Trabajo de Fin de Grado

Editor de Turnos Automático para Pilotos y Aviones (ETAPA)

Auto Shift editor for Pilots and Aircraft

Dailos Herrera Bencomo

La Laguna, 1 de septiembre de 2016

D. **Juan José Salazar González**, con N.I.F. 43.356.435-D profesor Catedrático de Universidad adscrito al Departamento de Matemáticas, Estadística e Investigación Operativa de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que la presente memoria titulada:

"Editor de Turnos Automático para Pilotos y Aviones (ETAPA)."

ha sido realizada bajo su dirección por D. **Dailos Herrera Bencomo**, con N.I.F. 78.700.555-Y.

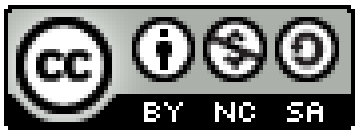
Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firma la presente en La Laguna a 1 de septiembre de 2016.

Agradecimientos

Esta memoria significa el culmen de días, meses y años de estudio, formación, trabajo y experiencias para lograr el título de Ingeniero Superior. Por ello, me gustaría agradecer en primera instancia, a mi director de Trabajo de Fin de Grado, Juan José Salazar González, por su gran ayuda, preocupación, apoyo y constancia que siempre me ha ofrecido.

A mi familia, amigos cercanos, compañeros de trabajo y compañeros de estudios que siempre han estado apoyándome, aportando su grano de arena para lograr esta meta tan importante para mí.

Licencia



Resumen

El objetivo de este trabajo ha sido diseñar una herramienta informática para la gestión de vuelos de una compañía aérea en un día con incidencias. Asumimos dada una planificación inicial que no se puede ejecutar porque suceden imprevistos que obligan a retrasar o cancelar vuelos previstos. Estas alteraciones deben modificar tanto la planificación de pilotos como de aviones. La herramienta está implementada en Java, y contiene tanto un algoritmo heurístico como elementos para edición manual.

Palabras clave: *Planificación, gestión de vuelos, algoritmo heurístico, logística.*

Abstract

The objective of this work has been to design a software tool for managing an airline flight in one day with incidents. Assuming an initial planning that can't run because unforeseen happen that force delay or cancel flights scheduled. These alterations must modify both planning pilots and aircraft. The tool is implemented in Java, and contains both a heuristic algorithm as elements for manual editing.

Keywords: *Planning, flight management, heuristic algorithm, logistics.*

Índice General

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

Este capítulo especifica el objeto y contenido del Trabajo Fin de Grado. Está dividido en tres apartados donde se expone la descripción del mismo, su análisis, y la tecnología y herramientas utilizadas para llevarlo a cabo.

1.1 Descripción

Este Trabajo Fin de Grado intenta cubrir una necesidad primordial en cualquier línea aérea. Es la necesidad de conseguir crear una herramienta (software) interactiva para la gestión eficiente de vuelos de una compañía aérea en un día con incidencias.

Para ello, a partir de una solución con una planificación dada, dicha herramienta deberá contemplar la posibilidad de incluir o modificar la misma incluyendo los posibles problemas que surgen en el día a día como pueden ser, por ejemplo, retrasos o cancelaciones de vuelos de forma sencilla para el usuario tanto a nivel visual como de usabilidad.

A partir de este nuevo escenario, la herramienta debe lograr obtener una solución con una planificación de la forma más rápida y óptima posible que contenga los cambios efectuados y ésta, le sea mostrada al usuario.

1.2 Análisis

Como paso inicial, una vez entendido el cometido que se debe llevar a cabo en el Trabajo Fin de Grado, se debe realizar el análisis de la herramienta a desarrollar y todo lo relacionado con la misma.

Esta herramienta surge por la necesidad que tiene el usuario de mejorar en usabilidad, tiempo y forma el proceso actual que realiza para incluir alteraciones sobre una solución (planificación) dada y obtener una nueva solución óptima. Esta herramienta debe tener 3 puntos fundamentales:

- Visualización: Deber ser una herramienta estable, sencilla y cómoda de forma visual para que su uso sea correcto y que aporte, con un solo vistazo, la mayor información posible.
- Interacción: Debe ser una herramienta totalmente interactiva con el usuario para que éste pueda realizar las acciones que crea oportunas de una forma rápida y entendible.
- Resolución: Deber ser una herramienta capaz de aportar una solución rápida, óptima y coherente a las modificaciones planteadas sobre una solución inicial.

Se analiza el proceso actual que viene realizando el usuario, del cual se obtiene que se parte de una solución del problema para un día en concreto. Normalmente se planifica 6 meses antes aproximadamente usando sólo horarios de vuelos y no condiciones climatológicas ni averías ni cualquier otra incidencia o problema. Estas planificaciones se calculaban a mano con anterioridad a la inclusión de una aplicación que es quien aporta la solución (planificación) resolviendo el problema de optimización.

Esta aplicación sitúa los aproximadamente, 100-150 vuelos de cada día en las 18 líneas (aviones) y les da colores (pilotos) de forma que se cumpla una serie de requisitos/restricciones:

- No más de 8 vuelos para un mismo piloto.
- Cada piloto comienza en Tenerife Norte (T) o en Las Palmas (L) debe terminar en su misma isla
- Los aviones deben cambiar de isla para una revisión cada dos días (se realiza en Las Palmas).
- Los vuelos comienzas como mínimo a las 06:00 horas (ningún vuelo debe comenzar antes)
- Los aeropuertos cierran a las 23:00 horas (ningún vuelo debe superar esa hora de llegada)
- Entre vuelo y vuelo de un mismo avión, al menos 20 minutos (preferiblemente 30).

Actualmente, el usuario (se asume que es un técnico especialista y es conocedor de las restricciones anteriormente mencionadas, entre otras), hace uso de un Excel para realizar las modificaciones (retrasos, cancelaciones, etc.) de forma manual. Este proceso no es nada cómodo ni intuitivo. Lo ideal (y lo que se ha pretendido) es encontrar una herramienta que aporte automatización y sea mucho más usable y robusta para resolver satisfactoriamente el problema basándose en el uso de un algoritmo eficaz y eficiente.

Se debe conseguir una herramienta muy diáfana, que no sea compleja para su uso y que se base en un proceso sencillo que sea, básicamente, leer una solución, modificarla y obtener la nueva solución, así de sencilla.

La herramienta leerá la información necesaria desde archivos de texto externos, a resaltar los siguientes:

- **<fecha>_vuelosMMMAA.txt**: fichero que contiene la información de los vuelos: fecha, compañía, número de vuelo, origen, destino, hora salida y hora de llegada.
- **config_<fecha>.txt**: fichero que contiene la configuración necesaria para llamar a la aplicación externa (optimizador) que se encarga de obtener la solución óptima.
- **<fecha>_salida.txt**: fichero que generalmente, contiene una solución de planificación. Es el esquema de nomenclatura que utiliza el optimizador para devolver la solución encontrada.
- **aviones-operador.txt**: fichero que indica los aviones disponibles y su operador: (N)aysa, (C)anAir y (B)inter Canarias.
- **tabla-actividad.txt**: fichero que indica dada una ruta (secuencia de vuelos) para un piloto, la duración máxima desde que comienza hasta que finaliza. Depende del instante en el que comienza y del número de vuelos que la componga.
- **mantenimiento.txt**: fichero que indica si algún avión está fuera de servicio algunos días.

Las estructuras que almacenan la información necesaria para el funcionamiento de la herramienta se basará en la lectura de la información de los tres primeros ficheros, vuelos, configuración y solución inicial.

La parte visual, como se comentaba anteriormente, será bastante intuitiva con las opciones justas y necesarias para realizar su cometido concreto. Se mostrará la solución inicial en dos dimensiones, porque aunque toda la información que se maneja se basa en una naturaleza tridimensional (tiempo, aviones, pilotos), al ser mostrada en una pantalla sólo disponemos de dos dimensiones: tiempo y aviones, aportando una mayor riqueza informativa haciendo uso de colores y las líneas oblicuas para indicar las rutas de los diferentes pilotos. Habían otras posibilidades de representar la información, tiempo y pilotos en las dos dimensiones de la pantalla (y colores y líneas para aviones) o pilotos y aviones en las dos dimensiones de la pantalla (y colores y líneas para el tiempo) pero se decidió por la indicada al ser más cómoda, útil y fácil de apreciar al ojo humano.

Además también se mostrará información significativa de la solución expresada como puede ser el valor de la misma, el número de aviones, vuelos, pilotos o cambios de aviones de éstos últimos.

El usuario realizará las modificaciones o informará de las incidencias de forma rápida y clara haciendo uso de las siguientes posibilidades:

- Adelanto de un vuelo.
- Retraso de un vuelo.
- Cancelación de un vuelo.
- Creación de un vuelo nuevo.

Una vez se hayan definido las incidencias, la herramienta hará una llamada a la aplicación externa que es la encargada de facilitar la solución óptima a nuestra herramienta para que sea mostrada al usuario.

Esta aplicación externa es uno de los aspectos importantes de este Trabajo Fin de Grado ya que ambas aplicaciones, la nueva herramienta y ésta, deben estar integradas y depende una de la otra para un correcto funcionamiento y rápida respuesta al usuario.

La aplicación externa se desarrolló en 2014 utilizando un algoritmo heurístico para obtener una solución eficiente. En 2016 se tiene una versión mejor, que en ocasiones genera una solución óptima, basada en la librería gratuita SCIP, aunque también puede usar la librería comercial CPLEX. La nueva versión reduce notablemente el tiempo de proceso y aumenta la calidad

de la solución (en muchos casos termina con la solución óptima). Para la obtención de una solución óptima tiene 30 minutos de tope de cálculo.

Esto es francamente bueno, puesto que dicho motor resuelve problemas óptimamente, siendo mejor que un heurístico, ya que los heurísticos quizás en ocasiones dan malas soluciones o incluso no dan soluciones. Cuando un óptimo la encuentra seguro si existe y es por ello, que nuestra herramienta, la utiliza como una dll, por lo explicado hasta ahora y además que el tiempo de proceso será escaso en leer y pintar la nueva solución.

Dicho todo esto, no podemos obviar que las incidencias o modificaciones planteadas por el usuario podrían ni ser óptimas, ni ser factibles o simplemente determinarse que no hay solución.

Todas las modificaciones planteadas por el usuario, serán almacenadas o contenida en un log de cambios para poder verificar o comprobar que la solución inicial y final, difieren en los cambios introducidos si forman parte de la nueva solución porque no se han desechado al entrar en conflicto con las restricciones que debe cumplir una solución.

1.3 Tecnología

La tecnología utilizada para el desarrollo del Trabajo Fin de Grado ha sido la siguiente:

- Lenguaje de programación

Java, elegida de entre las 3 alternativas posibles: C, C++ y Java.

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra.

Este ha sido el aspecto más importante para la toma de la decisión del lenguaje a utilizar para el desarrollo de la herramienta.

- Entorno de desarrollo

Eclipse, elegido por recomendación de compañeros y buscando/analizando información de la misma, viendo que es bastante extendido y recomendado su uso.

Eclipse es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

- Control de versiones

Git, elegido para el control de versiones y así no perder información con los diferentes cambios o pruebas que surgirán durante el desarrollo del Trabajo Fin de Grado.

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente.

- Alojamiento externo

Bitbucket, elegido porque ya lo he usado y es cómodo para tener de una manera centralizada y accesible todo el código fuente.

Bitbucket es un servicio de alojamiento basado en web, para los proyectos que utilizan el sistema de control de revisiones Mercurial y Git. Ofrece planes comerciales y gratuitos. Se ofrece cuentas gratuitas con un número ilimitado de repositorios privados.

https://bitbucket.org/alu0100036862_DailosHB/tfg_alu0100036862.git

1.4 Estado del arte

Actualmente se sigue investigando y realizando muchos estudios que permitan responder de una forma más rápida, y cada vez, más eficiente y óptima ante los posibles imprevistos que puedan surgir en la gestión de interrupciones de aerolíneas (airline disruption management).

De algunos de esos estudios que se pueden encontrar por internet, podemos observar el impacto que pueden tener las diferentes incidencias que se pueden dar en este negocio tan complejo y sus actores:

Figura 1. **Efectos de las incidencias**.

O también, la relación con el coste:

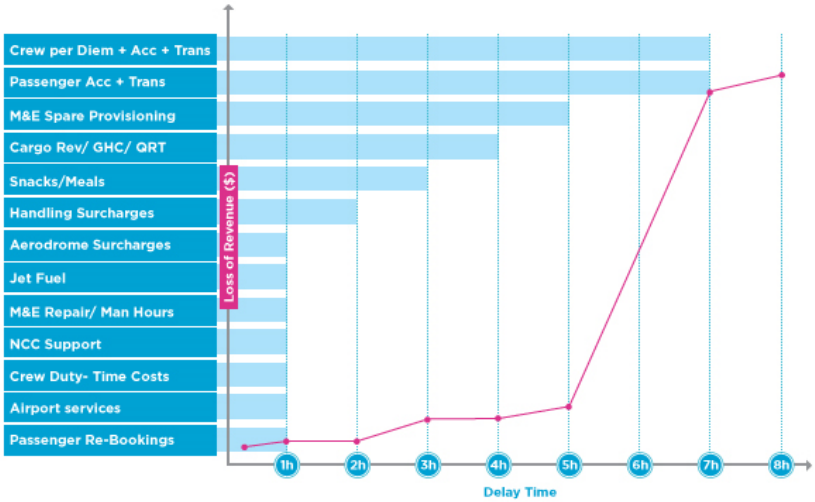


Figura 2. Relación **incidencias-coste**.

Es un campo muy complicado porque aunque la tecnología avanza y cada vez, se presupone, hay menos fallos o problemas, pero éstos no disminuyen de una forma cuantitativa importante debido a múltiples factores como puede ser el mayor uso de las aerolíneas como modo de transporte, el aumento de vuelos que conllevan más planificación tanto de tiempos, como de vuelos, aviones, pilotos, etc. Por ello, aunque se parta de una planificación dada, se debe usar prácticamente a diario, por muy pequeña que sea la compañía/empresa, una herramienta que sea capaz de solventar los problemas derivados del día a día aportando una solución de forma rápida y óptima.

No existen muchas aplicaciones capaces de reestructurar una planificación que haya sido alterada por diferentes incidencias y seguramente tendrán un elevado coste económico. Actualmente, las compañías pequeñas toman decisiones de forma manual, y es por ello que la herramienta a desarrollar en este Trabajo Fin de Grado será de gran utilidad.

Figura 3. **Factores a restructurar**.

Capítulo 2.
Desarrollo de la Herramienta

Este capítulo será el encargado de detallar el proceso del desarrollo de la herramienta en la que se centra este Trabajo Fin de Grado.

2.1 Objetivos

El objetivo principal de este Trabajo fin de Grado diseñar y desarrollar una herramienta informática para la gestión de vuelos de una compañía aérea en un día con incidencias.

Para su consecución se deben acometer una serie de objetivos secundarios, los cuales son:

- Usabilidad: Diseñar una herramienta totalmente intuitiva y a la vez, que su uso sea fácil y sencillo para el usuario.
- Rapidez y eficiencia: La herramienta debe resolver de forma satisfactoria, se obtenga o no solución, el problema planteado de forma eficaz y eficiente.
- Robustez: La herramienta debe ser robusta en cuanto a funcionamiento, lo que conllevará mayor seguridad y confianza al usuario.
- Cubrir necesidades y mejoras: La herramienta deberá cubrir las necesidades que tiene el usuario y, por supuesto, aportar mejoras para que sea de utilidad.
- Uso de aplicación externa: La herramienta deberá apoyarse en el uso del optimizador para la obtención de una solución óptima.
- Multiplataforma: La herramienta debe implementarse en un leguaje multiplataforma para que pueda ser utilizado en cualquier máquina.

2.2 Fases

Antes de enumerar las diferentes fases que se han llevado a cabo para la consecución del Trabajo Fin de Grado es importante indicar que la metodología utilizada para su desarrollo ha sido una metodología ágil llamada **metodología XP** (extreming programming).

Esta metodología se basa en la simplicidad, la comunicación entre el cliente y desarrollador y realimentación del código que se va desarrollando. Aporta una programación organizada y una menor taza de errores, además que es aconsejable para proyectos a corto plazo, como lo es este caso.

Las fases no se distinguen de cualquier otro desarrollo de software ya que son generalizadas y comunes aunque con sus peculiaridades.

- Planificación: En esta fase se han establecido los requisitos del usuario, definido las historias de usuario con el cliente y establecido las restricciones principales de los diferentes procesos.
- Diseño: En esta fase se ha determinado cómo funcionará la herramienta de forma general sin entrar en detalles incorporando consideraciones de la implementación tecnológica.
- Implementación: Se ha traducido el diseño a código atendiendo a estándares de codificación ya creados, para facilitar su comprensión y escalabilidad. Eta es la parte más obvia del trabajo de ingeniería de software y la primera en que se obtienen resultados "tangibles", por ello se desarrollará de forma exhaustiva en el siguiente punto.
- Pruebas: Esta fase no se ha considerado como la última en un proceso secuencial, ya que se han ido realizando pruebas a medidas que se han ido obteniendo pequeños hitos durante la implementación para comprobar el funcionamiento de lo que se va implementando.

2.3 Operatividad

En este apartado se muestra la operatividad y funcionamiento de la herramienta además de detallar los procesos que se llevan a cabo para cada una de las tareas que se realizan.

El funcionamiento está centrado en el objetivo principal de la herramienta, y esquematizado en 3 hitos:

1. Carga de solución inicial.
2. Introducir incidencias.
3. Generación de la nueva solución.

Inicialmente, la aplicación se mostrará con un formulario vacío que contiene el siguiente menú en la parte superior:

Figura 4. **Menú principal** de la herramienta.

La funcionalidad de cada uno de los botones del menú es la siguiente:

- Ayuda: Este botón muestra la información sobre cómo funciona la herramienta.
- Guardar solución: Este botón permite guardar la solución cargada en el formulario (solución inicial o resultante de generar solución editada)
- Seleccionar configuración: Este botón permite seleccionar el fichero de configuración. Este fichero es indispensable para realizar cualquier proceso en la herramienta.
- Seleccionar solución: Este botón permite seleccionar el fichero que contiene la solución con la planificación a cargar.
- Generar solución: Este botón permite generar una nueva solución basándose en la configuración del fichero elegido previamente.
- Cargar solución: Este botón carga la solución o bien desde un fichero o la generada.
- Modo Edición: Este botón permite entrar en el modo edición de la solución para informar las incidencias que se crean oportunas sobre la misma.

- Generar solución: Este botón es el encargado de realizar la llamada al optimizador para generar una nueva solución con la planificación que contenga las incidencias informadas si son viables en la misma.

El botón de ayuda, tal y como su nombre indica, ayuda al usuario de la herramienta mostrando un formulario explicativo de las acciones que acometen cada uno de los botones de la aplicación. El formulario es el siguiente:

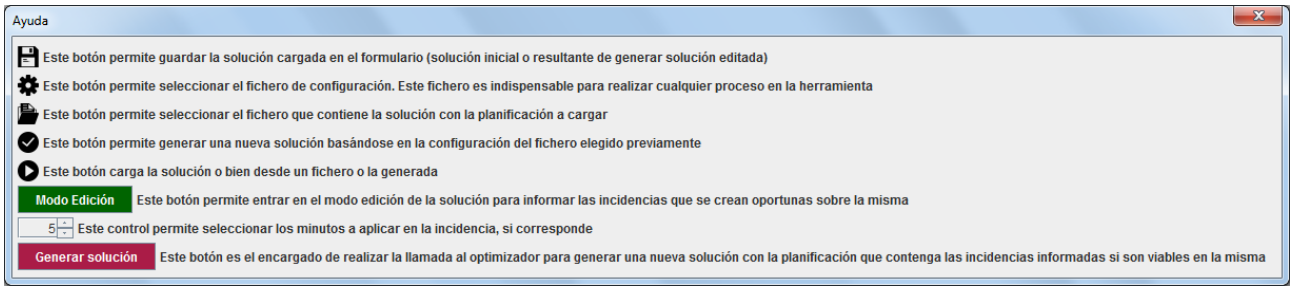


Figura 5. Formulario de **ayuda**.

Una vez explicada la visión inicial de la herramienta, el proceso a realizar por el usuario es bastante sencillo. Se especifica su funcionamiento dividiendo el contenido sus 3 hitos:

2.3.1 Carga de la solución

Siempre se parte de la selección del fichero de configuración ya que éste es necesario tanto para la carga de la estructura de la información necesaria y relativa a los vuelos del día que se pretendan modificar como para el funcionamiento del optimizador.

El fichero debe tener la siguiente nomenclatura: **config_<fecha>.txt**

De su contenido se obtiene la fecha a procesar y el nombre del fichero de vuelos que tendrá la nomenclatura: **<fecha>_vuelosMMMAA.txt**, como norma general. Éste último fichero, debe tener la siguiente estructura (fecha, compañía, vuelo, origen, destino, hora salida y hora llegada):

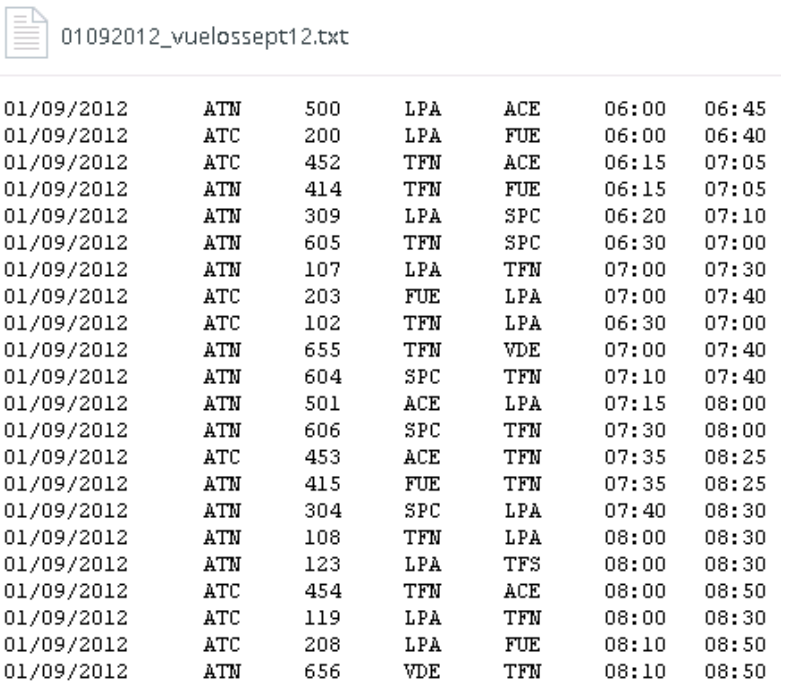


Figura 6. **Fichero de vuelos**.

En este punto, el usuario de la herramienta, tiene dos posibilidades para la carga de la solución con la planificación:

- Seleccionar un fichero de salida (a través del botón indicado anteriormente) que tenga la solución. Este fichero debe tener la nomenclatura: **<fecha>_salida.txt**

- Generar una solución nueva (a través del botón indicado anteriormente). Este proceso puede tardar varios minutos ya que llama al optimizador quien se encargará de dicha tarea.

Una vez se tenga la solución, simplemente pinchamos en cargar, si eligió generar la solución, se cargará automáticamente la misma, y se mostraría algo así:

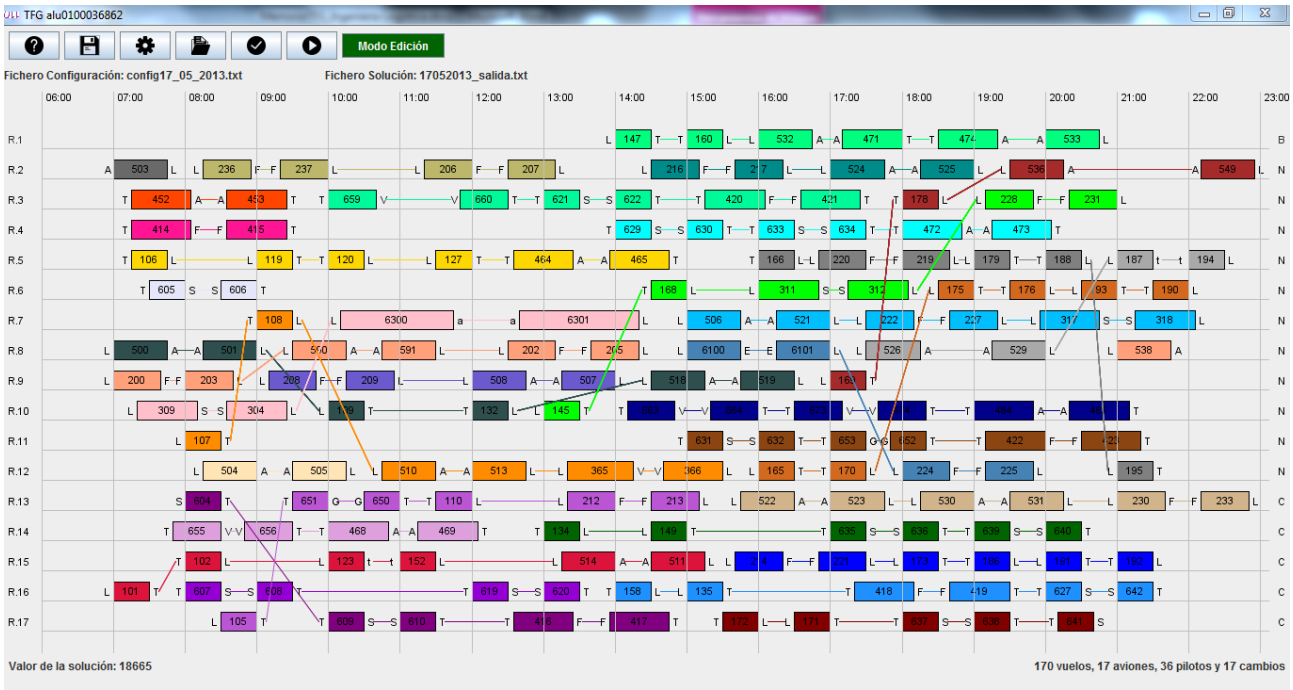


Figura 7. **Solución cargada**.

En este punto podemos ver mucha información significativa de la solución, ya no sólo las líneas aéreas o aviones (R.1, R.2, etc.) en la parte izquierda o la compañía (N, C o B) en la parte derecha sino que también podemos ver en la parte inferior el valor de la solución cargada, el número de vuelos, el número de aviones, el número de pilotos y el número de cambios de avión de los mismos que se muestran en la solución cargada.

Además de lo anterior, está lo más llamativo que son los vuelos.

Figura 8. Ejemplo de **vuelo**.

Éstos están situados en su horario y duración correctos y cada uno de ellos muestra su número de vuelo, aeropuerto de origen y aeropuerto de destino:

SIGLA	AEROPUERTO
L	Aeropuerto de Gran Canaria, España
T	Aeropuerto de Tenerife-Norte, España
t	Aeropuerto de Tenerife-Sur, España
A	Aeropuerto de Lanzarote, España
S	Aeropuerto de La Palma, España
G	Aeropuerto de La Gomera, España
V	Aeropuerto de El Hierro, España
F	Aeropuerto de Fuerteventura, España
E	Aeropuerto de Laayoune, Sahara Occidental
R	Aeropuerto de Marrakech-Menara, Marruecos
f	Aeropuerto de Madeira, Portugal
N	Aeropuerto Internacional de Nouadhibou, Mauritania
C	Aeropuerto Internacional Mohámmed V, Marruecos
a	Aeropuerto de Agadir-Al Massira, Marruecos

Tabla 1. Tabla de **aeropuertos.**

El color de los vuelos representa cada una de los vuelos que tiene un piloto, es decir, que cada color representa a un piloto. También se puede apreciar la unión de los vuelos según el color, indicando que el piloto pasa de un vuelo a otro, incluso cambiando de avión:

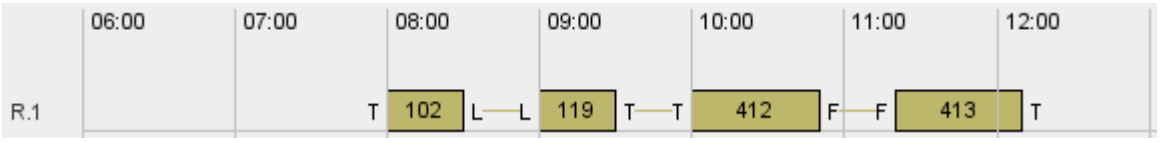


Figura 9. Ejemplo de **ruta de piloto**.

En la figura anterior vemos un ejemplo de ruta de piloto, la cual indica que está formada por 4 vuelos (102, 119, 412 y 413) en el mismo avión: R.1, dispuestos o llevados a cabo en este orden (ver horarios en la parte superior de la imagen) y que comienzan en Tenerife Norte – Las Palmas – Tenerife Norte – Fuerteventura y Tenerife Norte nuevamente.

Un aspecto que hace más amigable la herramienta es la opción de ver la información de un vuelo en cuestión pulsando sobre el vuelo con el botón izquierdo del ratón, con ello, se muestra la siguiente pantalla:

Figura 10. **Información de un vuelo**.

2.3.2 Introducir incidencias

Una vez llegados a este punto, el usuario de la herramienta puede introducir las incidencias que crea oportunas. Existen 4 tipos de incidencias:

COLOR	TIPO INCIDENCIA
	Adelanto de un vuelo
	Retraso de un vuelo
	Cancelación de un vuelo
	Creación de un vuelo nuevo

Tabla 2. Tabla de tipos de **incidencias.**

Para realizar cualquier acción referente a la edición (incidencias) en sí, el usuario dispondrá de un menú pulsando el botón derecho del ratón. Según donde se pulse, sobre un vuelo o en el espacio entre vuelos, o el estado del vuelo, dicho menú variará:

- Sobre un vuelo sin editar.

El menú dará la posibilidad al usuario de **Adelantar** o **Retrasar** o **Cancelar** el vuelo.

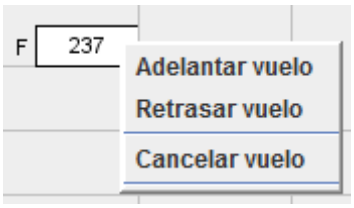


Figura 11. **Opción de menú 1.**

- En el espacio entre vuelos.

El menú dará la posibilidad de **Crear** un vuelo nuevo.

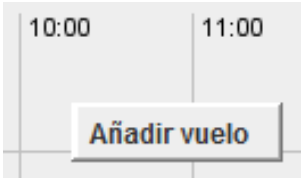


Figura 12. Opción de menú 2.

- Sobre un vuelo editado.

El menú dará la posibilidad de **Eliminar** la edición.

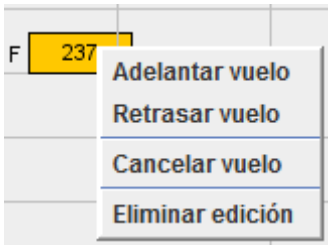


Figura 13. Opción de menú 3.

Nota: Para las incidencias de Adelantar y Retrasar vuelo, hay un control en la parte superior derecha donde se indicarán los minutos a aplicar:

Figura 14. Control establecido para los minutos a aplicar.

El usuario informará las incidencias de la siguiente manera:

1. Adelanto de un vuelo.

El usuario puede adelantar cualquier vuelo, seleccionando el número de minutos a adelantar el vuelo y pulsando sobre el vuelo con el botón derecho, seleccionando la opción correspondiente.

Figura 15. Resultado de la incidencia (adelanto).

2. Retraso de un vuelo.

El usuario puede retrasar cualquier vuelo, seleccionando el número de minutos a retrasar el vuelo y pulsando sobre el vuelo con el botón derecho, seleccionando la opción correspondiente.

Figura 16. Resultado de la incidencia (**retraso**).

3. Cancelación de un vuelo.

El usuario puede retrasar cualquier vuelo, seleccionando el número de minutos a retrasar el vuelo y pulsando sobre el vuelo con el botón derecho, seleccionando la opción correspondiente.

Figura 17. Resultado de la incidencia (**cancelación**).

4. Creación de un vuelo nuevo.

El usuario puede crear un vuelo pulsando sobre el espacio que hay entre vuelos con el botón derecho y debe introducir la información con el número del nuevo vuelo (que no exista), el avión donde situarlo (puede no situarse en él en la nueva solución), la compañía, la hora de salida, la hora de llegada y los aeropuertos de origen y destino, en el formulario que se muestra a continuación:

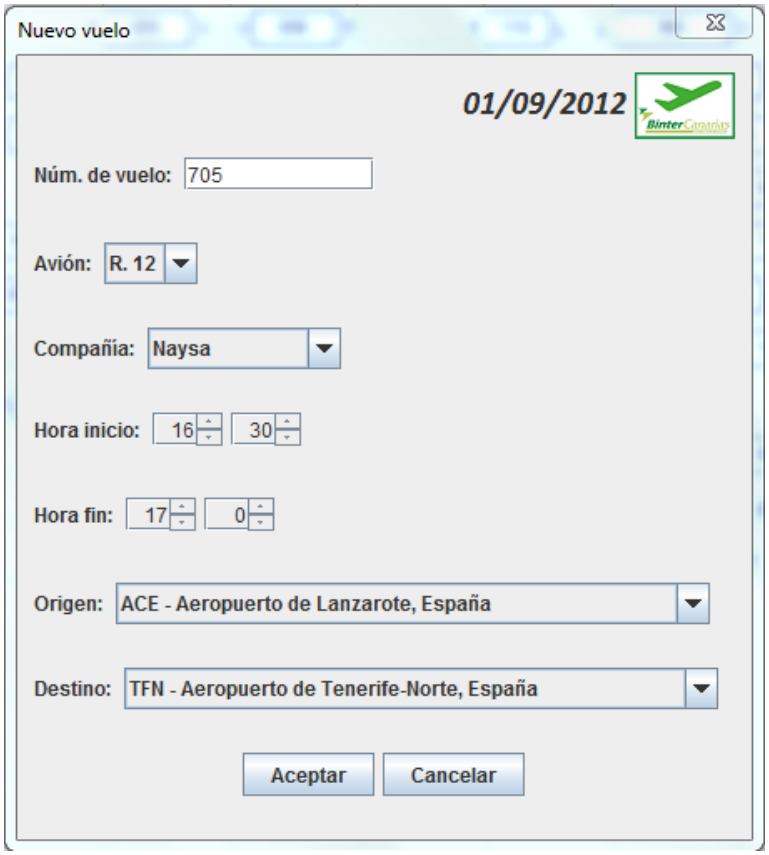


Figura 18. Formulario de **creación** de un vuelo nuevo.

Si todos los datos introducidos están correctos, se crea el nuevo vuelo:

Figura 19. Resultado de la incidencia (**nuevo vuelo**).

2.3.3 Generación de la nueva solución

Para finalizar, está el hito de la generación de la nueva solución.

Este proceso es transparente desde el punto de vista del usuario, el cual, simplemente tiene que pulsar el botón de "Generar solución".

El proceso, lo que hace es modificar el fichero de vuelos incluyendo las incidencias informadas por el usuario, previamente haciendo un backup del fichero original de vuelos y llama al optimizador para que comience a trabajar y encuentre una solución óptima al nuevo problema planteado.

Como bien se indica, el proceso puede durar varios minutos, dependiendo de las incidencias y las casuísticas que éstas provoquen en la planificación, con un máximo de 30 minutos:

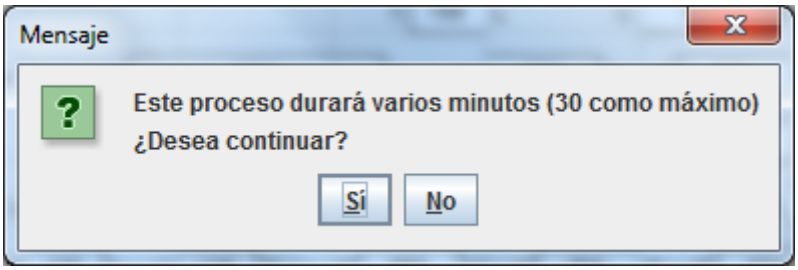


Figura 20. **Llamada al optimizador** (aviso).

Durante la ejecución o llamada al optimizador, la herramienta queda bloqueada.

El resultado del optimizador podrá ser:

- Encuentra solución: El optimizador encuentra solución y crea un nuevo fichero con la salida, el cual es leído por la herramienta y pinta el nuevo escenario con la solución.
- No hay solución: El optimizador encuentra que no hay solución y la herramienta muestra un aviso.
- No se encuentra solución: El optimizador no encuentra solución durante los 30 minutos y la herramienta muestra un aviso.

2.4 Estructura de la herramienta

La herramienta está formada por la siguiente estructura de archivos para completar su implementación:

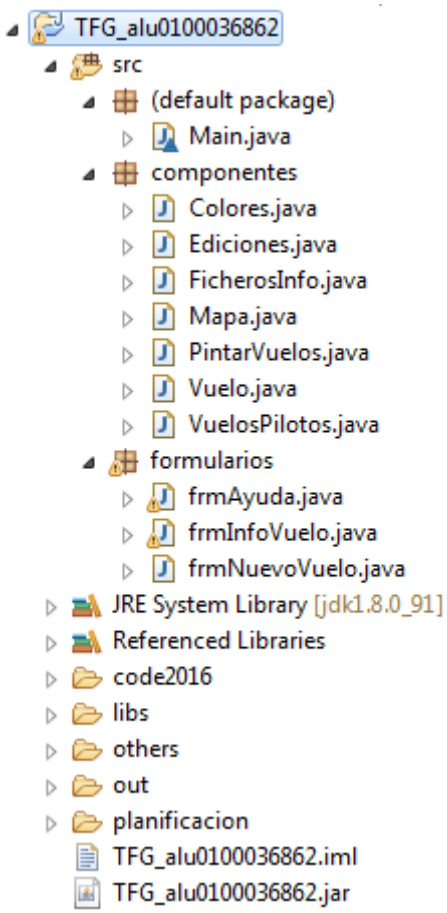


Figura 21. **Estructura de archivos**.

Está compuesta por seis carpetas de las que cabe destacar las dos de ellas:

- **src:** Carpeta que contiene el código fuente implementados, estructurado en 3 paquetes (default, componentes y formularios).
- **code2016:** Carpeta que contiene la implementación del optimizador el cual es llamado desde nuestra herramienta.

Tal y como se ha indicado, el código fuente está en la carpeta src divido en 3 paquetes. Este es el detalle generalizado de los ficheros que contiene cada uno de los paquetes:

1. Default
   - Archivo **Main.java:** Este archivo contiene el programa principal, el cual llama a la clase Mapa.java (ver en el siguiente punto su definición).

2. Componentes
   - Archivo **Colores.java:** Este archivo implementa la clase que almacena la información estática de los diferentes colores que se utilizarán para pintar las rutas de los pilotos.
   - Archivo **Ediciones.java:** Este archivo implementa la clase que almacena las ediciones (incidencias)) que crea el usuario sobre la solución inicial.
   - Archivo **FicherosInfo.java:** Este archivo implementa la clase que se encarga de trabajar con los ficheros y su información.
   - Archivo **Mapa.java:** Este archivo implementa la clase que se encarga del proceso del negocio, es decir, del funcionamiento de la herramienta realizando las acciones y llamadas a los diferentes procesos.
   - Archivo **PintarVuelos.java:** Este archivo implementa la clase que pinta las soluciones.
   - Archivo **Vuelo.java:** Este archivo implementa la clase que almacena la información de los vuelos en general.
   - Archivo **VuelosPilotos.java:** Este archivo implementa la clase que almacena la información de los pilotos y sus los vuelos.

3. Default

   - Archivo **frmAyuda.java:** Este archivo implementa la clase correspondiente al formulario Ayuda, tanto su parte visual como sus acciones.
   - Archivo **frmInfoVuelo.java:** Este archivo implementa la clase correspondiente al formulario que muestra la información de un vuelo, tanto su parte visual como sus acciones.
   - Archivo **frmNuevoVuelo.java:** Este archivo implementa la clase correspondiente al formulario con el que se crea un nuevo vuelo (incidencia), tanto su parte visual como sus acciones y/o comprobaciones.

Capítulo 3.
Resultados

Este capítulo será el encargado de mostrar de forma ilustrativa y numérica el funcionamiento y los tiempos de ejecución respectivamente.

Indicar que la máquina donde se han realizado las pruebas tiene las siguientes características:

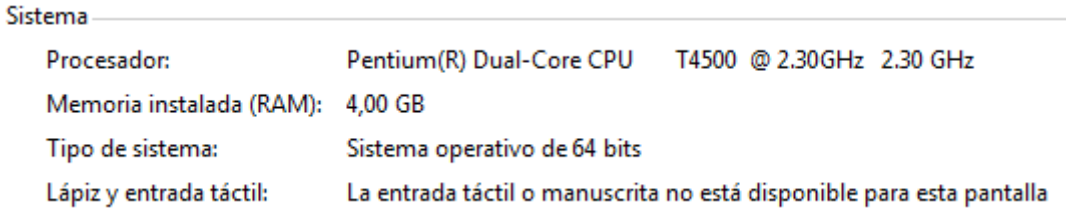


Figura 22. Características de la máquina.

Se han realizado tres tipos de pruebas/experimentos:

- Generación de una nueva solución completa a partir de una inicial informando una serie de incidencias.
- Comparación de tiempos de proceso con diferentes escenarios de incidencias sobre una misma solución inicial.
- Comparación de tiempos de proceso para la generación de soluciones.

Generación de una nueva solución completa a partir de una inicial informando una serie de incidencias

Para esta prueba hemos cargado la solución inicial de ejemplo del día 01/09/2012 seleccionando los ficheros **config01__09__2012.txt** y **01092012__salida.txt**:

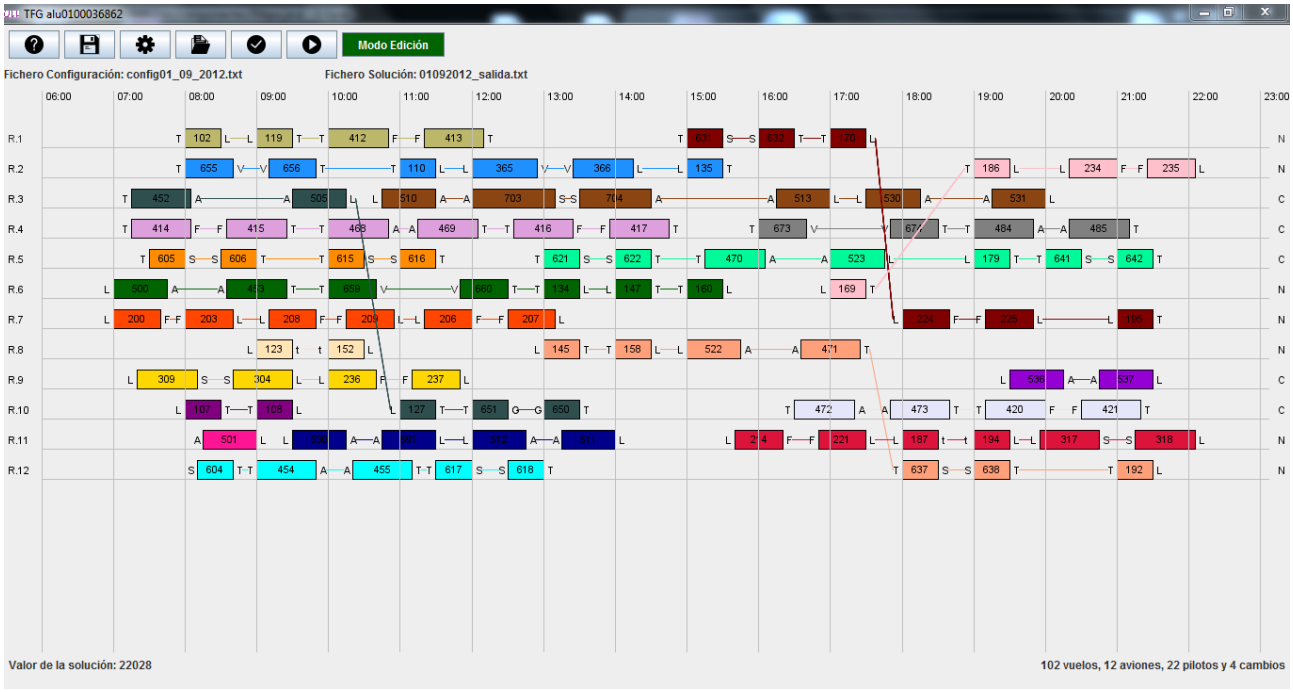


Figura 23. Solución inicial.

Hemos informado las siguientes incidencias en la solución inicial:

ACCIÓN	NÚM. VUELO	TIEMPO
ADELANTO	102	30 min.
ADELANTO	145	15 min.
RETRASO	413	30 min.
CANCELACIÓN	590	-
CANCELACIÓN	187	-
NUEVO VUELO	700	-
NUEVO VUELO	705	-

Tabla 3. Incidencias informadas

Este es el formulario en modo edición con las incidencias informadas:

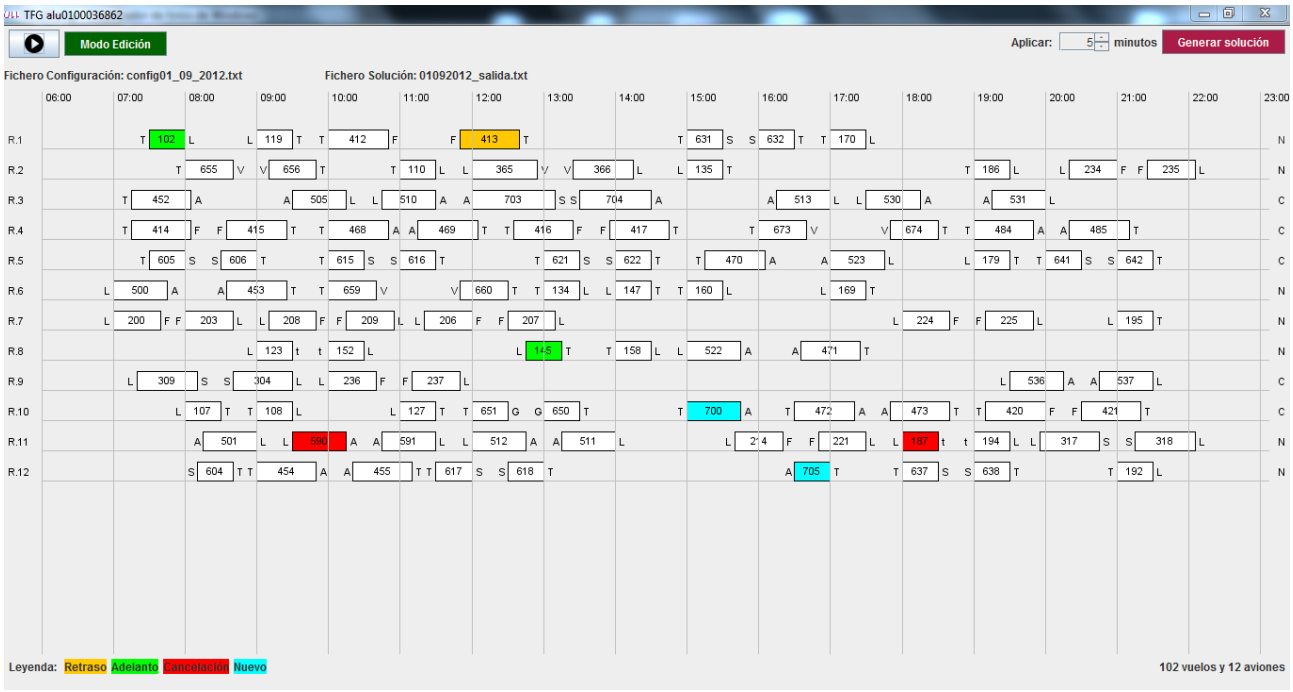


Figura 24. Edición de la solución inicial.

Una vez introducidas las incidencias, se ha pulsado el botón de **"Generar solución"** de la parte superior derecha, el cual hace la llamada al optimizador, y después de 4'25" se ha creado y cargado la nueva solución con las incidencias aplicadas:

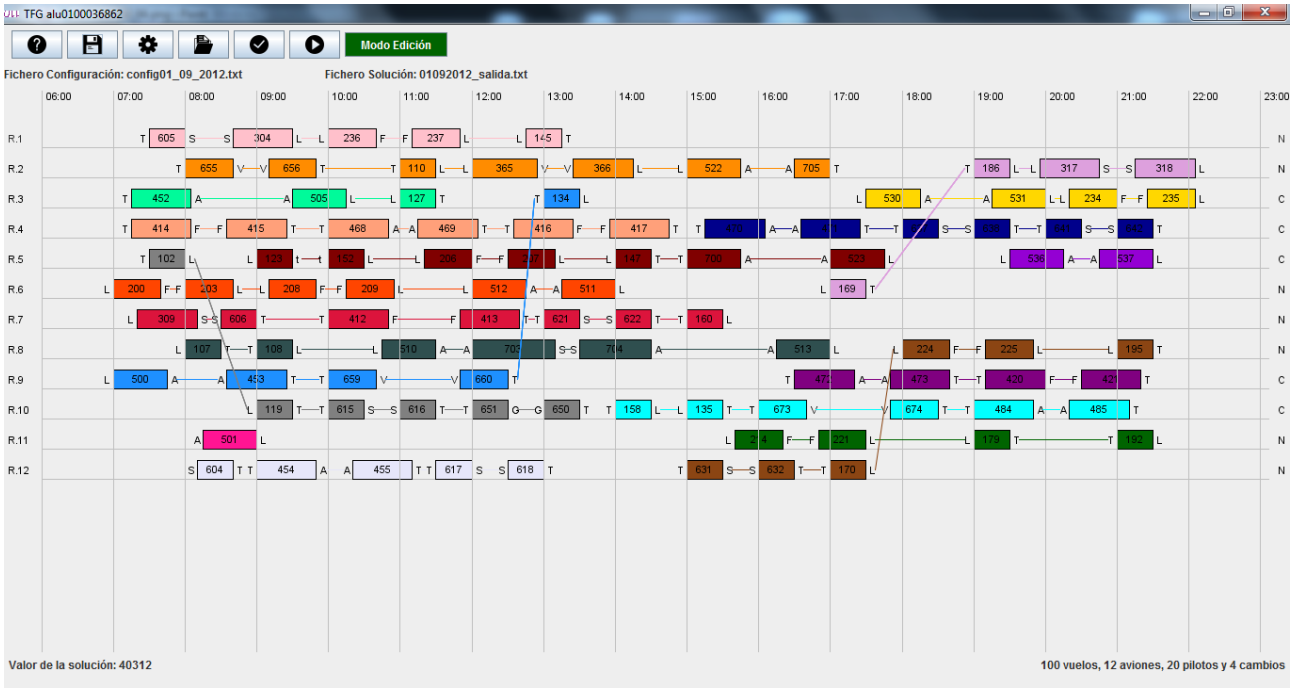


Figura 25. Nueva solución.

La siguiente imagen muestra una comparativa entre la solución inicial y la obtenida después de informar las incidencias y obtener la nueva solución con la planificación:

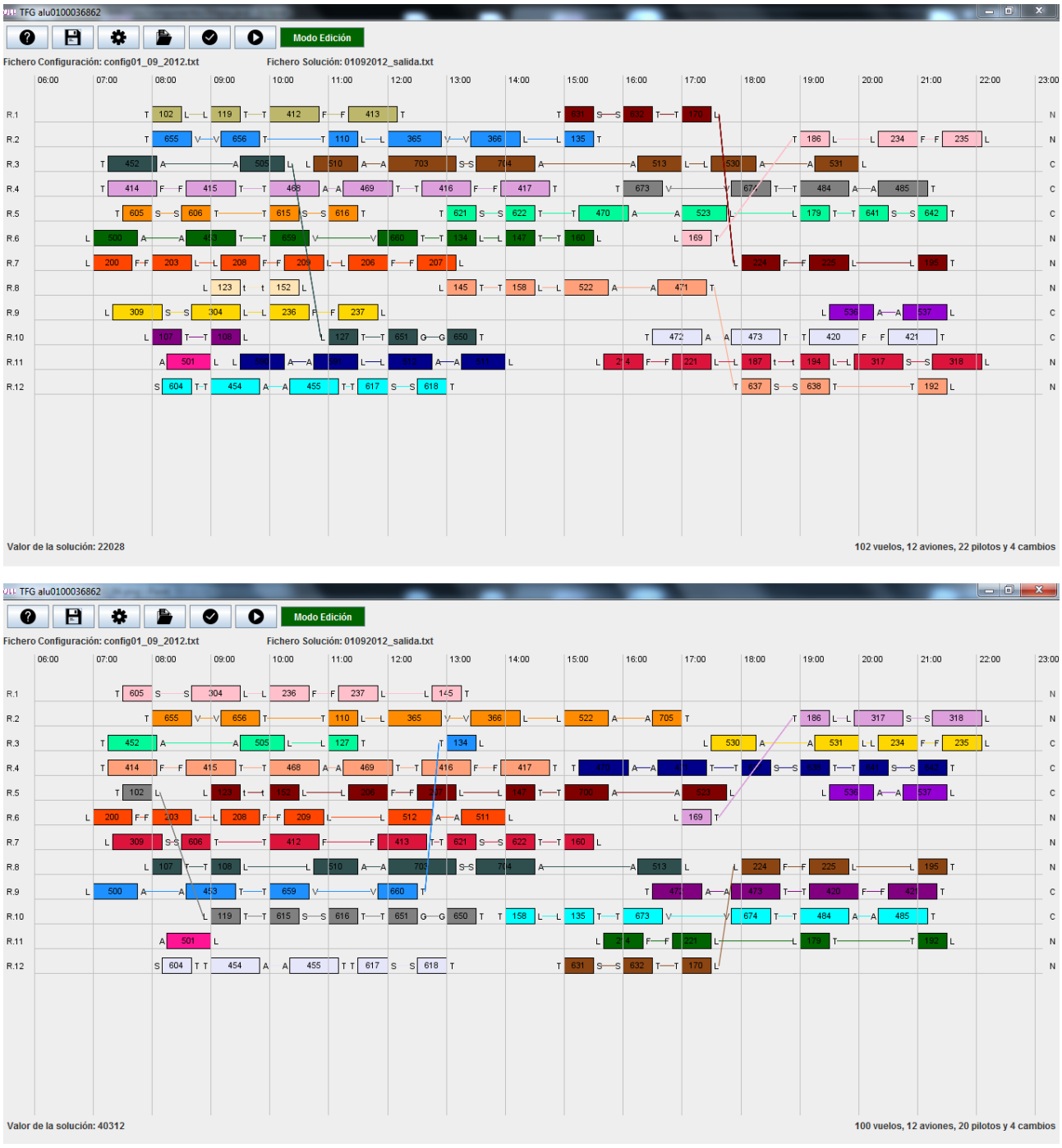


Figura 26. Comparativa de soluciones.

Comparación de tiempos de proceso con diferentes escenarios de incidencias sobre una misma solución inicial

Para esta prueba hemos cargado la solución inicial de ejemplo del día 01/09/2012 seleccionando los ficheros **config01__09__2012.txt** y **01092012__salida.txt**. Esta solución es la de partida en todos los procesos y vemos los tiempos de ejecución obtenidos:

NÚM (Id)	NÚM. INCIDENCIAS INFORMADAS	TIEMPO (minutos)
1	3 (2 A y1 R)	2'10"
2	3 (3 N)	12'02"
3	5 (1 A, 1 R, 1 C y 2 N)	4'25"
4	5 (2 A y 3 C)	6'33"
5	7 (2 A, 1 R, 2 C y 2 N)	4'25"
6	7 (3 A, 1 R, 2 C y 1 N)	22'44"
7	7 (2 A, 4 R y1 C)	17'17"
8	7 (1 A y 6 N)	Sin solución
9	9 (3 A, 3 R y 3 C)	24'56"
10	9 (2 A, 3 R, 2 C y 2 N)	30' (*)
11	9 (2 A, 2 R, 1 C y 4 N)	Sin solución
12	9 (2 A, 1 R, 5 C y 1 N)	13'20"
13	15 (3 A, 2 R, 3 C y 7 N)	30' (*)
14	15 (4 A, 4 R, 6 C y 1 N)	30' (*)

Tabla 4. Comparativa (misma solución inicial)

(A)delanto (R)etraso (C)ancelación (N)uevo vuelo

(*) Sin solución correcta, puesto que omite incidencias.

A continuación, se muestra el detalle de tres experimentos realizados.

Todos los experimentos parten de la misma solución inicial, la mostrada en la **30).**

- **Experimento número 1:**

En este experimento se informaron 3 incidencias (2 Adelantos y 1 retraso) y se procesó con un tiempo de 2 minutos y 10 segundos:

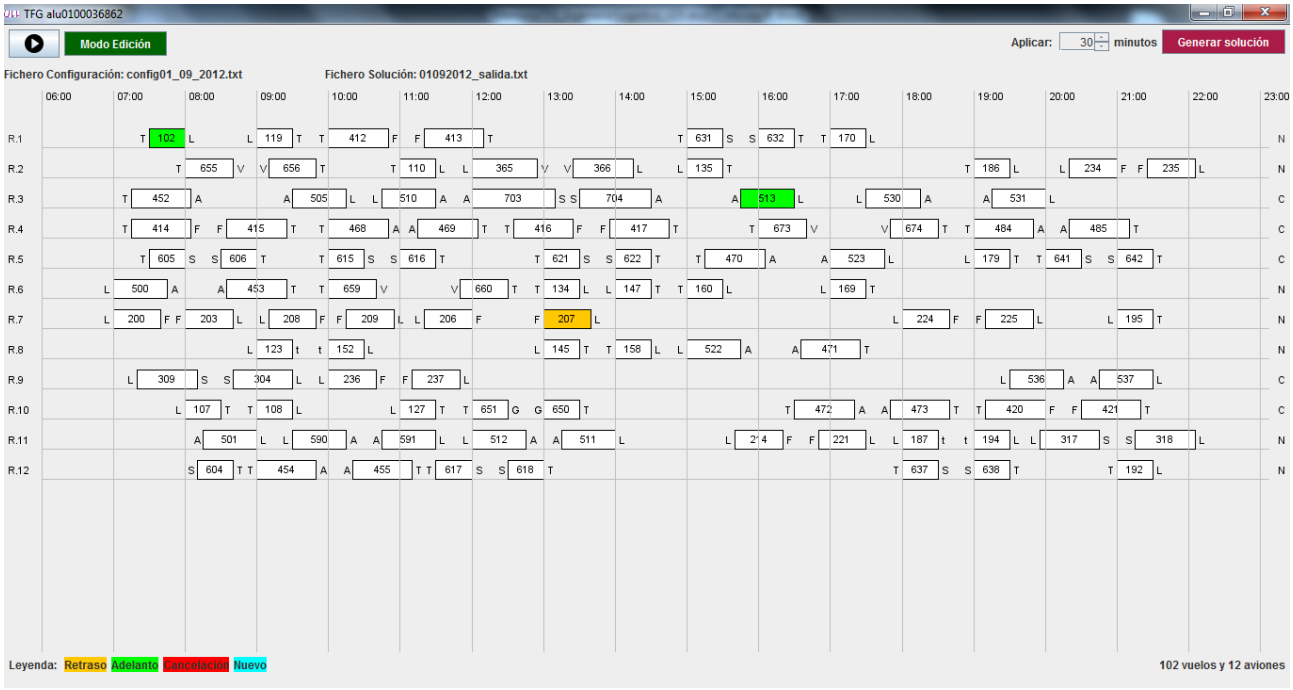


Figura 27. Incidencias Experimento 1.

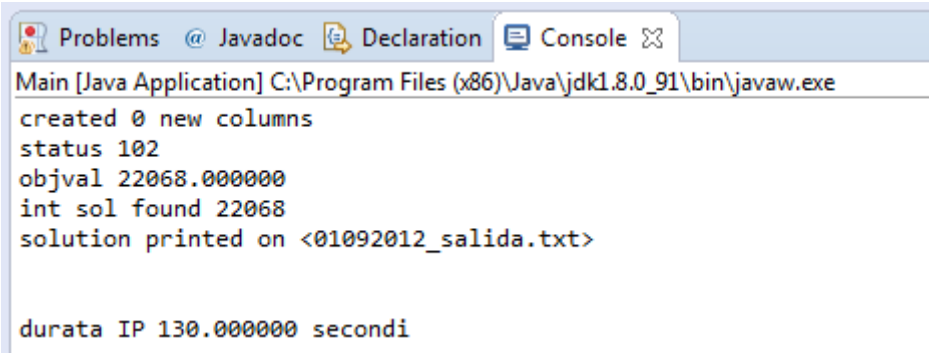


Figura 28. Proceso Experimento 1.

La solución final del experimento es la siguiente:

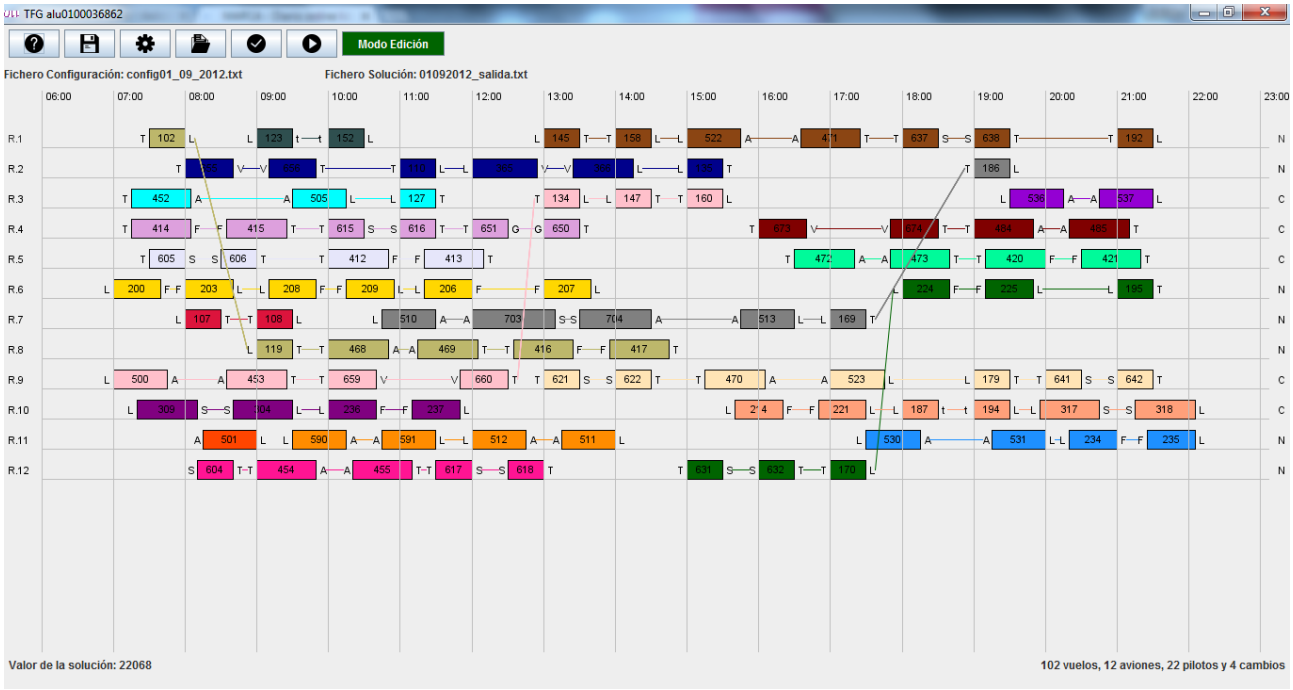


Figura 29. Solución final Experimento 1.

- **Experimento número 6:**

En este experimento se informaron 7 incidencias (3 Adelantos, 1 retraso, 2 cancelaciones y 1 vuelo nuevo) y se procesó con un tiempo de 22 minutos y 44 segundos:

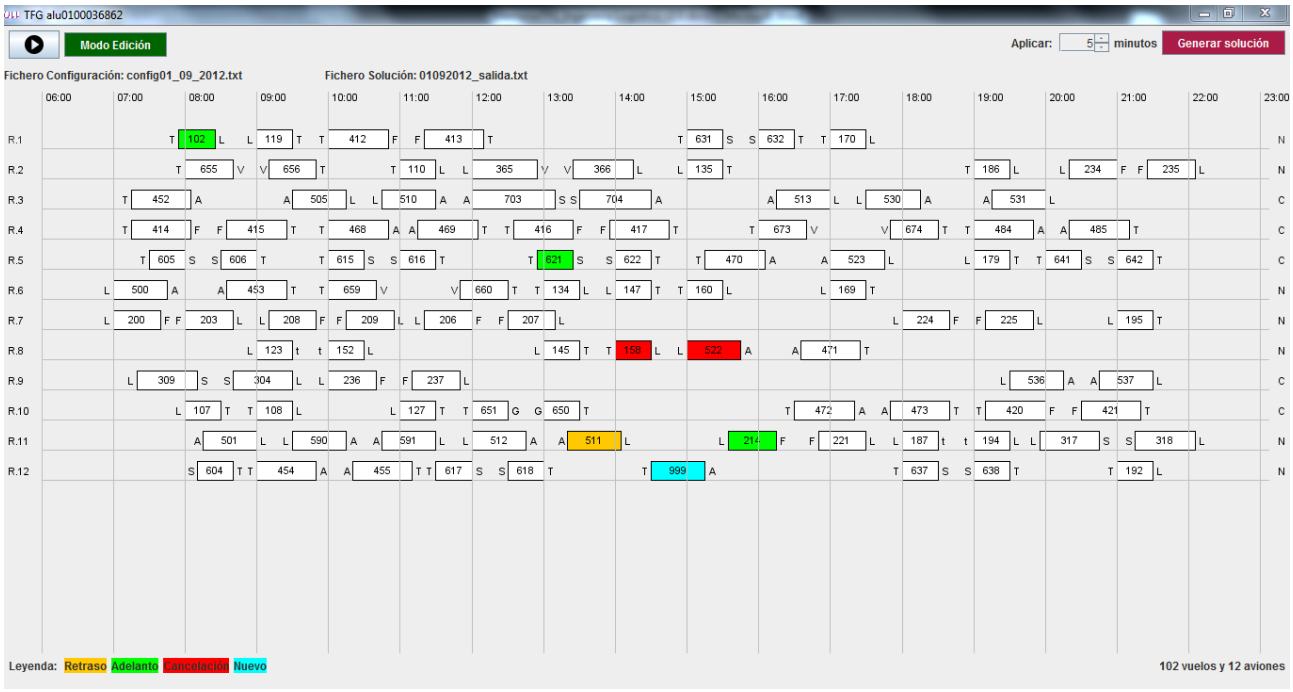


Figura 30. Incidencias Experimento 2.

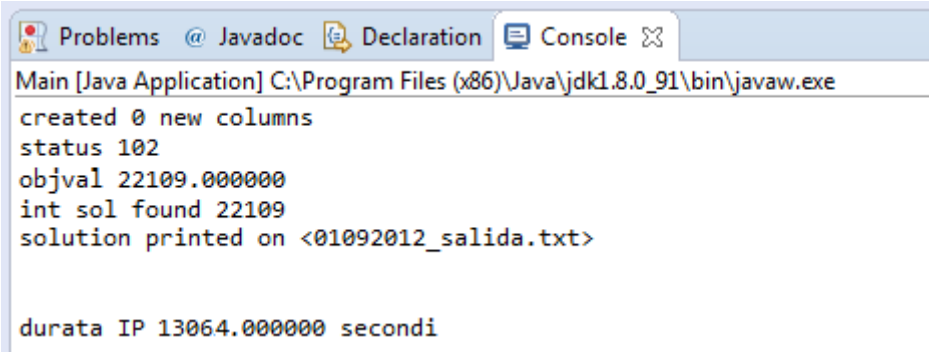


Figura 31. Proceso Experimento 2.

La solución final del experimento es la siguiente:

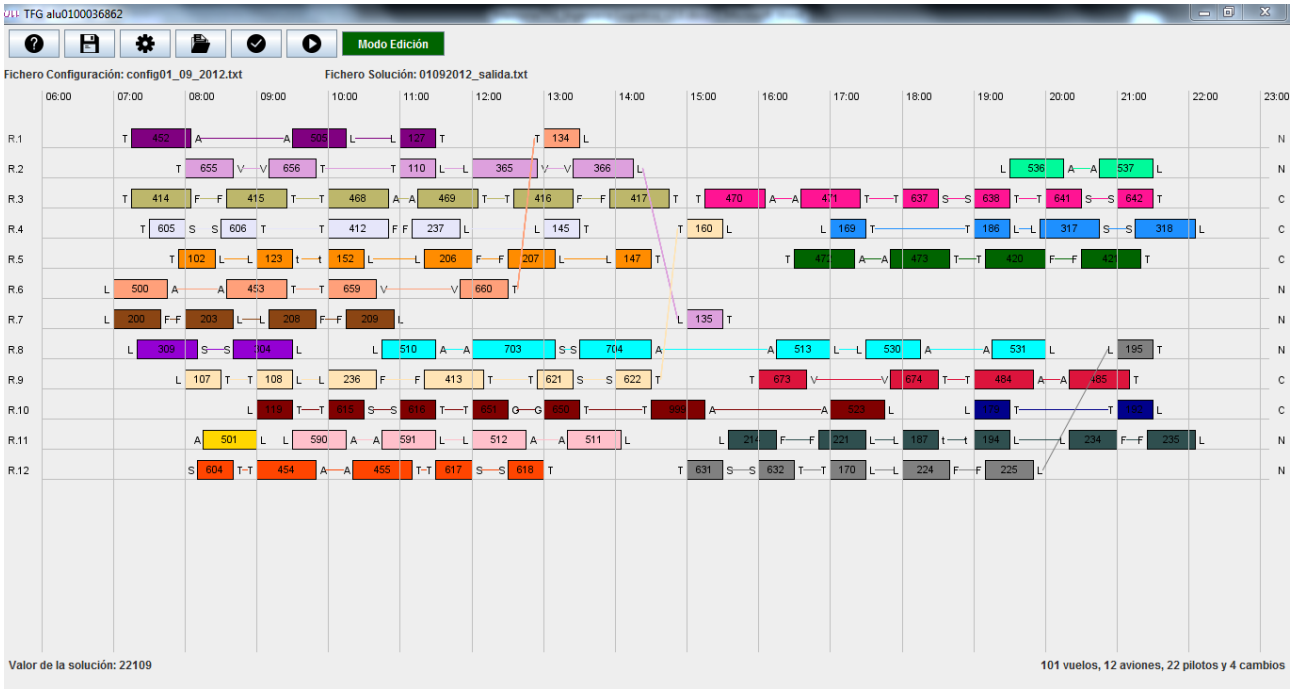


Figura 32. Solución final Experimento 2.

- **Experimento número 13:**

En este experimento se informaron 15 incidencias (3 Adelantos, 2 retrasos, 3 cancelaciones y 7 vuelos nuevos) y el optimizador alcanzó su tope de proceso (30 minutos) devolviendo una solución que podemos comprobar que se omiten muchas de las incidencias, en este caso, los vuelos nuevos, algunas incidencias como el retraso del vuelo 655, por ello muestra un aviso indicando dicha situación:

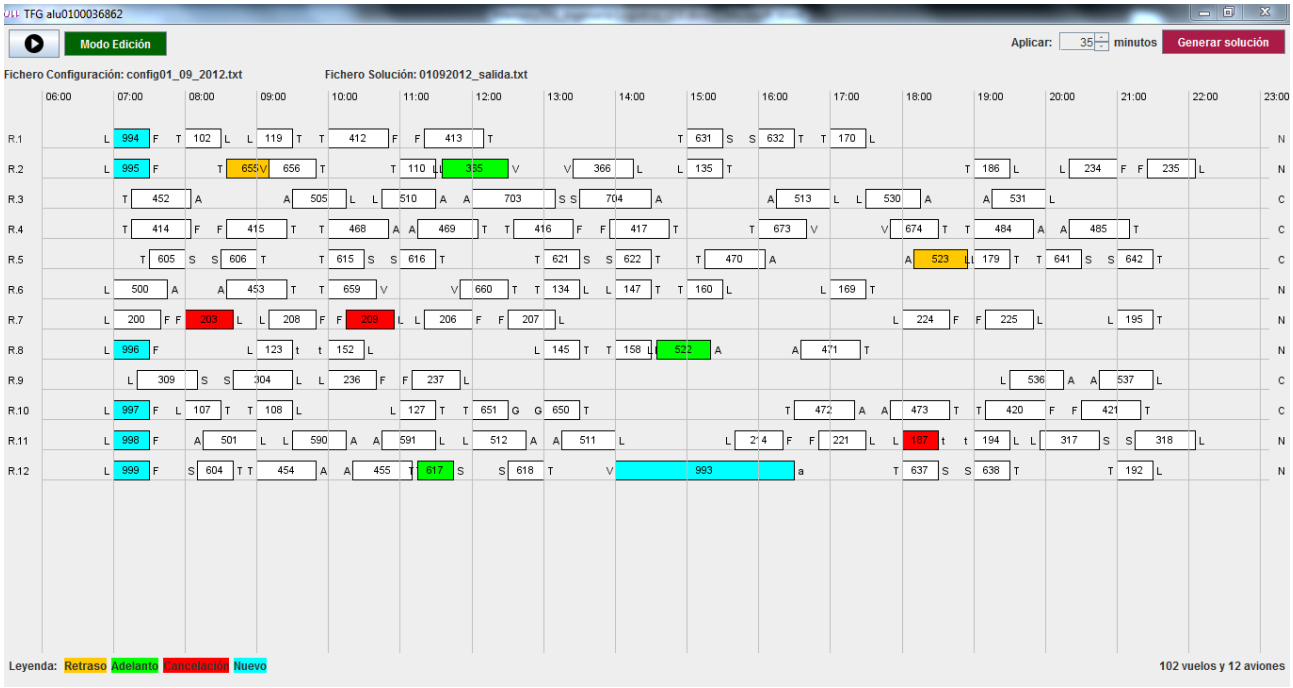


Figura 33. Incidencias Experimento 3.

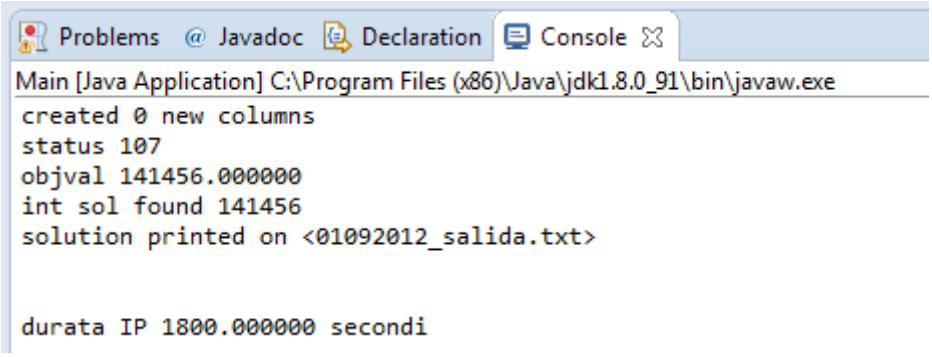


Figura 34. Proceso Experimento 3.

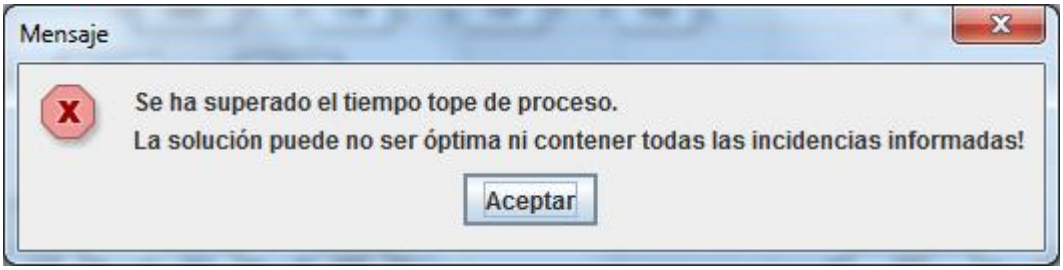


Figura 35. Aviso Experimento 3.

La solución final del experimento es la siguiente:

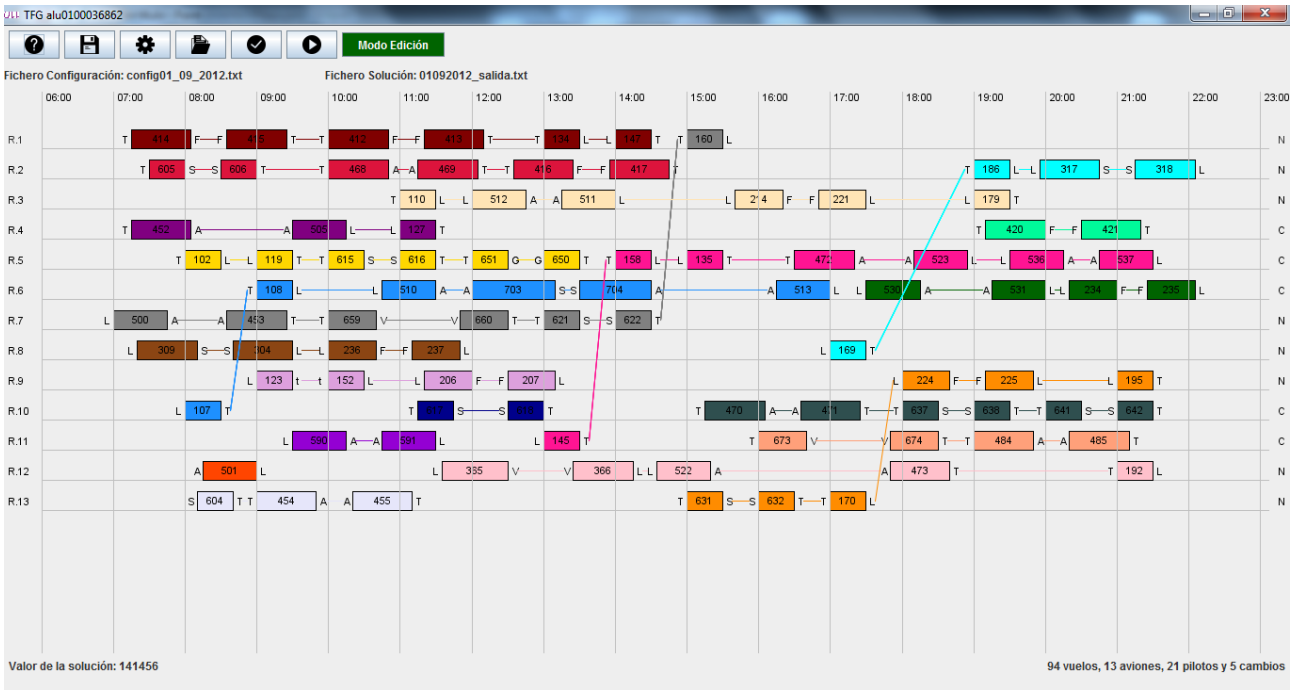


Figura 36. Solución final Experimento 3.

No se puede obtener una conclusión general para estimar el tiempo de proceso de la herramienta para obtener la solución óptima porque de los resultados obtenidos, no se desprende que influya el número de incidencias, ya que depende de la solución con la aplicación sobre la que se aplique.

Lo único que se puede obtener, y no como algo que se cumpla siempre, es que cuanto mayor sea el número de incidencias del tipo (N)uevo vuelo, mayor es el porcentaje de que no se encuentre solución.

Comparación de tiempos de proceso para la generación de soluciones

Para esta prueba hemos cargado varias configuraciones y hemos lanzado el proceso de "Generar solución" obteniendo los siguientes resultados:

NÚM (Id)	FICHERO CARGADO	TIEMPO (minutos)
1	config01_09_2012.txt	8'14"
2	config02_09_2012.txt	16'03"
3	config03_09_2012.txt	30'
4	config04_09_2012.txt	26'38"
5	config05_09_2012.txt	19'40"
6	config06_09_2012.txt	19'32"
7	config07_09_2012.txt	22'44"
8	config17_05_2013.txt	30'

Tabla 5. Comparativa (diferentes configuraciones)

De igual manera que las pruebas anteriormente detalladas, no se puede obtener una conclusión general para estimar el tiempo de proceso de la herramienta la obtención de la solución depende de la configuración de vuelos detallada, aunque podríamos determinar que su tiempo de proceso puede ser proporcional al número de vuelos.

Detalle de la comparativa entre las configuraciones que menos y más tiempo de proceso se ha obtenido:

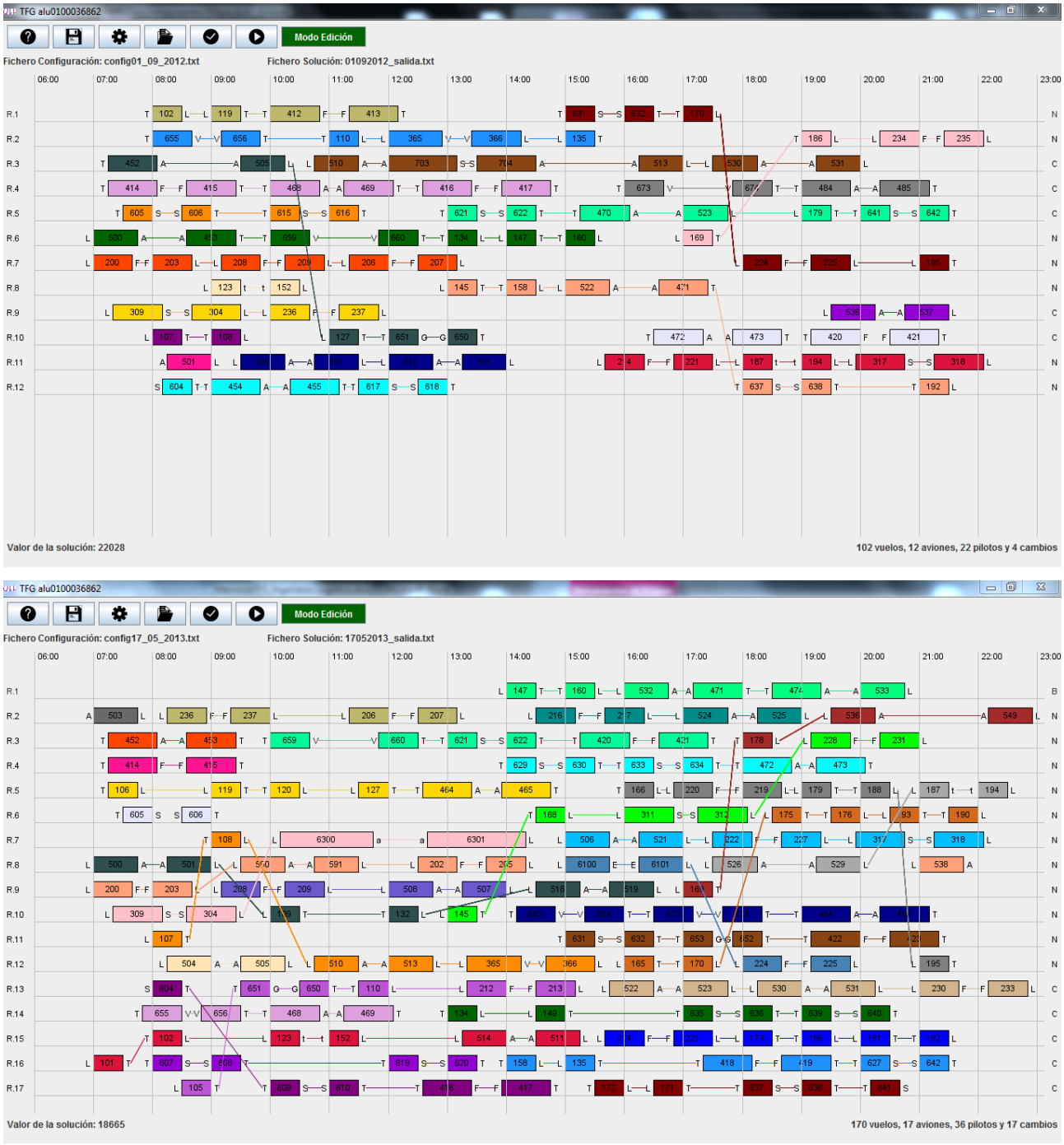


Figura 37. Comparativa de configuraciones (menos y más tiempo).

Capítulo 4.
Conclusiones y líneas futuras

En este capítulo hacemos labores de autocrítica y valoración, además de tener visión para mejorar la herramienta desarrollada.

La meta que se tenía inicialmente ha sido conseguida. Se ha logrado desarrollar una herramienta buena, que ha cumplido con los requisitos establecidos en el comienzo de este camino, camino a veces duro por la falta de conocimiento del negocio, la tecnología aplicada, etc. Se ha culminado con la gran satisfacción de ir logrando cada uno de los hitos para conseguir llevar este Trabajo Fin de Grado a buen puerto.

A nivel personal estoy contento por haber aprendido un nuevo lenguaje y una nueva herramienta de programación que son bastantes comunes para el resto de informáticos pero sobretodo haber aportado mi grano de arena para dar un salto de calidad con el uso de la herramienta en sustitución del uso del programa actual que hace tediosa la tarea del usuario.

Durante el transcurso de este trabajo he comprobado que la Informática no solo es programación, sino es un conglomerado de varias asignaturas que aportan con su contenido los conocimientos necesarios para lograr objetivos como el que se ha buscado en este Trabajo Fin de Grado.

El tiempo y el esfuerzo han valido la pena. Pero aún se puede seguir mejorando el aplicativo con algunas mejoras:

- Añadir mayor especificación de las incidencias que se pueden informar.
- Añadir más información a nivel de pilotos, compañías, aviones, etc.
- Mejorar los tiempos de respuesta del optimizador.
- Incluir un control para fijar una solución hasta una hora determinada y a partir de ésta, generar la solución, es decir, una solución parcial, como si fuera una modificación en directo.

Capítulo 5.
Summary and Conclusions

This chapter touches make self-criticism and assessment work, as well as having vision to improve the tool developed.

The goal than initially has been achieved, it has developed a good tool, it has met the requirements established in the beginning of this road, way sometimes hard for the lack of business knowledge, applied technology, etc. but with the satisfaction of going accomplishing each of the milestones for bringing this Final Project to fruition.

On a personal level I'm happy to have learned a new language and a new programming tool that are quite common for other computer but most have contributed my bit to take a leap of quality with the use of the tool in place of use the current program that makes the user's task tedious.

During the course of this work I have checked that the computer is not just programming, it is a whole of several subjects that provide their content knowledge needed to achieve the goal.

The time and effort have been worth it but you can still continue to improve the application with some improvements:

- Add further specification of the issues that can be reported.
- Add more level drivers, companies, aircraft, etc.
- Improve response times optimizer.
- Include a check to secure a solution to a specific time and from this, build the solution, that is to say, a partial solution, like a modified live.

Capítulo 6. Presupuesto

En este último apartado se detallará el presupuesto correspondiente para llevar a cabo la obtención de la herramienta **"Editor de Turnos Automático para Pilotos y Aviones (ETAPA)"**.

La duración en tiempo será de tres meses aproximadamente para su consecución.

Concepto	**Cantidad**	**Precio (€)**	**Total (€)**
Gastos estructurales (luz, mobiliario, alquiler, etc.)	3 (meses)	200€	600€
Desarrollo de la Herramienta	1	1800€	1800€
Coste programador (salario + obligaciones empresa)	3 (meses)	200€	6000€
TOTAL			8400€

Tabla 6. Presupuesto.

Bibliografía

[1] http://etd.dtu.dk/thesis/57968.

[2] http://link.springer.com/chapter/10.1007/3-540-46004-7_31.

[3] http://www.tibco.com/assets/blt4beb92d84056e223/wp-airline-disruption-management.pdf.

[4] https://integrantes1.wikispaces.com/CUADRO+COMPARATIVO.

[5] http://www4.tecnun.es/asignaturas/Informat1/AyudaInf/aprendainf/Java/Java2.pdf.

[6] http://recursosformacion.com/wordpress/2013/05/java-para-programadores-5-3los-metodos-graphics-y-paint/?doing_wp_cron=1472043612.7541470527648925781250.

[7] http://personales.unican.es/corcuerp/java/Labs/LAB_5a.htm.

[8] http://dalila.sip.ucm.es/~manuel/JSW1/Slides/Swing.pdf.

[9] http://es.slideshare.net/njca01/layouts-java.

[10] http://www.java2s.com/Code/Java/Event/Moveandscalegraphicalobjectswithamouseonthepanel.htm.

[11] http://jsfiddle.net/spedwards/c2Caj/.

[12] http://www.java2s.com/Tutorial/Java/0240___Swing/CreateaMessageDialogBox.htm

[13] http://www.igt.in/blog/airline-disruption-management/.

[14] https://sigarra.up.pt/feup/pt/pub_geral.show_file?pi_gdoc_id=391782

[15] http://www.tibco.com/assets/blt4beb92d84056e223/wp-airline-disruption-management.pdf

[16] https://brage.bibsys.no/xmlui/bitstream/handle/11250/2351128/13986_FULLTEXT.pdf?sequence=1&isAllowed=y