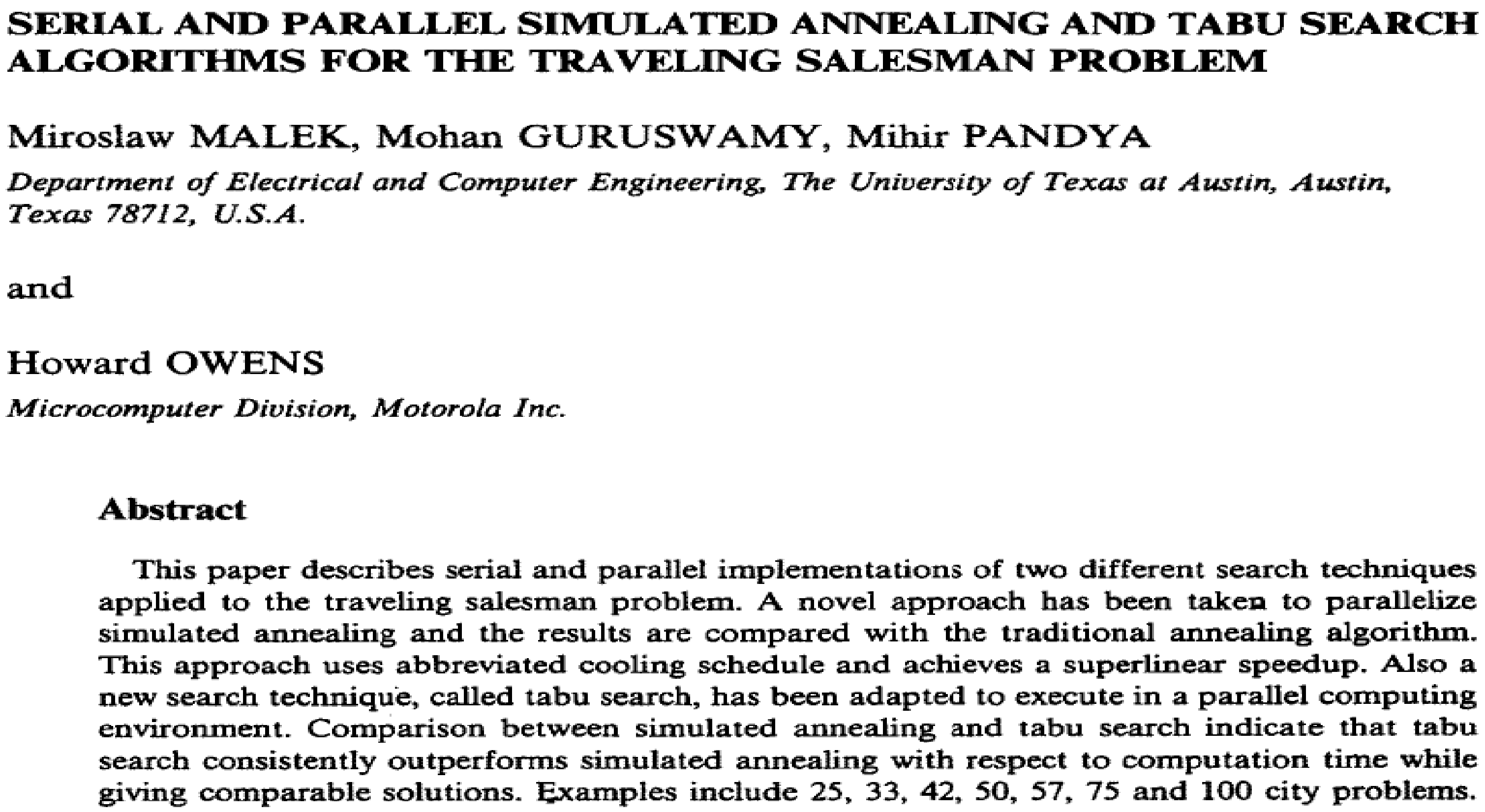
模擬退火演算法(SA)



- 模擬退火演算法與初始值無關
- Metropolis準則: 若Δf′<0則接受S′作為新的當前解S，否則以概率$\exp(\frac{-\Delta f'}{T})$接受S′作為新的當前解S
- 停止準則:溫度T降至某最低值時，完成給定數量疊代中無法接受新解，停止疊代，接受當前尋找的最優解為最終解
- 如何對溫度做有效的調整為模擬退火法中極為重要的一環。
  1. 溫度夠高時，系統要能夠自由變化，這種移動可以稱為Random Walk。
  2. 溫度變小時，移動受限制，逐漸往低能量的地方聚集

模擬退火演算法(SA)

- It is often used when the search space is **discrete**
- to approximate **global optimization** in a **large search space** for an optimization problem.
- 應用:



SERIAL AND PARALLEL SIMULATED ANNEALING AND TABU SEARCH ALGORITHMS FOR THE TRAVELING SALESMAN PROBLEM

Miroslaw MALEK, Mohan GURUSWAMY, Mihir PANDYA

Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, Texas 78712, U.S.A.

and

Howard OWENS

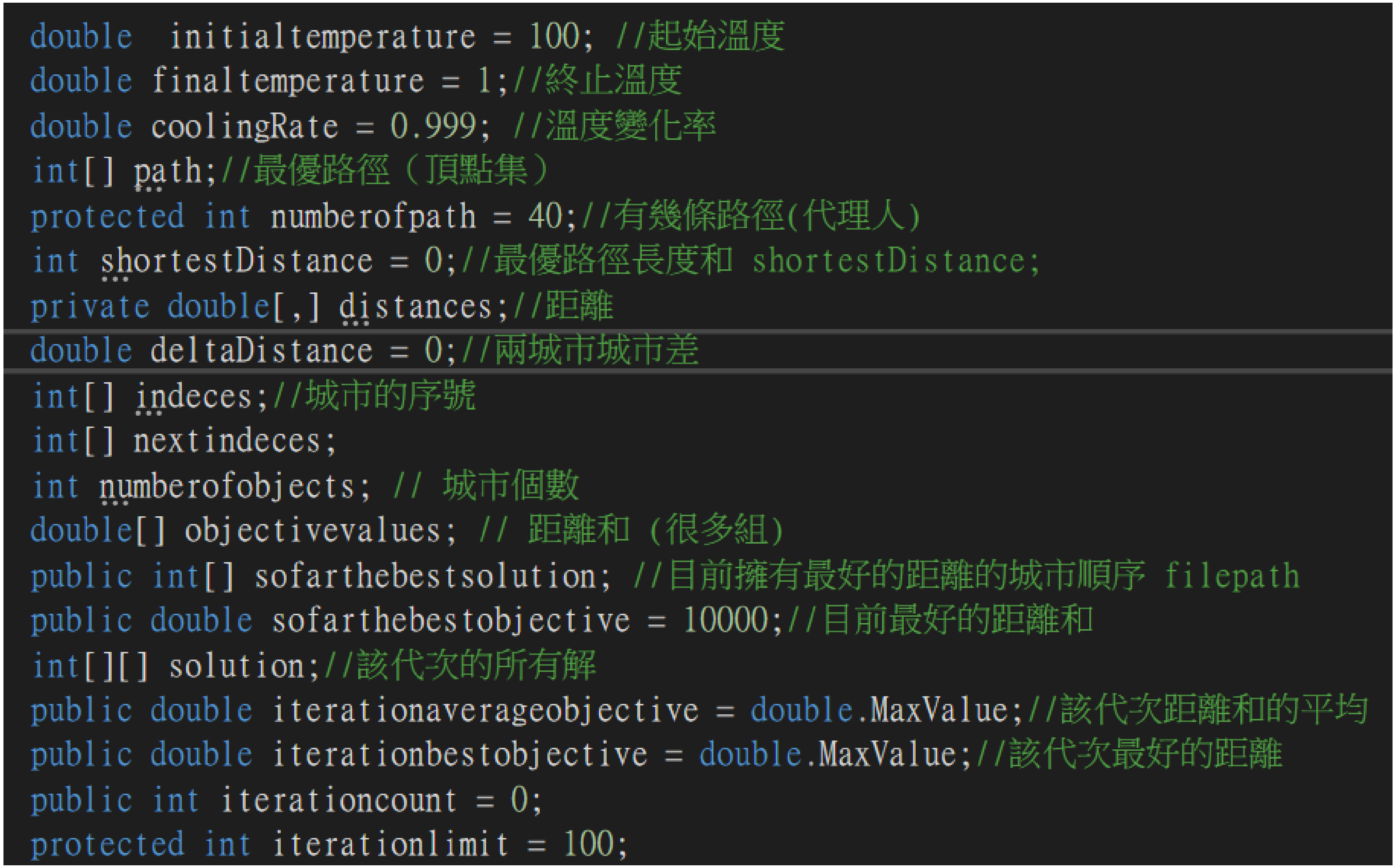
Microcomputer Division, Motorola Inc.

Abstract

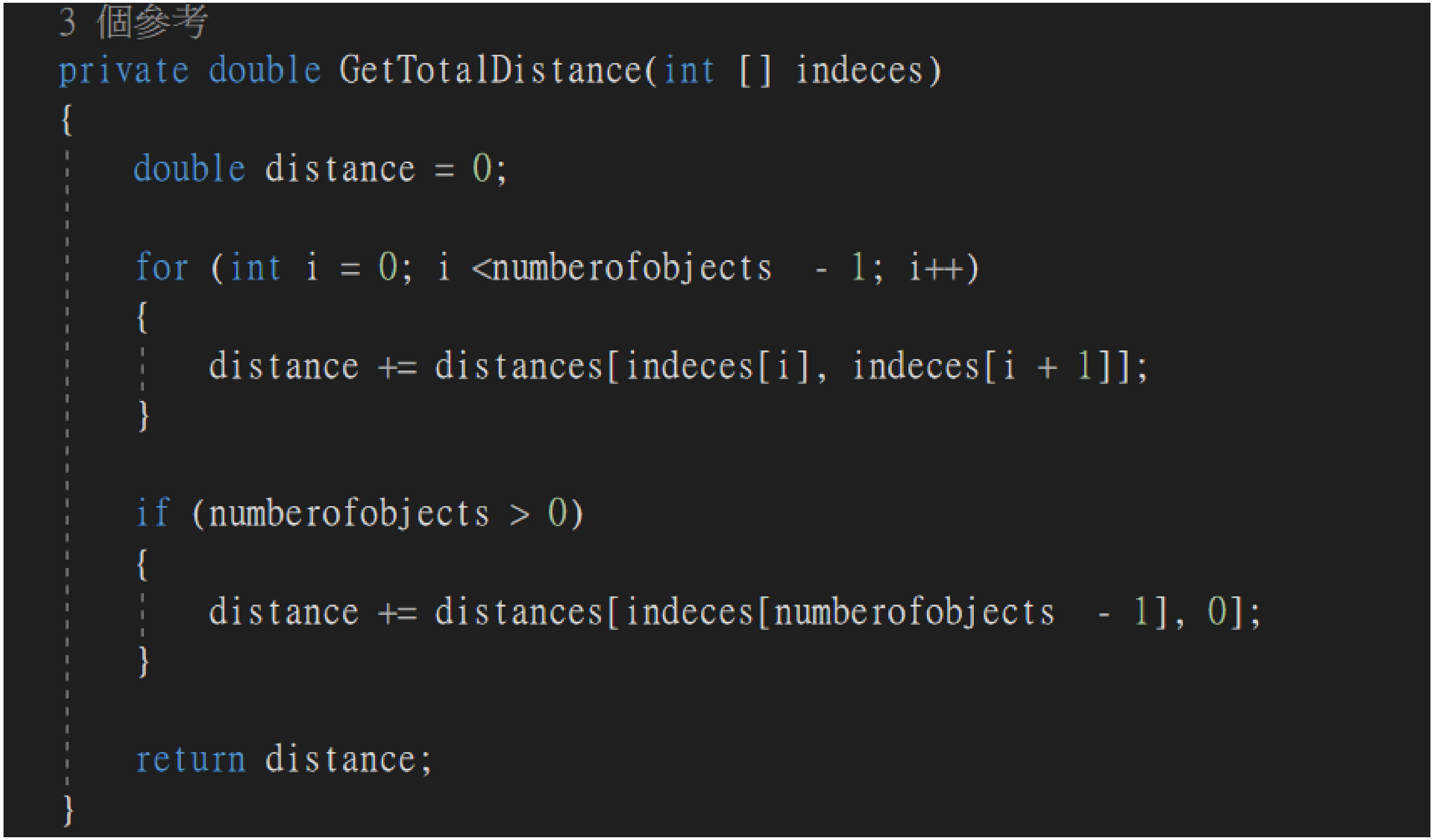
This paper describes serial and parallel implementations of two different search techniques applied to the traveling salesman problem. A novel approach has been taken to parallelize simulated annealing and the results are compared with the traditional annealing algorithm. This approach uses abbreviated cooling schedule and achieves a superlinear speedup. Also a new search technique, called tabu search, has been adapted to execute in a parallel computing environment. Comparison between simulated annealing and tabu search indicate that tabu search consistently outperforms simulated annealing with respect to computation time while giving comparable solutions. Examples include 25, 33, 42, 50, 57, 75 and 100 city problems.

應用

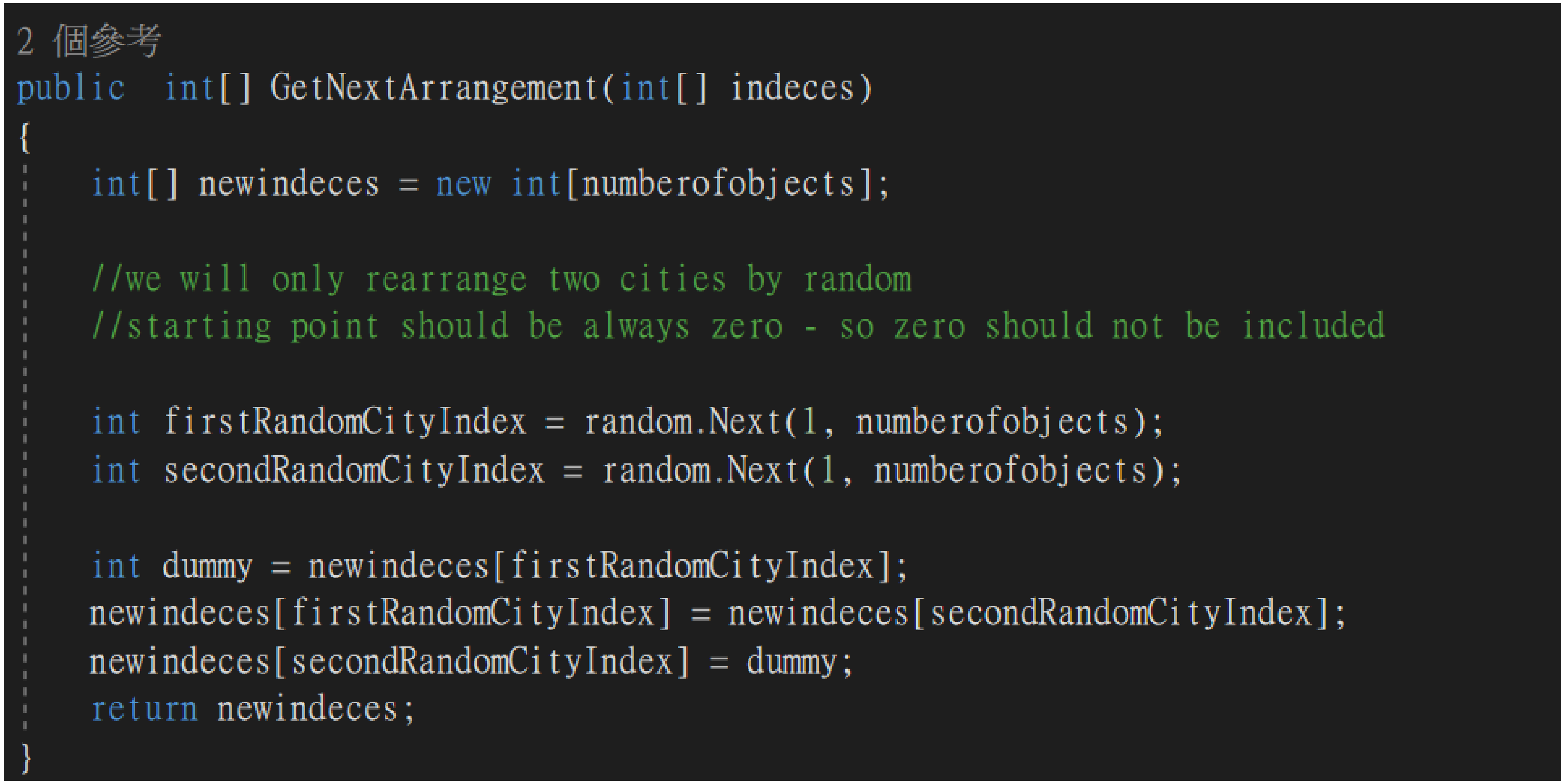
//思路：
1. 初始化：起始溫度，終止溫度，溫度變化率，最優路徑頂點集 ，最短長度length_sum
2. 利用蒙特卡洛法得到一個比較好的初始解
3. 當前溫度大於終止溫度時，利用兩點交換法在當前最優路徑基礎上構造一條新路徑，計算新路徑的長度S1，差值deltaDistance= S1 – S
4. 若 deltaDistance < 0，當前最優路徑更新為新路徑，最短長度更新為S1，否則，任意產生一個介於0~1的概率rt,並計算exp(-deltaDistance/T),T為當前溫度；若exp(-deltaDistance/T) > rt, 當前最優路徑更新為新路徑，最短長度更新為S1更新溫度 (代次)



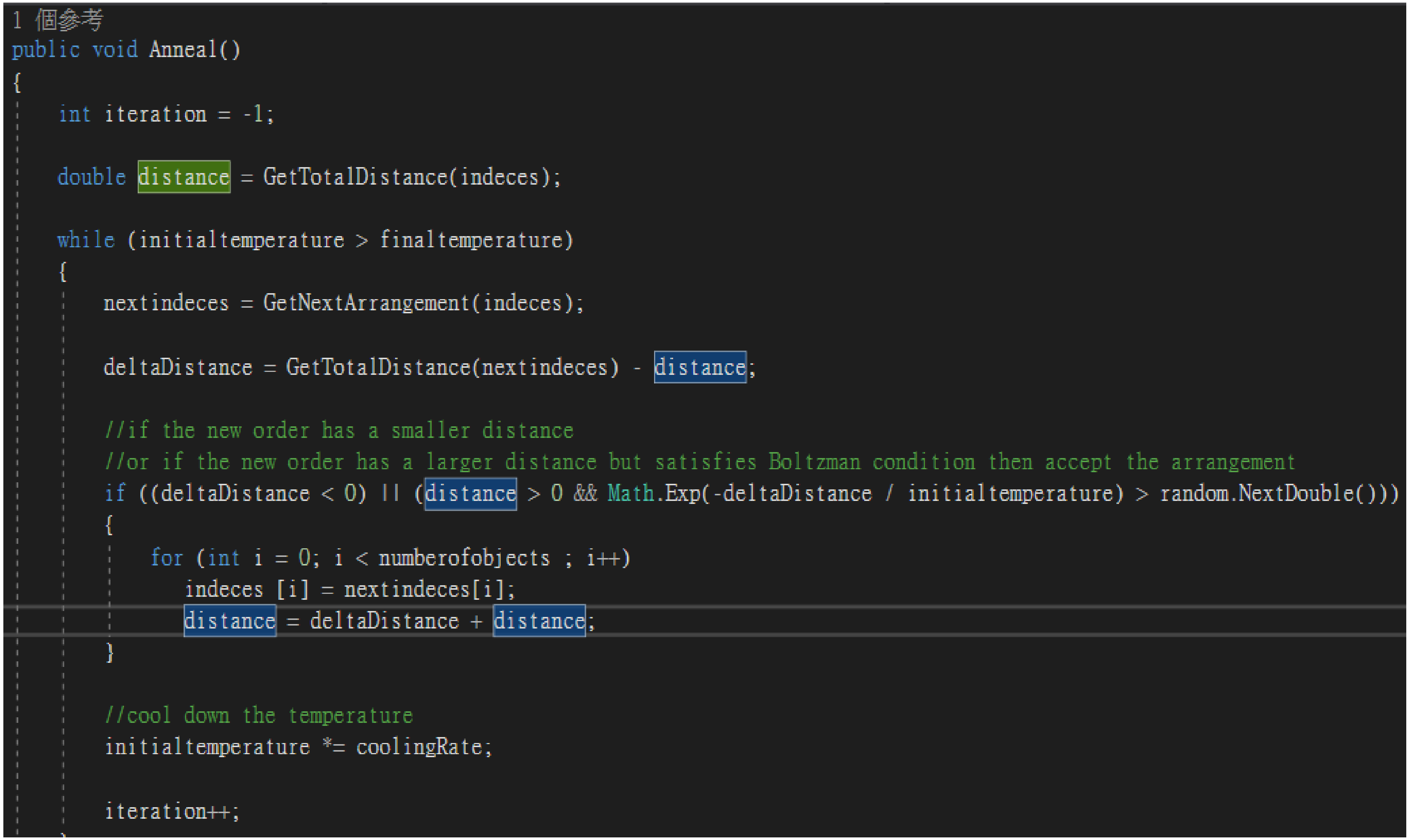
```
double  initialtemperature = 100; //起始溫度
double finaltemperature = 1;//終止溫度
double coolingRate = 0.999; //溫度變化率
int[] path;//最優路徑（頂點集）
protected int numberofpath = 40;//有幾條路徑(代理人)
int shortestDistance = 0;//最優路徑長度和 shortestDistance;
private double[,] distances;//距離
double deltaDistance = 0;//兩城市城市差
int[] indeces;//城市的序號
int[] nextindeces;
int numberofobjects; // 城市個數
double[] objectivevalues; // 距離和 (很多組)
public int[] sofarthebestsolution; //目前擁有最好的距離的城市順序 filepath
public double sofarthebestobjective = 10000;//目前最好的距離和
int[][] solution;//該代次的所有解
public double iterationaverageobjective = double.MaxValue;//該代次距離和的平均
public double iterationbestobjective = double.MaxValue;//該代次最好的距離
public int iterationcount = 0;
protected int iterationlimit = 100;
```



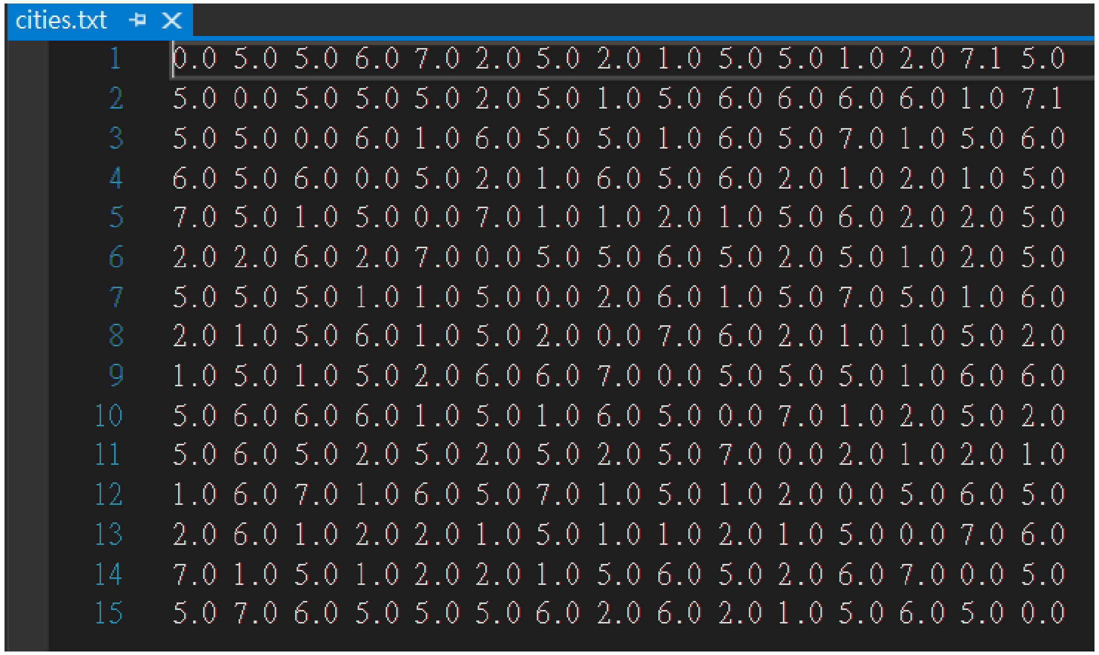
```
3 個參考
private double GetTotalDistance(int [] indeces)
{
    double distance = 0;

    for (int i = 0; i <numberofobjects  - 1; i++)
    {
        distance += distances[indeces[i], indeces[i + 1]];
    }

    if (numberofobjects > 0)
    {
        distance += distances[indeces[numberofobjects  - 1], 0];
    }

    return distance;
}
```



```
2 個參考
public  int[] GetNextArrangement(int[] indeces)
{
    int[] newindeces = new int[numberofobjects];

    //we will only rearrange two cities by random
    //starting point should be always zero - so zero should not be included

    int firstRandomCityIndex = random.Next(1, numberofobjects);
    int secondRandomCityIndex = random.Next(1, numberofobjects);

    int dummy = newindeces[firstRandomCityIndex];
    newindeces[firstRandomCityIndex] = newindeces[secondRandomCityIndex];
    newindeces[secondRandomCityIndex] = dummy;
    return newindeces;
}
```

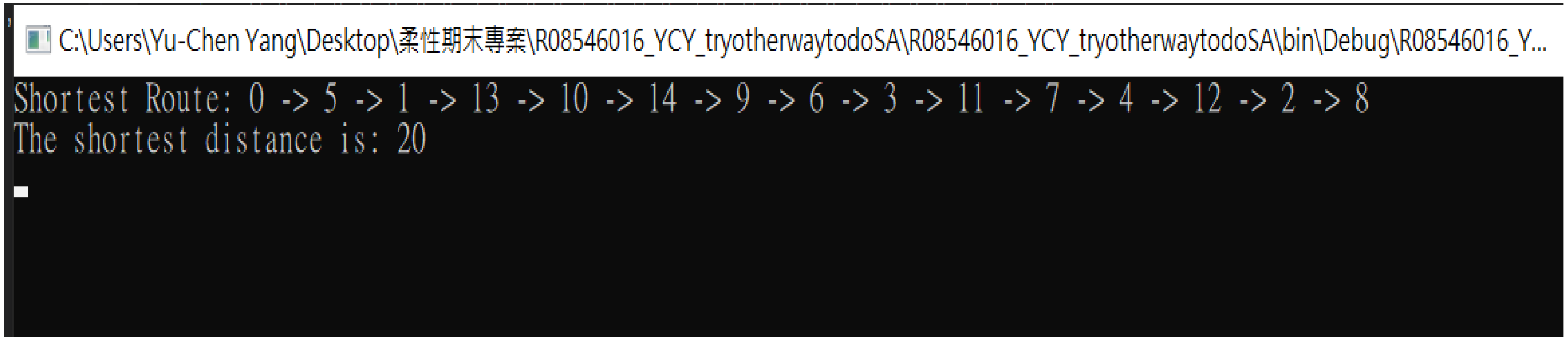


```
1 個參考
public void Anneal()
{
    int iteration = -1;

    double distance = GetTotalDistance(indeces);

    while (initialtemperature > finaltemperature)
    {
        nextindeces = GetNextArrangement(indeces);

        deltaDistance = GetTotalDistance(nextindeces) - distance;

        //if the new order has a smaller distance
        //or if the new order has a larger distance but satisfies Boltzman condition then accept the arrangement
        if ((deltaDistance < 0) || (distance > 0 && Math.Exp(-deltaDistance / initialtemperature) > random.NextDouble()))
        {
            for (int i = 0; i < numberofobjects ; i++)
                indeces [i] = nextindeces[i];
                distance = deltaDistance + distance;
        }

        //cool down the temperature
        initialtemperature *= coolingRate;

        iteration++;
```



cities.txt

```
0.0 5.0 5.0 6.0 7.0 2.0 5.0 2.0 1.0 5.0 5.0 1.0 2.0 7.1 5.0
5.0 0.0 5.0 5.0 5.0 2.0 5.0 1.0 5.0 6.0 6.0 6.0 6.0 1.0 7.1
5.0 5.0 0.0 6.0 1.0 6.0 5.0 5.0 1.0 6.0 5.0 7.0 1.0 5.0 6.0
6.0 5.0 6.0 0.0 5.0 2.0 1.0 6.0 5.0 6.0 2.0 1.0 2.0 1.0 5.0
7.0 5.0 1.0 5.0 0.0 7.0 1.0 1.0 2.0 1.0 5.0 6.0 2.0 2.0 5.0
2.0 2.0 6.0 2.0 7.0 0.0 5.0 5.0 6.0 5.0 2.0 5.0 1.0 2.0 5.0
5.0 5.0 5.0 1.0 1.0 5.0 0.0 2.0 6.0 1.0 5.0 7.0 5.0 1.0 6.0
2.0 1.0 5.0 6.0 1.0 5.0 2.0 0.0 7.0 6.0 2.0 1.0 1.0 5.0 2.0
1.0 5.0 1.0 5.0 2.0 6.0 6.0 7.0 0.0 5.0 5.0 5.0 1.0 6.0 6.0
5.0 6.0 6.0 6.0 1.0 5.0 1.0 6.0 5.0 0.0 7.0 1.0 2.0 5.0 2.0
5.0 6.0 5.0 2.0 5.0 2.0 5.0 2.0 5.0 7.0 0.0 2.0 1.0 2.0 1.0
1.0 6.0 7.0 1.0 6.0 5.0 7.0 1.0 5.0 1.0 2.0 0.0 5.0 6.0 5.0
2.0 6.0 1.0 2.0 2.0 1.0 5.0 1.0 1.0 2.0 1.0 5.0 0.0 7.0 6.0
7.0 1.0 5.0 1.0 2.0 2.0 1.0 5.0 6.0 5.0 2.0 6.0 7.0 0.0 5.0
5.0 7.0 6.0 5.0 5.0 5.0 6.0 2.0 6.0 2.0 1.0 5.0 6.0 5.0 0.0
```



C:\Users\Yu-Chen Yang\Desktop\柔性期末專案\R08546016_YCY_tryotherwaytodoSA\R08546016_YCY_tryotherwaytodoSA\bin\Debug\R08546016_Y...

```
Shortest Route: 0 -> 5 -> 1 -> 13 -> 10 -> 14 -> 9 -> 6 -> 3 -> 11 -> 7 -> 4 -> 12 -> 2 -> 8
The shortest distance is: 20
```

結果與討論

- 模擬退火法因為架構簡單被廣泛應用於組合最佳化問題，能快速地得到優良解。最終解受退火策略影響很大，應用在不同的問題上需要有適合的退火測略，通常以試誤法調整。
- 相較於其他演算法模擬退火法的結果都比較差，是因為它架構簡單，鄰近解是隨機產生。不會侷限在局部，能跳脫卡死狀態。
- 但論文中的另一個方法禁忌搜尋法又比模擬退火還更加好，期許自己在未來可以有能力再繼續探討下去。

THANK YOU
FOR LISTENING