Breve explicación del algoritmo

Introducción

En esta entrega, para poder resolver el problema de asignación cuadrática hemos utilizado dos algoritmos, el Branch&Bound i el Greedy, los dos están BranchBound.java.

Las clases que utiliza nuestro algoritmo:

- Clase Node.java
- Clase NodeComparador.java
- Clase BranchBound.java

La clase “node”, es una clase auxiliar de la clase BranchBound, donde se guardaran las posibles soluciones del nuestro problema. Cada “node” contiene tres atributos: un vector de les asignaciones pendientes, un vector de les asignaciones ya realizadas y el coste de la solución.

La clase NodeComparador, es una clase donde esta implementa un método que ordena los “nodes” según sus costos, de tal forma que el “node” que tenga menor coste será el primero en la cola.

Y la clase BranchBound, es la clase principal del algoritmo, sus atributos son:

- Una cola de prioridad de Nodes con la ordenación NodeComparador
- La mida de la solución
- La matriz de estadísticas (o de flujo)
- La matriz de distancias
- Un “node” con la mejor solución
- El mejor coste

Funcionamiento del algoritmo

Al inicializar un BranchBound, se le pasa dos matrices, la de estadísticas y la de distancia, con estas matrices se inicializará un “node” que tendrá un vector vacío de asignaciones realizadas con la mida de les matrices, un vector con todas las asignaciones pendientes y un coste igual a 0.Este “node” se añadirá a la cola de prioridad. A continuación, se ejecutará algoritmo greedy.

Greedy

El algoritmo durante tres iteraciones asignará de forma aleatoria las asignaciones y después calculará su coste, y si el coste de la solución aleatoria es mejor que el mejor coste, el mejor coste será igual al coste de la solución aleatoria. Al salir del greedy tendremos una muy buena cota para poder "podar" las ramas que ya tengan un coste peor al del greedy.

Cuerpo principal del algoritmo (método solve ())

Este método realiza la principal tarea del algoritmo y dará la solución del problema.

Este método se ejecutará mientas quede algún ""node"" en la cola de prioridad, para cada iteración del bucle se tratara un ""node"" de la cola, este ""node"" será el de menor coste de la cola, y se comprobará si es una posible solución (el coste del ""node"" < el mejor coste).

Si es una solución parcial (aun quedan asignaciones pendientes), se crearan tantos hijos como asignaciones pendientes tenga y para cada uno de los hijos se le asignará una asignación diferente, se calculará su nuevo coste y si el hijo tiene un coste superior al mejor coste no se añadirá a la cola.

En cambio, si la solución es total (no le quedan asignaciones pendientes) y su coste es mejor que el mejor coste hasta el momento, la mejor solución será igual a la solución