多處理機平行程式設計 2021 Homework 6

學號：F74076027
姓名：林政傑
系級：資訊 111

測試環境：Linux ubuntu 5.11.0-40-generic
with 4 Docker ubuntu:18.04 image
CPU：Intel i7-10750H (6 core 12 thread) @ 2.6GHz
GCC version：7.5.0

Ant Algorithm

What have you done

中規中矩的實做 Ant colony algorithm，在 process 0 讀入 txt 檔，透過 `MPI_Bcast` 把 txt 的矩陣散播到所有 process，執行演算法時，每做一次 iteration 就透過 `MPI_Allreduce(..., MPI_SUM, ...)` 同步每個 process 的費落蒙，最後計算出 best tour。

透過以下 marcro 可以更改 ant-algorithm 的參數：

```
#define ITERATION 10000 // 釋放螞蟻的次數
#define ANT_NUM   500   // 螞蟻數量
#define ALPHA     0.5   // 費落蒙的指數
#define BETA      3.0   // （距離分之一）的指數
#define RHO       0.5   // 費落蒙蒸發的比率
#define THREAD_NUM 1    // OpenMP 的執執行序數量
```

透過以下指令編譯執行：

```
$ mpicc h6_problem1.c -o a -lm -fopenmp
$ mpiexec -np 4 ./a ./gr17_d.txt
Iteration 0
Iteration 100
Iteration 200
...
Iteration 9900
Best tour: 2085
Time: 5.811188
```

Analysis on your result

在我的實做中，使用 OpenMP 平行化更新費落蒙的迴圈會因為 loop dependency 導致錯誤的結果，其餘迴圈能用 OpenMP 的區塊都有用。

我將以下參數固定：

```
#define ITERATION 10000 // 釋放螞蟻的次數
#define ANT_NUM   500   // 螞蟻數量
#define ALPHA     0.5   // 費落蒙的指數
#define BETA      3.0   // （距離分之一）的指數
#define RHO       0.5   // 費落蒙蒸發的比率
```

透過調整 MPI 的 `-np` 和 OpenMP 的 thread 數量觀察程式執行 GR17、FRI26、DANTZIG42、ATT48 的表現。首先固定 OpenMP 的 thread 為 1，單純調整 MPI 的 `np`：

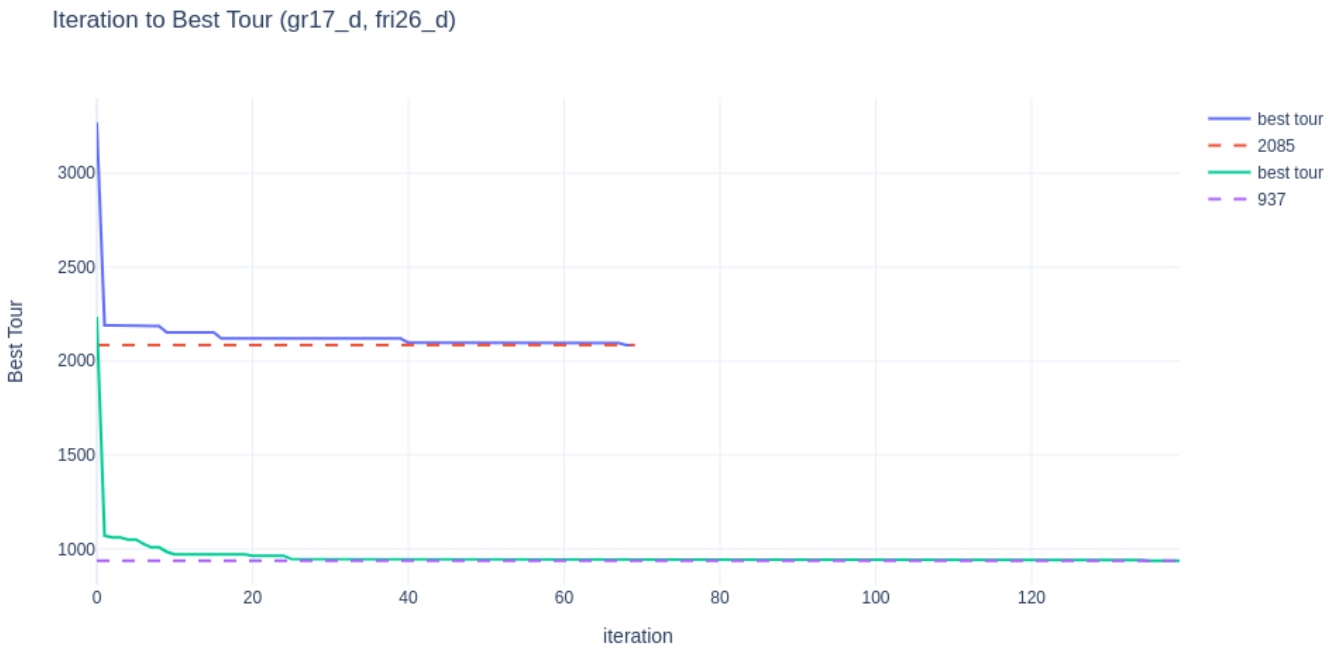
跟預期的一樣，`-np` 越高執行時間越短。接下來調整 OpenMP 的 thread 數量：

首先可以發現一個很有趣的現象，對於 `gr17_d` 和 `fri26_d` 來說 OpenMP (OMP) 的 thread 數量越高執行時間反而越長，也就是說當輸入資料太小，使用 OpenMP 的 overhead 反而高於平行帶來的優化效果。

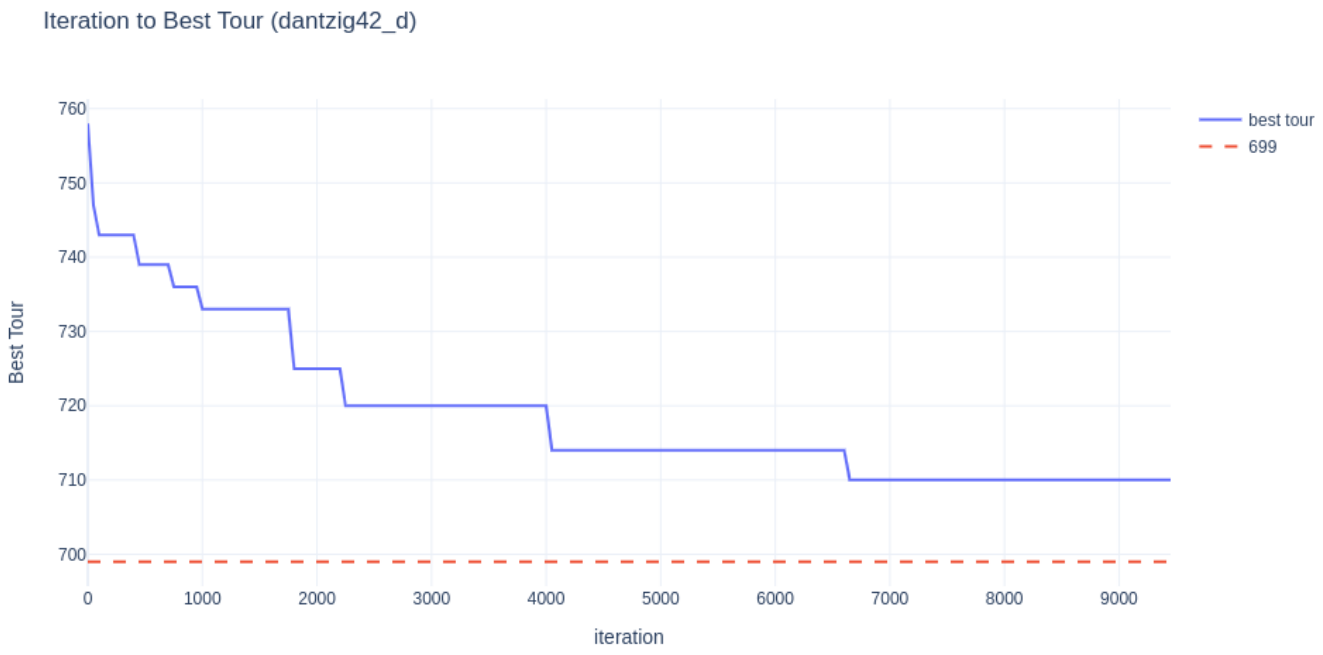
再來，真正帶來顯著優化的是調整 MPI 的 `-np`，而非調整 OpenMP 的 thread。就 att48_d 來說，在 MPI_1 的狀況下 [OMP_2, OMP_4, OMP_6] 執行時間介於 74 到 85 秒，而 MPI_2 幾乎降低至 MPI_1 的一半，也就是 39 到 49 秒。所以 OpenMP 在 Ant-Algorithm 中比較偏細節的調優，並沒有決定性的效能突破。

不過上述兩張圖的結果也可能是 Docker 環境的特性，因為在 Docker 中不同的 Container 實際上還是在同一台電腦上跑，所以 MPI 的 overhead 很低。考慮到大型分散式系統的環境，也許 MPI 會因為需要通過網路線傳輸使得 overhead 大幅提升，實驗的結果就可能完全不同了。

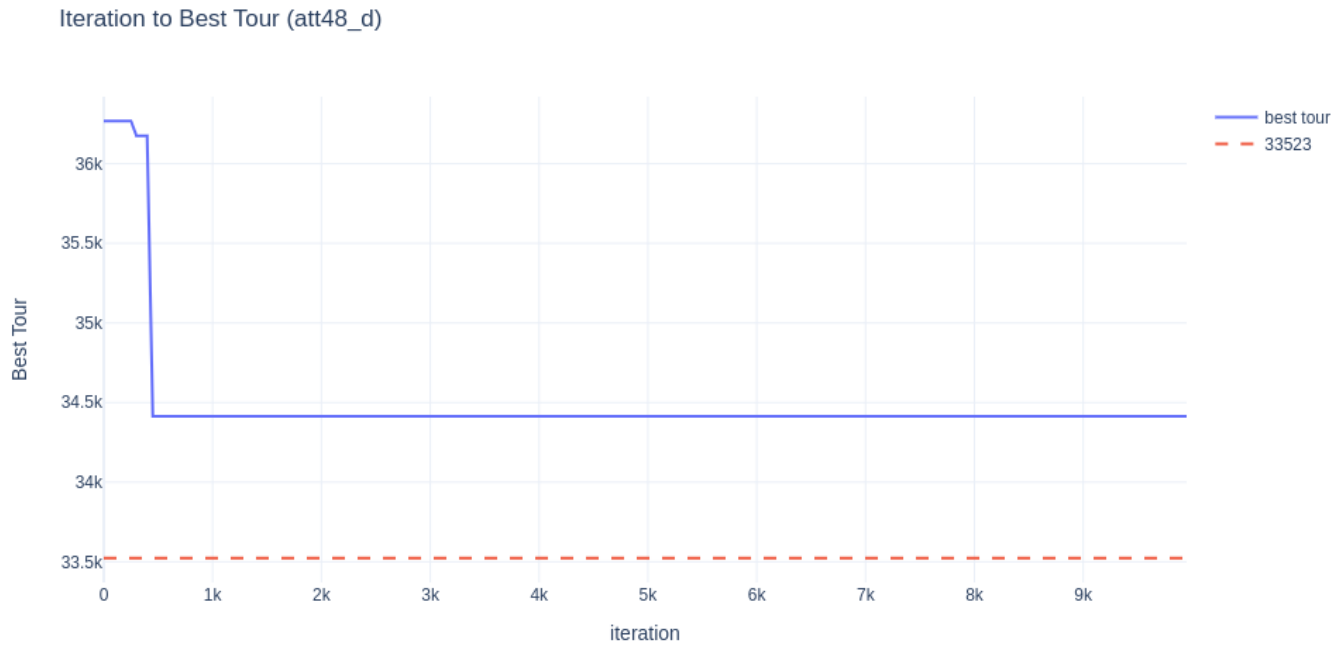
最後這個實驗展示 Ant-Algorithm 收斂到最佳解需要的 Iteration，因為時間有限，我沒有測試不同的 `ALPHA`、`BETA`、`RHO` 帶來的影響，所有參數都與先前實驗一樣。



上圖顯示 `gr17_d` 和 `fri26_d` 在 150 個以內的 iteration 就收斂到最佳解。



`dantzig42_d` 就沒那麼幸運，在 6700 個 iteration 收斂到 710 並且無法到達 699。



`att48_d` 也無法收斂到最佳解。

Any difficulties?

在 MPI 平行程式管理 malloc 實在太難，查了半天還是不知道怎麼用 gdb 搭配 mpiexec，最後只好把 `int *xxx = malloc(n*sizeof(int));` 全都改成 `int xxx[n];` 才解決問題。

一開始的寫法只針對 MPI 寫，沒有考慮到 OpenMP 不好處理 loop dependency，所以很多計算 tour 和 pheromone 的部份就無法使用 OpenMP。