F74109016 葉惟欣 多處理機平行程式設計 HW6

題目: Ant Algorithm

Compile and execute

```
F74109016@sivslab-pn1:~$ mpicc -o problem6.exe problem6.c -fopenmp -lm
F74109016@sivslab-pn1:~$ mpiexec -np 5 ./problem6.exe gr17_d.txt
filename = gr17_d.txt
number of ant of colony : 40
number of thread : 10
cities : 17
alpha = 2
beta = 5
rho = 0.800000
Q = 2000
The shortest path of core2 is 2159
The shortest path of core1 is 2159
The shortest path of core0 is 2159
The shortest path of core3 is 2159
The shortest path of core4 is 2095
The shortest path is 2095
```

```
F74109016@sivslab-pn1:~$ mpicc -o problem6.exe problem6.c -fopenmp -lm
F74109016@sivslab-pn1:~$ mpiexec -np 5 ./problem6.exe fri26_d.txt
filename = fri26_d.txt
number of ant of colony : 40
number of thread : 10
cities : 26
alpha = 2
beta = 5
rho = 0.800000
Q = 2000
The shortest path of core1 is 965
The shortest path of core4 is 963
The shortest path of core0 is 955
The shortest path of core2 is 937
The shortest path of core3 is 946
The shortest path is 937
```

```
F74109016@sivslab-pn1:~$ mpiexec -np 5 ./problem6.exe dantzig42_d.txt
filename = dantzig42_d.txt
number of ant of colony : 40
number of thread : 10
cities : 42
alpha = 2
beta = 4
rho = 0.700000
Q = 1000
The shortest path of core1 is 753
The shortest path of core2 is 762
The shortest path of core3 is 796
The shortest path of core4 is 792
The shortest path of core0 is 737
The shortest path is 737
F74109016@sivslab-pn1:~$
```

```
F74109016@sivslab-pn1:~$ mpicc -o problem6.exe problem6.c -fopenmp -lm
F74109016@sivslab-pn1:~$ mpiexec -np 5 ./problem6.exe att48_d.txt
filename = att48_d.txt
number of ant of colony : 40
number of thread : 10
cities : 48
alpha = 2
beta = 5
rho = 0.800000
Q = 40000
The shortest path of core3 is 37327
The shortest path of core4 is 38105
The shortest path of core0 is 34751
The shortest path of core2 is 36761
The shortest path of core1 is 36417
The shortest path is 34751
```

問題 1 : What have you done?

使用 MPI+OpenMP 實作，每一台電腦各啟動一個 process，每個 process 再 fork 出 multi-thread

(一) Prepare :

一開始先將程式透過 MPI 的方式讓每個電腦各啟動一個 process

```
27
28      /**MPI*****************************************/
29      int myid, numprocs;
30      char processor_name[MPI_MAX_PROCESSOR_NAME];
31      MPI_Init(&argc,&argv);
32      MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
33      MPI_Comm_rank(MPI_COMM_WORLD,&myid);
34      /**MPI*****************************************/
```

由 rank 0 去 determine the number of cities

and then rank 0 broadcast the number of cities to other core

```
/****rank = 0 ***************************/
if(myid == 0){
    filename = argv[1];
    //determine the number of cities
    if(strcmp(filename,"fri26_d.txt")==0) cities = 26;
    else if(strcmp(filename,"gr17_d.txt")==0) cities = 17;
    else if(strcmp(filename,"dantzig42_d.txt")==0)cities = 42;
    else if(strcmp(filename,"att48_d.txt")==0)cities = 48;
    printf("filename = %s\n",filename);
    printf("number of ant of colony : %d\n",numberAnt);
    printf("number of thread : %d\n",num_threads);
    printf("cities : %d\n",cities);
    printf("alpha = %d\n",alpha);
    printf("beta = %d\n",beta);
    printf("rho = %f\n",rho);
    printf("Q = %d\n",Q);
}
/****rank = 0 ***************************/


//rank0 broadcast the number of cities to other core
MPI_Bcast(&cities,1,MPI_INT,0,MPI_COMM_WORLD);
// every core have to allocate a two dimensional matrix
malloc2dint(&dist, cities, cities);
```

Then rank 0 broadcast the number of cities to other core

```
//rank0 broadcast the number of cities to other core
MPI_Bcast(&cities,1,MPI_INT,0,MPI_COMM_WORLD);
// every core have to allocate a two dimensional matrix
malloc2dint(&dist, cities, cities);
```

rank = 0 read file and record distance and broadcast the distance of the matrix to other core. To broadcast the distance matrix I have to broadcast every row one by one . Because I allocate memory dynamically, 我們不能確定發送端 或 接收端兩個 allocate 的二維陣列為連續的只能確認每一列為連續的記憶體空間，因此發送將每一列 broadcast 都單獨送給接收端的每一列來避免 segmentation fault。

```
62      /****rank = 0 read file and record distance***/
63      if(myid == 0 ){
64          FILE *fp = fopen(filename,"r");
65          for(int i=0;i<cities;i++){
66              for(int j=0;j<cities;j++){
67                  fscanf(fp,"%d ",&dist[i][j]);
68              }
69          }
70          fclose(fp);
71      }
72      /****rank = 0 ***************************/
73
74      //Wair for the finish of root processor read file
75      MPI_Barrier(MPI_COMM_WORLD);
76
77      /****rank 0  broad cast the distance of the matrix to other core****/
78      for(int i=0;i<cities;i++){
79        MPI_Bcast(&(dist[i][0]),cities,MPI_INT,0,MPI_COMM_WORLD);
80      }
81      /****rank 0  broad cast the distance of the matrix to other core****/
```

接下來宣告 global matrix of pheromone 的二維陣列，越來存每個 process 中最佳解時的 pheromone 濃度。

(二) paralleling process by (num_threads) 個 thread by Open MP

```
95      /**** Each thread runs a whole ant colony, which will affect the critical section *****/
96  # pragma omp parallel num_threads(num_threads) shared(global_best_between_clonies)
```

Step 1 : Variable

Shared variable ：

global matrix of pheromone 的二維整數陣列 與

整數變數 global_best_between_clonies

Private variable ：

Visit 1 D integer array : record which city the ant visit

next 1 D integer array : record the next city ant pass

Prob 1 D double array : record the probability of not visited city

local_best_in_every_colony integer : the local shortest of every colony (thread)

pheromone , newpheromone 2 D double array

Step 2 : Initialize the pheromone matrix

Case 1 : If it is not the first ant of the colony it will reference the pheromone that ant pass previous, but the pheromone will decrease with parameter "rho"

$$\tau_{ij} \leftarrow (1-\rho).\tau_{ij} + \sum_{k=1}^{m} \Delta \tau_{ij}^{k}$$

Where ρ is the evaporation rate, m is the number of ants, and $\Delta\tau_{ij}^{k}$ is the quantity of pheromone laid on edge(i, j) by an ant k:

Case 2 : If it is the first ant of the colony every path is unknown ,so the pheromone is random number.

```
116         for(int i=0;i<numberAnt;i++){
117             for(int j=0;j<cities;j++){
118                 visit[j] = false;
119                 next[j] = -1;
120                 for(int k=0;k<cities;k++){
121                     if(i!=0){
122                         // If it is not the first ant of the colony it will reference the pheromone
123                         // that ant pass previous, but the pheromone will decrease with parameter "rho"
124                         pheromone[j][k] = rho * newpheromone[j][k];
125                     }
126                     else{
127                         // If it is the first ant of the colony every path is unknown
128                         // So the pheromone is random.
129                         pheromone[j][k] = ((double)(rand()%100000)) /(double)100000 ;
130                         newpheromone[j][k] = pheromone[j][k] ;
131                     }
132                 }
133             }
```

Place ant on a random cities , so the first city is the random choose

```
134             //Place the m ants on n random cities
135             int first = rand()%cities;
136             int nextCity = 0,city = first,TotalDistance = 0; // record the total distance;
```

Then every ant visit all the city by reference pheromone matrix.

First I calculate the total Numerator of

$$P_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{j\epsilon S_p}\tau_{ij}^{\alpha}\eta_{ij}^{\beta}} & if\ j\epsilon S_p \\ 0 \end{cases}$$

Then I record the Numerator in our probability 1D array.

```
//visit all the city
for(int j=0;j<cities-1;j++){
    visit[city] = true;
    double total = 0;        // total probability
    //first initialize every probability.
    for(int k=0;k<cities;k++)  prob[k] = 0;

    /******calculate every Numerator of every city**********************/
    for(int k=0;k<cities;k++){
        if(!visit[k]){
            double distR = 1/(double)dist[city][k];
            double child = pow(pheromone[city][k],alpha)*pow(distR,beta);
            prob[k] = child;
            total += prob[k];
        }
    }
```

Second, I determine our randnum will locate on which range of cumulative probability. Because I have ignore those visited cities. When I find it will be the next city the ant will go. So I can add to variable TotalDistance, and record next[city].

```
/******calculate nextCity by cumulative probability***************/
while(1){
    double randnum = ( (double)(rand()%100000)) /(double)100000 ;
    double cumprob = 0; // cumulative probability
    bool find =false;
    for(int k=0;k<cities;k++){
       if(!visit[k]){
           double a = prob[k]/(double)(total);
           cumprob += a;
           if(cumprob >= randnum) {
              nextCity =  k;
              TotalDistance += dist[city][nextCity];
              next[city] = nextCity;
              find = true;
              break;
           }
       }
    }
    if(find) break;
}
city = nextCity;
/******calculate nextCity by cumulative probability***************/
```

After number of cities - 1 turn, the ant have visited all the city.

In the last I will add the distance between the last city and the first city,so the ant can back to source.

Step3 : Update the pheromone matrix

Calculate the delta tao. Then add the delta tao to every path that the ant visited, which means every ant will bring its delta tao to the pheromone matrix.

$$\Delta\tau_{ij}^{k} = \begin{cases} {}^{Q}/_{L_k} & if\ ant\ k\ uses\ edge\ (i,j)in\ its\ tour, \\ 0 & otherwise \end{cases}$$

Where Q is a constant and L_k is the length of the tour constructed by an ant k.

```
177             //Add the distance when back to the source city
178             TotalDistance += dist[nextCity][first];
179             visit[nextCity] = true;
180             double deltatao = (double)Q/(double)TotalDistance; // cacluate the deltatao.
181             city = first;
182
183             for(int j=0;j<cities-1;j++){
184                 // add the deltatao to every path that the ant visited.
185                 //every ant will bring its deltatao to the pheromone matrix
186                 newpheromone[city][next[city]] += deltatao;
187                 newpheromone[next[city]][city] += deltatao;
188                 city = next[city];
189             }
```

Step 4: share the best tour of pheromone matrix every five ant in colony

Critical section (inter)

Because the process of updating global_best_between_colonies will cause many thread update the global_best_between_colonies at the same time, so I add a critical and name it inter(區別 another critical section(outer) in the last turn)

After every thread check if itself is the best answer I can update the pheromone matrix (Add a barrier)

Case1:

If the thread is the best answer,update the global pheromone matrix

Case2 :

If the thread is not the best answer

Get the best **global pheromone matrix** and use it to update the its **local pheromone matrix** after the best thread update the global pheromone matrix.

So I add the barrier between them.

```
192             if(i%5 == 0){   // share the best tour of pheromone matrix every five ant in colony;
193                 #pragma omp barrier
194                 // find which one is the best tour of pheromone matrix
195                 #pragma omp critical(inter)
196                 {
197                     // update the global best shortest path between every thread
198                     if(local_best_in_every_colony < global_best_between_clonies){
199                         global_best_between_clonies = local_best_in_every_colony;
200                     }
201                 }
202                 #pragma omp barrier
203                 // If my local best shortes path is the global shortest path,
204                 // then my pheromone matrix is the best
205                 if(global_best_between_clonies == local_best_in_every_colony){//
206                     for(int j = 0;j <cities;j++){
207                         for(int k = 0;k <cities;k++){
208                             globalpheromone[j][k] = newpheromone[j][k];
209                         }
210                     }
211                 }
212                 #pragma omp barrier
213                 // If my local best shortes path is not the global shortest path,
214                 // then my pheromone matrix is the best
215                 if(global_best_between_clonies != local_best_in_every_colony){
216                     for(int j = 0;j <cities;j++){
217                         for(int k = 0;k <cities;k++){
218                             newpheromone[j][k] = globalpheromone[j][k];
219                         }
220                     }
221                 }
222             }
```

Step 5 : After 每個 thread 都跑一整個蟻巢

將每 core 由不同 thread 得到的 global_best_between_colonies 答案傳給 root core(myid == 0),去比較哪個 core 得到的總距離最短後。由 root core 印出答案。

```
232     //the shortest path of every core
233     printf("The shortest path of core%d is %d\n",myid,global_best_between_clonies);
234
235     MPI_Barrier(MPI_COMM_WORLD);
236
237     if(myid == 0){
238        int temp = 0;
239        for(int i = 1; i < numprocs;i++){
240            MPI_Recv(&temp     ,1,MPI_INT,i,0,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
241            if(temp < global_best_between_clonies) global_best_between_clonies = temp;
242        }
243     }
244     else{
245        MPI_Send(&global_best_between_clonies, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
246     }
247
248     if (myid == 0){ //Printing the best tour
249         printf("The shortest path is %d\n",global_best_between_clonies);
250     }
251     MPI_Barrier(MPI_COMM_WORLD);
252
253     free2dint(&dist);
254     MPI_Finalize();
255     return 0;
```

問題 2 : Analysis on your result

Analysis the four parameter :

Parameter 1- Q : Pheromone Intensity

在調整參數的過程，我發現 constant Q 很重要，一開始我的 Q 設定的很小，所以跑出來的結果都與最優解插了快要一倍。後經過觀察常數 Q 在整個決策的過程所佔的腳色 : 是用來除 螞蟻在經過每座城市後又回到原點的總距離。 而這個除完的結果 delta tao 會加到 pheromone matrix 中， 因此如果要讓 pheromone matrix 真的會因為這次經過的結果改變很大時，Q 就要設定接近或是大一點。才會讓 pheromone 大概在 1 左右的範圍，因為起初我 pheromone 矩陣的初始化(在沒有任何一隻螞蟻經過時)，pheromone 濃度接界在 0-1 之間，也因此，我希望增加的 delta tao 不要太大幅度影響原先的 pheromone 濃度，同時也更不要完全不影響。

Q 為 total pheromone 影響 ant colony algorithm 的收斂速度。太大，則 pheromone concentration 高度集中，會導致收斂到區域最佳解。太小則會降低收斂速度(也就是要跑很多次才跑得出結果)

Parameter 2 - Alpha：(pheromone 濃度變數) Information Elicitation Factor

pheromone 濃度變數 represents the relative importance of the pheromone, reflects the importance of the accumulation of the pheromone with regard to the ants' path selection.

當 alpha 為 0 時，螞蟻會完全根據 cities 的路徑去做選擇。但因為 pheromone 濃度可能為小數點，所以如果 alpha 太大，且當前費洛蒙濃度小於 1，反而會讓 pheromone 濃度越乘越小，反倒沒有達到指引下個螞蟻的用途。而如果 alpha 太大，當前費洛蒙濃度大於 1， 螞蟻會傾向去選擇前面螞蟻選擇的相同路徑，如此一來導致 stronger cooperation among the ants. 雖然這加速了 ant colony algorithm 的收斂速度，但有可能他只是會掉路區域最加解，而降低找到全域最佳解的可能性。

Parameter 3 - Beta : (能見度) expected heuristic factor

Beta relative importance of the visibility. It reflects the importance of the heuristic information with regard to the path selection.

當 beta 為 0 時，螞蟻則完全根據費洛蒙濃度做判斷，這會讓螞蟻規劃的路徑快速收斂，而不是還會適當的觀察 cities 間的距離，如此會很難達到最優。如果 beta 很大，那他將 ant colony algorithm 將趨近於貪婪演算法，他只會觀察當前最近的城市，這可能為區域最佳解但可能不是全域最佳解。

Parameter 4 - pheromone 揮發係數(pheromone evaporation coefficient)

代表前一隻螞蟻留下的資訊會有幾成留給下一隻螞蟻 ,其實也意味著螞蟻間的交互作用力。這個值通常介於 0-1，這樣避免 pheromone 濃度無限的累加到無窮大。而這也是為什麼 pheromone 揮發係數不能太大的原因，雖然會增加 ant colony algorithm 全局探索的能力，但會降低收斂速度。如果太小 ant colony algorithm 全局探索的能力會被降低。

Reference : https://www.hindawi.com/journals/mpe/2016/6469721/

Result:

GR17 is a set of 17 cities, from TSPLIB.

```
F74109016@sivslab-pn1:~$ mpicc -o problem6.exe problem6.c -fopenmp -lm
F74109016@sivslab-pn1:~$ mpiexec -np 5 ./problem6.exe gr17_d.txt
filename = gr17_d.txt
number of ant of colony : 40
number of thread : 10
cities : 17
alpha = 2
beta = 5
rho = 0.800000
Q = 2000
The shortest path of core2 is 2159
The shortest path of core1 is 2159
The shortest path of core0 is 2159
The shortest path of core3 is 2159
The shortest path of core4 is 2095
The shortest path is 2095
```

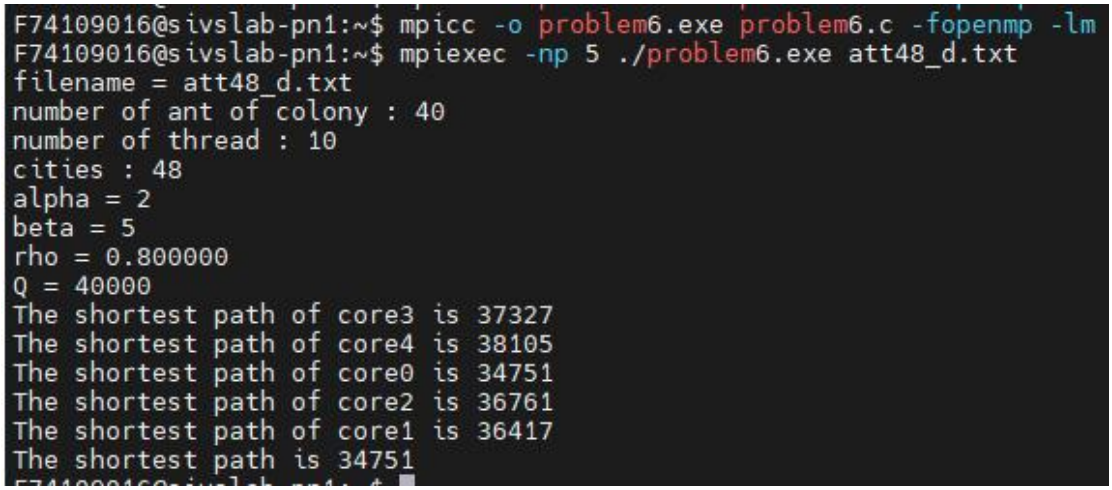
FRI26 is a set of 26 cities

```
F74109016@sivslab-pn1:~$ mpicc -o problem6.exe problem6.c -fopenmp -lm
F74109016@sivslab-pn1:~$ mpiexec -np 5 ./problem6.exe fri26_d.txt
filename = fri26_d.txt
number of ant of colony : 40
number of thread : 10
cities : 26
alpha = 2
beta = 5
rho = 0.800000
Q = 2000
The shortest path of core1 is 965
The shortest path of core4 is 963
The shortest path of core0 is 955
The shortest path of core2 is 937
The shortest path of core3 is 946
The shortest path is 937
```

DANTZIG42 is a set of 42 cities

```
F74109016@sivslab-pn1:~$ mpiexec -np 5 ./problem6.exe dantzig42_d.txt
filename = dantzig42_d.txt
number of ant of colony : 40
number of thread : 10
cities : 42
alpha = 2
beta = 4
rho = 0.700000
Q = 1000
The shortest path of core1 is 753
The shortest path of core2 is 762
The shortest path of core3 is 796
The shortest path of core4 is 792
The shortest path of core0 is 737
The shortest path is 737
F74109016@sivslab-pn1:~$
```

ATT48 is a set of 48 cities (US state capitals) from TSPLIB



問題 3 : Any difficulties?

這個作業我遇到三大困難的地方:

困難 1 : 看不懂助教給的 hackmd

經過上網查資料跟看 youtube 影片，我才知道原來 ant of colony 是這樣實現的，這樣我才有如何撰寫的概念。再來我把影片的距離與費洛蒙濃度，甚麼參數都自己手動算一次確定吻合後，演算法要怎麼寫我就會了。

Reference : https://youtu.be/IP4Fe_flXeU

困難 2 : 不清楚要平行化哪些部分。

我想很久都不知道 MPI 與 Open MP 各自要怎麼平行，謝謝同學的提問，助教與教授的回答讓我清楚方向。

困難 3 : 記憶體問題

在寫完六個作業，我最常遇到的問題就是 Segmentation fault。常常出現的 error signal 6 9 11。我一看到就大概有感覺是哪邊有錯了。

signal 6 就是自己寫錯，沒有初始化陣列，獲釋放宣告的一大堆指標，或是 for 迴圈超出自己設的範圍。

Signal 9 就是 MPI 寫錯了導致某些 process 直接掛掉，那我就要再去思考平行的問題了。

Signal 11 就是出大事情了，我可能整個程式都要非常仔細的檢查了。

寫多平行的作業真的很好玩，因為都不知道自己到底哪裡錯，只能經過一次次 signal 錯誤修正得到的教訓慢慢變厲害。雖然每次 debug 的過程我真的都遇到非常多的困難。

為了讓自己壓力比較小，我這次先把完全沒有平行的 code 寫好，先去給他 run 很少的 city 數量，確保能夠在多次後得到最佳解，我再加上 open MP，

來讓很多 thread 同時去平行做事，再交換解。最後確保這個可以後我又加上 MPI 來讓很多 core 去跑。雖然即便這樣實做，每次多做一個平行，又會多了一堆 bug 要處理完。但 de 完 bug 的感覺真的很快樂。

問題 4 : (optional) Feedback to TAs

作業很有趣，當我在了解這個演算法怎麼運作時，發現前人真的好偉大，能夠想出這麼巧秒的計算公式，而且每個變數都真的很有意義，也很扣合實際的例子，最好玩的是調整參數的過程，有時候調一下加一點就很接近答案，但再加又會遠離答案 ,要試過很多遍 ,通過這個作業 ,我也更理解 MPI 為 core 之間的平行，而 Open MP 為 thread 間的平行。雖然都是平行，但兩個底層平行的東西不一樣，這個觀念我認為對我是最有幫助的，因為這個作業激發我很多以前沒想通的概念 ,這個整合兩個實作方法的作業讓我融會貫通了。寫完這次作業真的得到很多成就感謝謝助教用心出作業。