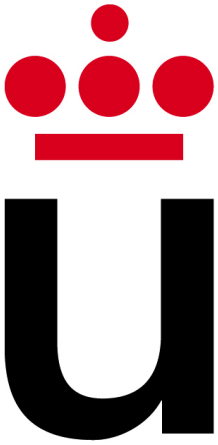

Universidad Rey Juan Carlos

Escuela Técnica Superior Ingeniería Informática

XXXXXX XXXX

TFG

GIS - Carlos Vázquez Sánchez

Móstoles - 11 de febrero de 2017

Índice

1. Introducción

2. Descripción del problema

3. Versiones anteriores del problema y nuevos objetivos

3.1. Versión anterior

Este Trabajo Final de Grado es la continuación del trabajo que llevaron a cabo Diego Ruiz Aguado y Gonzalo Quevedo García en 2012 en sus Proyectos Finales de Carrera, los cuales se apoyaron a su vez en la Tesis Doctoral de Alba Agustín Martín.

A continuación se da una breve descripción del trabajo de Diego Ruiz Aguado y Gonzalo Quevedo García:

1. Se mejoró la BBDD que contenía toda la información del problema, pasando de un modelo no relacional y con redundancias a uno relacional y bien estructurado.

2. Para obtener los datos que necesitaba el problema, se realizó un programa en JAVA que se conectaba a la BBDD y creaba varios ficheros .txt en la que se volcaba toda la información necesaria para el posterior modelado del problema.

3. A continuación, se leían estos ficheros .txt y se creaban las estructuras de datos necesarias (árbol de rutas, vuelos, wapoints, etc).

4. Posteriormente una subrutina en C se encargaba de definir un problema de CIPLEX con la función objetivo y las restricciones necesarias.

5. Finalmente se obtenía la mejor solución del problema.

3.2. Nuevos objetivos

La versión anterior del problema adolecía de un importante inconveniente: no podía salir de los máximos locales, ya que el heurístico que utilizaba para lanzar los vuelos era un algoritmo voraz. Por tanto los objetivos marcados para este TFG han sido los siguientes (ordenados en decreciente prioridad):

1. **Mejorar heurístico:** utilizar heurísticos más elaborados que mejoren la solución del problema.

2. **Desacoplar el programa de CIPLEX:** con la implementación de los nuevos heurísticos no es necesaria la librería de optimización. Se pasará de un sistema clásico de optimización (función objetivo y restricciones) a una estructura de objetos que permitan un manejo óptimo de las esrteucturas de datos durante el heurístico.

3. **Mejorar el sistema de lectura de datos:** el sistema actual crea ficheros .txt que pueden superar las 100.000 lineas. Hay que mejorar este sistema.

4. **Representación gráfica:** aunque no es estrictamente necesaria, si se dispusiera de tiempo suficiente se añadiría una representación gráfica de la solución del problema.

4. Tecnologías utilizadas

Las tecnologías utilizadas han sido las siguientes:

4.1. C++

Aunque se comenzó utilizando C en las primeras versiones del TFG, finalmente se optó por utilizar C++. Esto fue debido a que el cambio a C++ permitió mantener el trabajo ya realizado en C, y posee estructuras de datos ya implementadas como mapas o vectores que reducen en gran medida el tiempo de programación, además de permitir la orientación a objetos.

4.2. MySQL

Al igual que en la versión anterior del proyecto, la BBDD que usamos será la relacional que realizó Diego Ruiz Aguado en el 2012.

4.3. Scripting

Se crearon pequeños scripts que se encargan de importar la BBDD, exportar las tablas en el formato necesario y compilar el proyecto

4.4. HTML y JavaScript

Para la representación gráfica del problema se ha utilizado JS para leer y parsear la información almacenada en un fichero y HTML para su visualización (para la creación del grafo se ha utilizado la librería vis.js).

4.5. Git

Se ha utilizado Git como sistema de versiones, y todo el código se puede descargar y consultar en Github

5. Heurístico

El algoritmo se resume en el siguiente diagrama:
Los pasos son:

1. **Lanzamientos iniciales:** Lanzar vuelos solo con la mejor solución (solución inicial)
2. **Intercambio de vuelos:** solo con las soluciones iniciales (sin retardos ni rutas alternativas), intentamos cambiar un vuelo ok por 2 O MAS vuelos que fueron cancelados. para ello:
   a) Vemos si algún vuelo ok bloquea más de 2 vuelos cancelados. Comparamos los sectores que bloquea cada uno en un momento de tiempo t. Obtenemos una tupla <vueloOK,candidatos vuelos cancelados>
   b) De todos los candidatos, escogemos una combinación (al azar) de vuelos que puedan ser una solución válida respecto al número de sectores(un vuelo ok que bloquea 4 sectores necesitará como mucho 2 vuelos de 2 sectores cada uno)
   c) Con esa selección de candidatos, vemos si lanzando esos vuelos y cancelando el anterior ok la solución es válida.
   d) Si es válida, hacemos el cambio. Si no, nada.
3. **Vuelos con retrasos**: los vuelos pueden retrasarse, pero no optar por las rutas alternativas.
4. **Vuelos sin restricciones**: pueden coger rutas alternativas y retrasarse.

6. Resultados experimentales

A continuación se detallan los resultados obtenidos con distintos parámetros del problema

7. Conclusiones

8. Bibliografía