電物競賽-求解方法

111701035葉政凱、111703050李興旺

一、前言

因為已知組合最佳化是 NP-Hard 問題，所以我們一開始就從啟發式演算法著手，最終選擇了模擬退火以及基因演算法，並同時使用以確保我們得到的是全域最佳解。至於程式語言的部分，雖然 python 在使用上會較為方便，但考量到運行速率，我們最終決定使用 C++。

二、使用演算法簡介

模擬退火（simulated annealing, SA）：

模擬退火來自冶金學的專有名詞「退火」，是一種機率演算法。不同於梯度下降法，引入了溫度這個變數，有可能接受不是當前的較優解，可證明模擬退火會依機率收斂到全域最佳解。

基因演算法（Genetic Algorithm，GA）：

基因演算法是演化演算法的一種，把每個解當成是生物的染色體，然後透過交配、突變以及篩選等運算，不斷篩選出適應值較高的個體。是一種全域搜尋的隨機演算法，常用於解決最佳化問題，最後出來的的結果收斂到全域最佳解的機率較高。

三、程式使用說明

直接把 w101.txt（無須更動內容）與程式放在同個根目錄下後編譯（我們使用的編譯參數為-std=c++17 -O3）並執行程式即可。若要調整參數可於 main 中調整，註解掉第 246 行或第 254 行後可單獨執行其中一種演算法。執行完後會生成 distribution.txt 和 E.txt，為當次執行所找到最佳的分配方式和該分配下的 E 值大小。

四、程式說明

因為知道將全部的 A、B 班互換所得出的 E 值是與交換前相等的，所以我們固定編號 1 為 A 班（程式中為了方便換成 0-based，這邊以 1-based 作說明）。因為 SA 和 GA 都跟機率有關，因此我們讓它們各跑 Round_Times 輪，取其中出現的最大 E 值作為答案。

以下將分別說明 SA 和 GA。

SA：

在每一輪剛開始都會先隨機分配產生編號 2~101 的初始狀態並計算 E 值。先設目前溫度（cur_temperature）為初始溫度（Initial_Temperature），然後隨機尋找一組鄰近狀態，在這裡我們是在 2~101 中隨機選擇一個編號並把該編號的同學換到另一班並計算新的 E 值。接著利用類似 Metropolis 接受準則的方法，如果 ΔE>0 就直接保留新的狀態（因為要找最大值）；否則先生成一個隨機數∈[0, 1]，如果 exp(ΔE/cur_temperature) 大於這個隨機數就保留狀態（此時ΔE<0，ΔE前不用加負號），否則就把狀態換回。接著降低溫度（cur_temperature *= Alpha），重複執行以上動作直到目前溫度低於終止溫度（End_Temperature），就完成當前這輪。

在這邊要特別提到我們用來計算 E 值的方法，起初我們每次計算 E 值都是把 5050 組關係全部跑過，但是根據我們尋找鄰近狀態的方法，每次只改變一個編號，把每組關係都跑過有點太過冗餘。我們後來想到可以把每個編號看成頂點，每組關係看成邊，$W_{i,j}$ 為邊上的權重，形成一張完全圖（程式中為了方便沒有排除 i=j 的狀況，而是將權重設為 0）。改變一個點只需要跑過跟他相連的邊。在跑過一條邊時，只要把當前 E 值減掉兩倍的邊權重，並把邊權重加負號，全部跑完後當前 E 值即為改變那個點後的 E 值。除了計算初始狀態的 E 值，其他在計算 E 值時都可以看成圖，把 5050 次優化到 101 次。

GA：

我們把班級的分配當成染色體（distribution），每一個個體包含染色體以及適應值（E），然後根據 GA，想要求得最佳子代我們會需要以下運算：

1.計算適應值（CountFitness）

因為目標是最高適應值，我們把 E 值當作適應值。

2.挑選親代（ParentSelection ）

隨機取族群數量（Population_Size）10%的族群當作樣本，最後選出當中前兩高適應值的做為親代。

3.交配（Crossover）

使用 ParentSelection 所挑出的親代 ，隨機分配基因給兩個子代，接著為了增加個體多樣性，每一個子代的染色體都會發生突變。

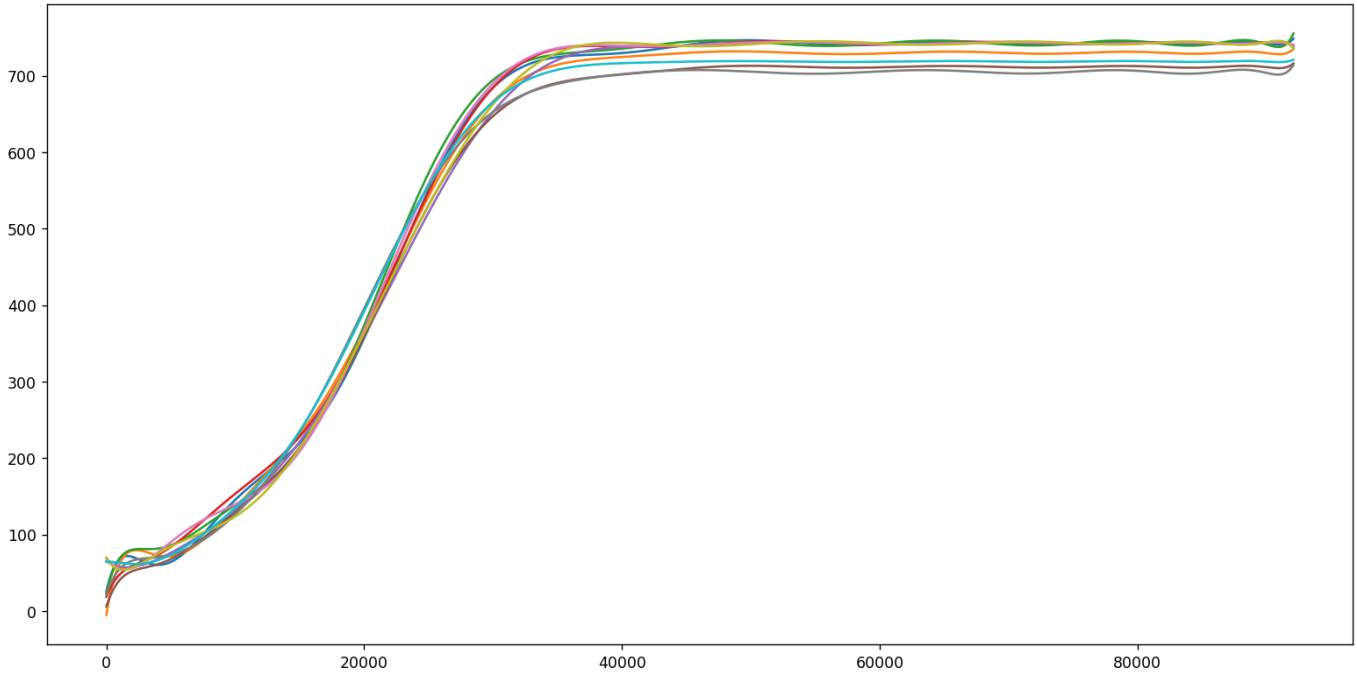
4.突變（Mutation）

我們把突變定義為對於編號 2~101 的基因位都有一定機率（Mutation_Ra-te）會交換班級。

在每一輪 GA 一開始，先用隨機初始化整個族群，然後開始迭代我們的族群， 為了產生的子代，我們先用 ParentSelection 函式挑選出父親和母親然後用 Crossover 函式產生子代，重複 Offspring_Quantity 次，最後把族群根據適應值進行排序，把適應值較低的個體淘汰直到族群恢復到原大小，這樣一次稱為一個世代（Generation），重複 Max_Generation 次，選出適應值最高的個體當作答案，完成這一輪。

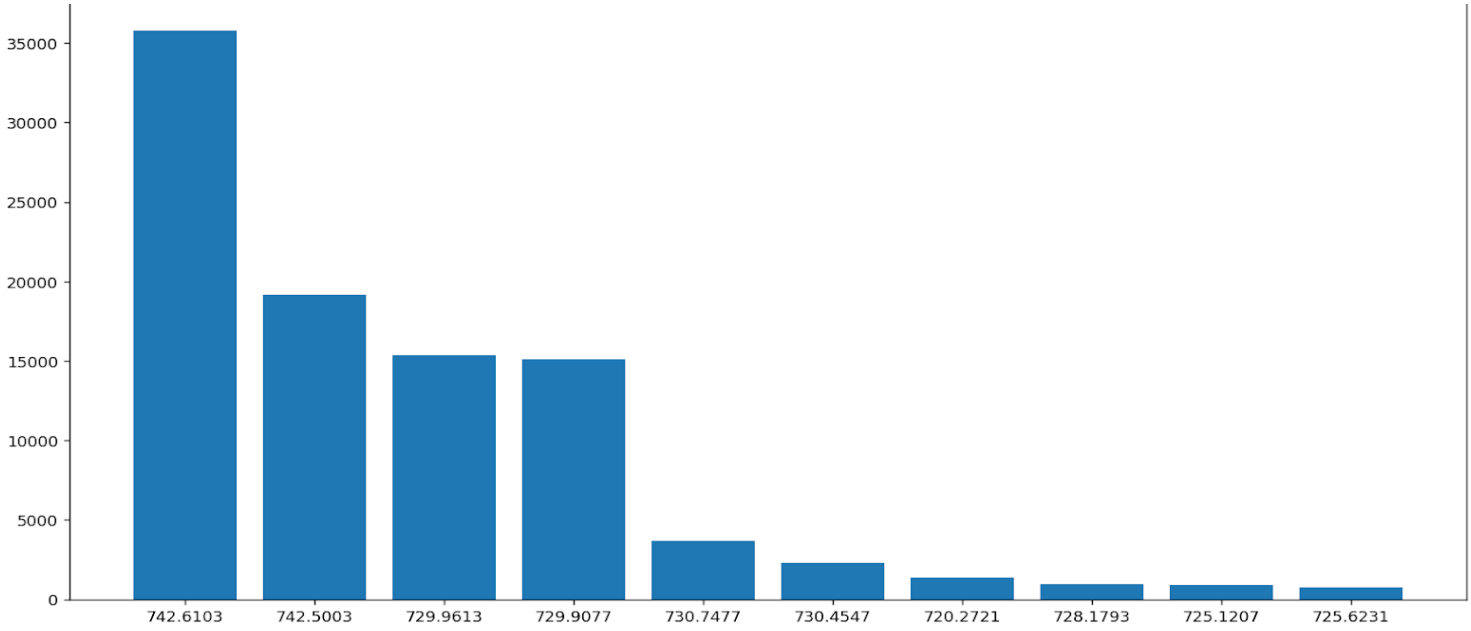
五、E 值的趨勢

為了確認演算法是對的，我們執行了 10 輪 SA 並記錄過程中的 E 值，為了方便觀察，我們利用 python 把數據擬合成線性函數，X 軸為迭代次數（溫度乘以 Alpha 的次數）、Y 軸為E 值，結果如下圖：



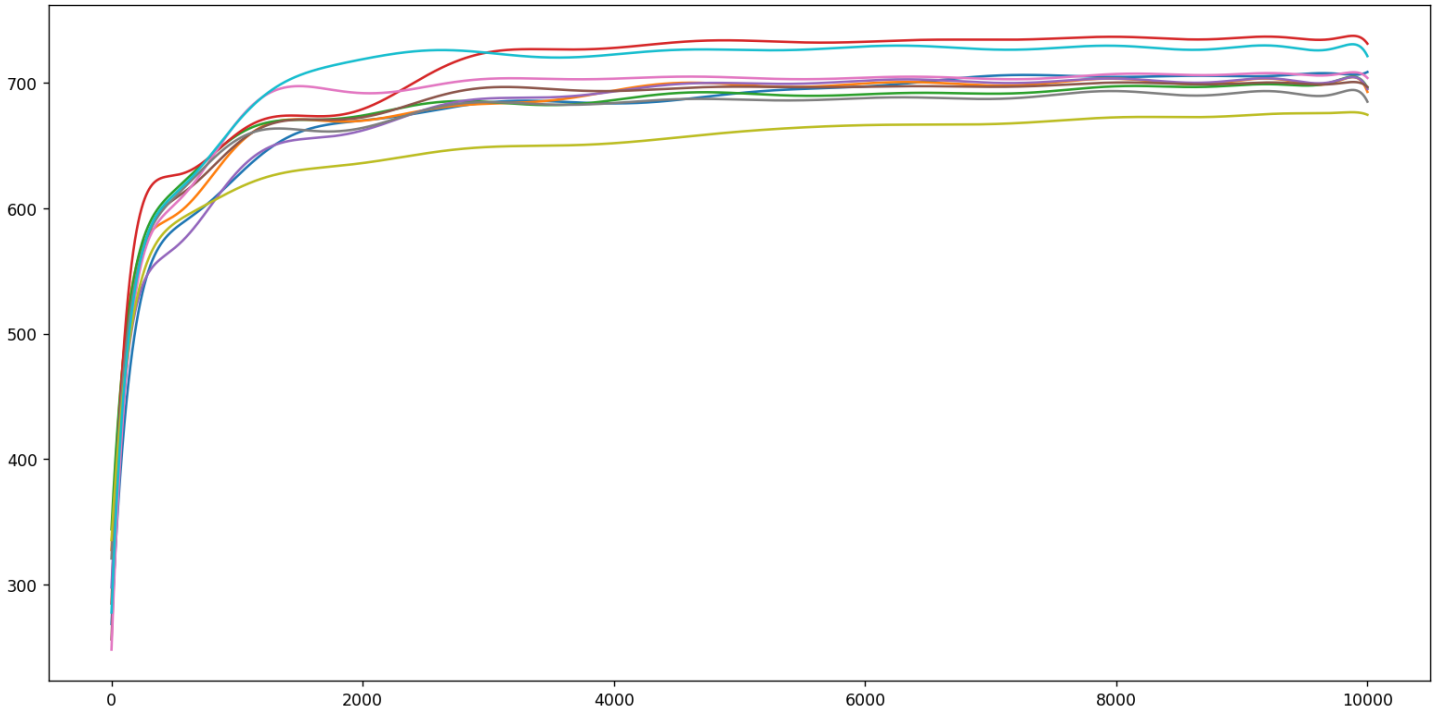
可以發現，不管初始狀態如何，E 值的趨勢都是隨著運行時間增加而增加，雖然最後 E 值有些許落差，但是可以確定我們的方向是對的。

另外我們做了十萬次的 SA，取出出現次數最高的前 10 名， E 值為 X 軸 、出現次數為 Y 軸，並把出現次數以降序排列，描繪出柱狀圖如下：



可以發現 742.6103 出現的次數最多同時也是我們所找到的最大值。

分析完 SA，同樣的我們取 10 輪 GA 的執行過程，擬合成線性函數，X 軸為迭代次數（第幾世代）、Y 軸為 E 值做成折線圖如下：



可以發現，在我們的參數設定下，E 值整體也是往 700 附近收斂，但陷入局部最佳的機率比 SA 大，不過收斂速度比 SA 快，但以時間複雜度來看 GA 較 SA 大許多，結果還是 SA 跑得比較快。

六、後記

為了確保我們的答案是全域最佳，我們還有改變 SA 的尋找鄰近狀態的方法，例如每次改變 2 個編號、3 個編號、或是根據溫度決定改變的個數，以及 GA 的交配和突變的方法，例如cut-and-crossfill crossover、固定突變2、3個基因位，所得到的最大 E 值一樣是 742.6103。

我們還使用了平行運算的手法，讓 SA 和 GA 分別跑一億輪、一千萬輪，最後得到的答案仍然為 742.6103，不過因為我們所寫的平行運算會有些 bug，就不附上程式碼了。