Sistemas Distribuidos
Tarea de Programación
Traveling Salesman Problem – TSP

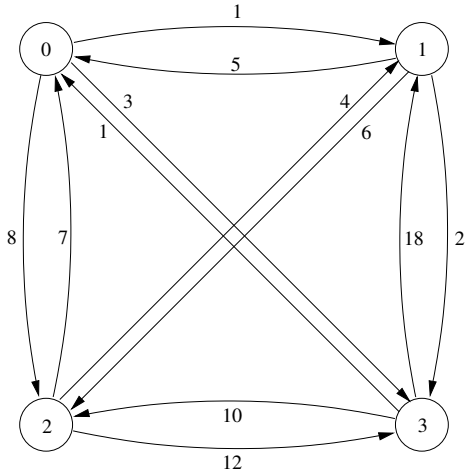
Entrega: A más tardar el martes 13 de mayo de 2014, a las 15:00 horas.

El vendedor viajero

El problema del vendedor viajero, o TSP, es un problema de minimización de caminos sobre un grafo dirigido. Un vendedor debe recorrer un grupo de ciudades, pasando por cada ciudad exactamente una vez, partiendo de un punto y volviendo, al finalizar, al mismo punto de origen. Los vértices del grafo representan las ciudades, y las aristas representan los caminos entre las ciudades, donde el peso determina algún costo asociado, usualmente la distancia.

Para simplificar el problema se supondrá que el grafo se encuentra completamente conectado, en ambas direcciones para cada par de vértices. Esto quiere decir que entre cada par de ciudades existe un camino de ida y uno de vuelta, aunque los costos en cada dirección no necesariamente son iguales.

La representación del grafo con las ciudades, caminos y distancias, se hace empleando una matriz de adyacencia D de tamaño $n \times n$, donde n es el número de ciudades involucradas. La posición (i, j) de la matriz representa la distancia que debe recorrerse para ir de la ciudad i a la ciudad j. La diagonal de la matriz es cero, es decir, $D[i, i] = 0$ para toda ciudad i. Se supondrá que el vendedor inicia y termina su recorrido en la ciudad identificada con 0. A continuación se muestra un ejemplo para $n = 4$ ciudades.



	0	1	2	3
0	0	1	8	3
1	5	0	6	2
2	7	4	0	12
3	1	18	10	0

El problema consiste en determinar una ruta para el vendedor que minimice la distancia total que se debe recorrer. El resultado se almacenará en un arreglo R, de dimensión n, que contiene una permutación de los enteros de 0 a $n-1$, tal que la suma de las distancias entre cada par de ciudades adyacentes sumado a la distancia entre la última ciudad y la primera ciudad en la permutación, sea mínimo.

Para n ciudades existen $(n-1)!$ rutas diferentes que empiezan y terminan en la ciudad 0. A no ser que n sea muy pequeño, este número es muy grande. Para resolver el problema en forma exacta se requiere reducir la cantidad de computación necesaria, y utilizar paralelismo para acelerar esta computación. Para lograr esto, se emplearán trabajadores replicados, una bolsa de trabajos, y un algoritmo *branch-and-bound*, muy útil en problemas de búsqueda combinatorial.

Task bag

Esta estrategia permite paralelizar cualquier algoritmo de tipo *divide-and-conquer*, en que los subproblemas sean independientes.

La idea es tener una bolsa de tareas, en la que inicialmente se coloca una única tarea con el problema global, o con una primera descomposición de éste. Múltiples procesos trabajadores toman tareas de la bolsa y las procesan, a menudo generando nuevas tareas que corresponden a subproblemas. Las nuevas tareas son colocadas de nuevo en la bolsa de tareas. La computación finaliza cuando todas las tareas han sido procesadas, es decir, cuando la bolsa está vacía, y los procesos no tiene trabajo pendiente.

Una forma de implementar este mecanismo es empleando un proceso administrador y varios procesos trabajadores. El administrador genera la primera descomposición y entrega subproblemas de la bolsa a los trabajadores cuando ellos los demandan. Además, cuando los trabajadores generan nuevos subproblemas, el administrador los coloca de nuevo en la bolsa para ser asignados posteriormente. Cuando el administrador se da cuenta de que no existen más problemas en la bolsa de tareas y que los trabajadores están ociosos, la computación finaliza.

Branch-and-bound

Suponiendo que $longMin$ es la longitud de la mejor ruta completa encontrada hasta ahora (la mínima), entonces cualquier ruta, ya sea parcial o total, cuya longitud sea mayor a $longMin$ no puede ser una solución factible para el problema. Por lo tanto, conforme se procesan las rutas es posible descartar aquellas que sean demasiado largas.

Implementación distribuida

Puesto que las rutas son independientes, pueden evaluarse en paralelo.

El administrador mantiene la variable $longMin$ y el arreglo R con la mejor solución hasta el momento. Inicialmente podrían inicializarse estas variables con cualquier ruta en particular y su longitud, por ejemplo, $0, 1, ..., n-1$. Inicialmente la bolsa contendrá $n-1$ tareas, representando las $n-1$ rutas distintas que empiezan en la ciudad 0.

Cada trabajador repetidamente solicitará al administrador una tarea de la bolsa y extenderá el camino con una ciudad que aún no haya sido visitada en esa ruta. Si la ruta no se ha completado y su longitud es menor que la de la mejor ruta hasta el momento, el trabajador envía la nueva ruta y su longitud al administrador para que sea colocada en la bolsa. Si la ruta se completó y su longitud es menor que la mejor ruta hasta el momento, es enviada al administrador para su consideración. Si no ocurre ninguna de las dos condiciones anteriores, la nueva ruta es desechada.

El trabajador continúa intentando extender, con nuevas ciudades, la ruta que obtuvo originalmente. Una vez que ha analizado todas las posibilidades de extensión a partir de la ruta original, solicita una nueva tarea al administrador.

Cuando el administrador recibe nuevas rutas incompletas, las incluye en la bolsa de tareas para su asignación posterior. Cuando recibe una ruta completa, la compara con la mejor hasta el momento y, si es del caso, actualiza la mejor ruta. De lo contrario, la descarta.

Condiciones de implementación y paralelización

La solución debe implementarse empleando `MPI`, en computadores Linux.

Debe emplearse un proceso administrador que será el encargado de crear 3 procesos trabajadores en otras máquinas. El número de procesos será siempre 4, un administrador y 3 trabajadores.

Las entradas del programa son el valor de n y el nombre de un archivo de texto que contiene $n \times n$ números enteros que representan la matriz de adyacencia para el grafo. Los argumentos deben recibirse por la línea de comandos del programa.

Dado que la matriz es constante, puede replicarse en cada uno de los trabajadores. Sin embargo, la mejor ruta hasta el momento y su longitud deben mantenerse en el administrador, aunque probablemente sea necesario distribuir a los trabajadores repetidamente la longitud mínima `longMin`. Esto puede hacerse al asignar una tarea (usando *piggybacking*), o cada vez que `longMin` cambie.

La ruta óptima y su longitud deben enviarse a la salida estándar del proceso administrador. Los procesos *trabajadores* no hacen uso de entrada y salida.

Forma de trabajo y entrega

En esta tarea se debe trabajar en equipos de dos estudiantes.

La tarea debe venir acompañada de un informe en formato PDF, en el que se describa la solución implementada, en especial en lo que se requiere al diseño de los procesos distribuidos y su interacción.

En caso de que el programa tenga limitaciones, éstas deben ser descritas en el informe, junto con las indicaciones de cómo podrían corregirse.

Tanto el programa fuente como el informe deben entregarse a través de la plataforma EDUCANDUS, antes de la fecha de entrega establecida.