Problema Viajante MPI

Etapas de la metodología de Foster:

1. Particionado o descomposición en tareas de grano fino:

   Para este apartado hemos optado por descomponer las tareas de los distintos procesos en tours, por lo que cada tarea se corresponde con un tour del tipo tour_t en nuestro programa que son con los que vamos a trabajar.

2. Comunicaciones:

   Para las comunicaciones hemos decidido hacer uso de las primitivas MPI_Bcast , MPI_Send y MPI_Recv.

   La primitiva MPI_Bcast la hemos utilizado tanto para difundir el valor de n (el tamaño de la matriz) como para difundir el digraph a todos los procesos.

   El MPI_Send lo hemos utilizado para difundir el número de tours que le toca a cada proceso, los datos de los tours que le hemos asignado y para una vez que cada proceso tiene su besttour enviárselo al proceso 0 para que se quede con el mejor global.

   El MPI_Recv lo hemos utilizado para recibir todos los datos que hemos mandado a través de los MPI_Send que hemos explicado anteriormente.

   No hemos utilizado la primitiva MPI_Scatterv ya que después de muchos intentos no fuimos capaces de conseguir que el programa funcionase con ella y decidimos utilizar MPI_Send y MPI_Recv para poder completar una versión funcional del programa. El mismo caso nos ha ocurrido con la primitiva MPI_Allreduce.

3. Agregación o aglomeración de tareas:

   En este caso una vez que tenemos todas las tareas repartidas a cada uno de los procesos que les tocan, cada proceso va a trabajar con ese conjunto de tareas y va a obtener un besttour de todas ellas.

4. Asignación de tareas a los procesadores:

   Para esta etapa de las 4 posibles opciones que teníamos para elegir nosotros hemos seleccionado las siguientes:

   - A la hora de repartir los tours a los procesadores hemos escogido la opción de ejecutar el programa secuencial hasta que, tras una iteración completa por el bucle *while*, haya los mismos o más recorridos parciales en la pila que procesos.

     Una vez que tenemos esto calculamos cuantos tours le tenemos que asignar a cada proceso y se los enviamos para que empiecen a trabajar con ellos.

     Hemos elegido esta opción porque creemos que es más sencilla de implementar en nuestro código y creemos que es más fácil de recorrer el árbol utilizando el bucle *while* que ya tenemos en lugar de tener que recorrerlo por niveles y ver si tenemos suficientes nodos.

- Para hacer el cálculo del mejor recorrido (besttour) lo que hemos elegido es que cada uno de los procesos trabaje con sus tours asignados y calcule su besttour y una vez que acabe, este besttour se le manda al proceso 0 que es el que calcula el besttour global e imprime sus datos.

  Hemos escogido esta opción porque creemos que es más sencillo que cada uno de los procesos trabaje en su besttour y una vez que lo tenga lo mande al proceso 0 en lugar de utilizar una variable que sea compartida por todos ya que esto puede dar problemas a la hora de realizar las modificaciones en ella porque se pueden dar situaciones en las que el acceso nos se realice de manera correcta. Además suponemos que utilizar la otra opción supondría un mayor coste de tiempo y recursos debido a las comunicaciones que deberíamos hacer mientras se está ejecutando el algoritmo.