Universitat Autònoma de Barcelona

escola d'enginyeria

Programación de rutas y asignación de aeronaves

Trabajo de fin de grado

Grado en Gestión Aeronáutica

Curso 2014-2015

Miquel Rius Carmona

Dirigido por: Dr. Ángel A. Juan

29 de Enero de 2015

El **Dr. Angel Alejandro Juan**, profesor de la escuela de ingeniería de la UAB,

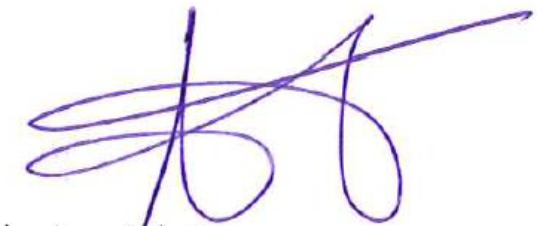
Certifica:

Que el trabajo al que corresponde la siguiente memoria ha estado realizado por:

Miquel Rius Carmona

Y para que conste firma la presente.

Sabadell, **julio de 2015**



Firmado: Dr. Angel Alejandro Juan

HOJA DE RESUMEN - TRABAJO FIN DE GRADO DE LA ESCUELA DE INGENIERÍA

Titulo del trabajo: Planificación de rutas y asignación de aeronaves
Autor: Miquel Rius Carmona
Tutor: Dr. Angel Alejandro Juan **Fecha:** Julio 2015
Titulación: Grado en gestión aeronáutica
Palabras claves: **Catalán:** Planificació, rutas, aerolinies, logística, flota, aeronaus. **Castellano:** Planificación, rutas, aerolíneas, logística, flota, aeronaves. **Ingles:** Planification, routes, airlines, logistics, fleet, aircrafts.
Resumen del trabajo de fin de grado: **Catalán:** En aquest projecte s'ha dut a terme la realització d'un algoritme per tal de solventar la problematica que se'ls hi presenta a les aerolínies comercials i a les de logística en el moment de fer la planificació de les rutes a cobrir, i el seu horari. Aquesta aplicació al mateix temps ajuda a resoldre en una primera fase el problema de la assignació d'aeronaus a aquestes rutes. **Castellano:** En este proyecto se ha llevado a cabo la realización de un algoritmo para solucionar la problemática que se les presenta a las aerolíneas comerciales y a las de logística en el momento de hacer la planificación de las rutas a cubrir, y su horario . Esta aplicación al mismo tiempo ayuda a resolver en una primera fase el problema de la asignación de aeronaves a estas rutas. **Ingles:** This project has been carrying out an algorithm to solve the problems that are presented to commercial airlines and logistics when it comes to planning the routes to cover, and it's schedule. This application while helping to solve in a first stage the issue of the air fleet assignment to the same routes.

Agradecimientos

Este proyecto no hubiese sido posible sin el soporte que he tenido por un conjunto de personas.
En especial, y primero de todo al Dr. Angel Alejandro Juan, el cual ha actuado como tutor del proyecto y ha ayudado en todas las fases del mismo dando soporte, opiniones, ideas entre otros.

Al mismo tiempo y no menos importante a Álex Moreno, compañero de carrera que ha ayudado en la evaluación de los resultados y en la aplicación en el campo real de los mismos a través de su experiencia en la planificación de rutas gracias a su trabajo.

Finalmente a Foix Coll por su dedicación en la revisión del texto y del apartado estético para garantizar un buen resultado y una buena presentación final.

0. Resumen

La asignación de flotas a los itinerarios es el segundo paso que realizan las aerolíneas en el proceso de planificación de las operaciones futuras. Como tal, la correcta definición de qué aeronave cubrirá cada vuelo llevará a una correcta planificación operacional en los otros ámbitos que se ven afectados por este, como si de una cascada se tratase. Además en esta fase, se gestionan los activos más valiosos de una aerolínea, sus aviones, y los que a la vez, por reglamentaciones internacionales, sufren grandes restricciones y no pueden ser asignados deliberadamente.

Este proceso se lleva a cabo a largo plazo, por lo que la desviación que puede sufrir esta planificación hasta el día de operaciones es más que visible. Esta desviación puede afectar la eficiencia de las operaciones, tanto en términos operacionales cómo económicos. El coste de asignar un avión con poca capacidad a una ruta con mucha demanda puede destruir la cuota de mercado de la aerolínea. Además, es importante entender que la asignación de flotas depende, a su misma vez, de las rutas que tenga que volar cada aeronave.

Así pues, en este proyecto se presenta un modelo de resolución que tratará como un único problema la planificación de rutas y la asignación de flotas. De esta manera se asegurará una asignación eficiente y eficaz de la flota a los saltos de cada vuelo a partir de crear unas rutas que cubran todos los clientes a servir y que minimizan los costes operativas, todo esto cumpliendo con las retracciones técnicas y legales.

SECCIÓN 1: INTRODUCCIÓN

1.1. Introducción

1.1.1. Interés y motivación

El mundo aeronáutico está experimentando últimamente grandes cambios en la gestión, que afectarán a todas las partes interesadas en este sector. Un claro ejemplo que se está desarrollando y que se quiere implementar en el futuro es el SESAR. Promovido por Eurocontrol, el SESAR tiene la intención de conseguir una descongestión del espacio aéreo beneficiando así a los pasajeros, a las empresas y al propio medioambiente.

Uno de los factores clave en este programa serán los itinerarios cubiertos por las aerolíneas. Entendemos cómo itinerario el conjunto de planes de vuelos ordenados por tiempos de despeje programados (Cheng, 2014). Especialmente la correcta gestión de itinerarios implicara una mejora en el desarrollo de cualquier programa que pretenda la sostenibilidad del sector aeronáutico a largo plazo, básicamente porqué cómo determina la lógica, la actividad aeronáutica se basa en los itinerarios que se cubren, por lo que si estos se optimizan, el conjunto global de actividades que dependen de estos serán optimizadas.

Carrier	Depart		Arrive	
Flight #	Time	Airport	Time	Airport
Delta 442	6:20 AM	ATL	7:39 AM	MCO
Delta 171	6:25 AM	ATL	7:46 AM	DFW
Delta 193	8:55 AM	CVG	10:28 AM	ATL
Delta 353	4:35 PM	CVG	6:10 PM	ATL
Delta 267	5:45 AM	DFW	8:52 AM	ATL
Delta 1264	7:45 PM	DFW	10:53 PM	ATL
Delta 1981	3:00 PM	JFK	5:28 PM	ATL
Delta 137	5:30 PM	JFK	8:40 PM	LAX
Delta 292	7:00 AM	LAX	2:20 PM	ATL
Delta 1886	3:15 PM	LAX	10:28 PM	ATL
Delta 929	7:35 AM	MCO	9:13 AM	ATL
Delta 622	10:05 AM	MCO	11:35 AM	ATL
Delta 2246	8:20 AM	MIA	10:13 AM	ATL
Delta 858	5:20 PM	MIA	7:22 PM	ATL

Source: www.delta.com

Figura 1: Tabla de itinerarios de Delta

Una manera de optimizar este conjunto de actividades es a través de una correcta asignación de flotas a estos itinerarios.

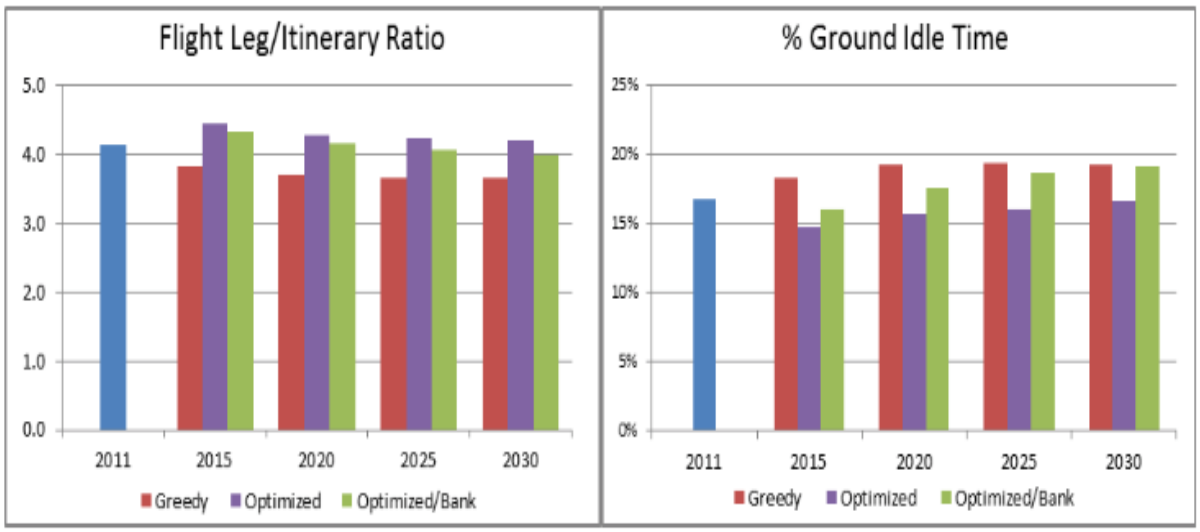


Figura 2: Resultados encontrados por Cheng.F y Gulding. J

Para dar un ejemplo, Cheng. F y Gulding. J (2014) consiguieron una mejora sustancial en el número de saltos por día y en una reducción del *idle time* (Figura 2.), a través de una correcta asignación de flotas respecto a los itinerarios que trabajaban.

Para comprobarlo ejecutaron dos simulaciones distintas, una sin contemplar las restricciones impuestas por las políticas de las aerolíneas y otras contemplándolas. La mejor manera para hacer una correcta asignación de flotas es crear correctamente las rutas que debemos cubrir. Si las rutas son estudiadas minuciosamente, asignar las aeronaves es más fácil, y en este punto se basa este proyecto.

Por ejemplo, no se puede pretender que un *log* sea cubierto por un modelo de aeronave que no sea capaz operacionalmente o legalmente de volarlo.
Así pues es necesario que las rutas estén correctamente definidas, para facilitar la tarea de la asignación de aeronaves. Por lo que es importante trabajar desde un inicio correctamente.

Por este motivo, en el siguiente proyecto se hace un replanteamiento del problema. ¿Podemos mejorar más los resultados en términos de costes y usabilidad de aeronaves si basamos el problema en buscar una solución conjunta a la creación de rutas y a la asignación de flotas?
Así pues, la motivación del proyecto es minimizar las aeronaves asignadas a las rutas a partir de crear el mínimo número de rutas, pero que a su vez minimicen los costes. De esta manera, se conseguirá que la asignación de flotas que depende intrínsecamente de las rutas sea más fácil.

Esto hace operacionalmente más competitiva a la empresa, beneficiando finalmente a las políticas comerciales a través de una reducción de activos (aviones) y ayudando a un descenso de los precios, mejorando así el posicionamiento de la aerolínea en el mercado altamente competitivo en el que viven.

Además, actualmente, las aerolíneas ejecutan la planificación de rutas manualmente. Primero estudian si es factible y rentable ir a un destino y desde qué origen, finalmente “escogen” *slot* y ponen a la venta ese vuelo. Aunque en ningún momento se busca una alternativa de conexión de los distintos vuelos para conseguir servir todos los vuelos con el mínimo número de aviones posibles y minimizando el coste operativo total.

1.1.2. Definición detalla del problema

Llegados a este punto y con la previa introducción al proyecto, se hablará de manera detallada y concreta del problema que se abarca a lo largo de este trabajo. Para ello es importante entender que el proceso de asignación de flota ha sido un tema tratado a lo largo de muchos años, pues optimizar esta parte del proceso de planificación ayuda a ser más competitivos. Y es que el proceso de planificación de una aerolínea no se acaba cuando has decidido dónde y cuando ofrecer los vuelos. Aunque nunca ha sido tratado desde el paso previo a este, es decir, la planificación de rutas.

El proceso de asignación de flotas se puede definir como ejercer una asignación de aeronaves a los saltos de vuelos, contemplando restricciones operativas, técnicas y legales.
Esta fase de asignación de flotas está integrada dentro del desarrollo estratégico, el cual es un proceso a largo plazo y sometido a muchas variables. Tan largo puede llegar a ser el proceso de planificación que, dependiendo de políticas de aerolíneas, se puede estar hablando de planificación a diez años vista.

+ 60 meses	**36-12 meses**	**12-3 meses**	**4-1 meses**
Planificación a largo plazo	Evaluación de mercados	Optimización de la programación	Asuntos de la programación

Figura 3: Cronología en la programación de vuelos.

Como se puede apreciar en la figura 3., la asignación de vuelos se realiza entre 1 y 4 meses antes del día de operaciones.

Esta asignación de flotas se trabaja en dos niveles:
- Nivel general: Se trabaja con qué tipo de aeronave deberá cubrir cada ruta. Por ejemplo, la ruta X debe ser cubierta por un A320.
- Nivel específico: Se trabaja a nivel más detallado. La ruta X la debe cubrir el A320 con matricula XYJ230, por ejemplo.

Actualmente, las aerolíneas trabajan en un ambiente determinista que afecta a una serie de inputs como son:

- Demanda
- Capacidad
- Tipo de flota
- Frecuencias de vuelo
- Aeropuertos
- Mercados que explotar
- Datos de mantenimiento
- Tripulaciones

Con estos datos construyen la planificación de vuelos, que no es nada más que determinar las horas de salida y de llegada de los distintos vuelos, además de qué flota deberá volar cada uno de los saltos. Claro estaá que rara vez ocurre que se cumplan estos tiempos de manera exacta, pues el sistema real es estocástico y no determinista, es decir se ve influenciado por otras variables aleatorias. Además, ciertos inputs como la demanda, varían en función de lo cerca que esté el día del vuelo.

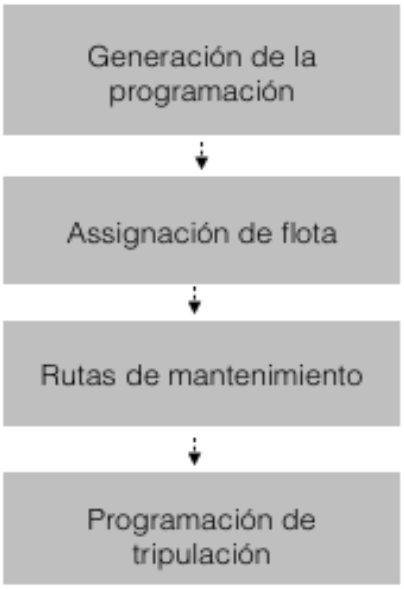


Figura 4: Fases del proceso de planificación

El proceso completo de planificación se divididae en 4 fases:

Cómo se puede ver en la figura 4. el primer paso es el conocido como *schedule generation*, es decir, la generación de la programación de los vuelos o itinerarios.

Esta fase empieza con un desarrollo de las rutas, las cuales son decididas por la aerolínea que opta por aquellos mercados que quiere explotar, basándose en la demanda pronosticada. Una vez saben el mercado, deben escoger la apropiada frecuencia de vuelo a esos mercados, el objetivo pues es satisfacer la demanda prevista.

Finalmente, se debe evaluar a qué tiempo, es decir horas, se ofrecerán estos vuelos. Cómo se ha dicho anteriormente no se comprueba que pueda haber una combinación de legs mejor de la actual con ningún algoritmo, por lo que existe un posible margen de mejora.

La segunda fase es cuando se determina esos vuelos qué flota los realizará. Posteriormente, se especifica qué aeronave cubrirá dicho salto, contemplando restricciones de rutas de mantenimiento (tercer paso).

Por último, se asigna a cada salto y aeronave una tripulación, acorde a las características de ambos.

La asignación de flota tiene un gran impacto en los ingresos y costes de la aerolínea. Un ejemplo es que si tenemos un avión A con capacidad de 100 pasajeros y asignamos dicho avión a una ruta de 50 pasajeros, estaremos desaprovechando capacidad, por lo que se generan demasiados costes fijos que no cubriremos con ese *load factor.* Si por el contrario cubriésemos una ruta de 150 pasajeros, estaríamos generando un alto *sil cost*, que se define como aquel coste que se produce al perder ingresos de pasajeros por el hecho de asignar un avión con menor capacidad que demanda.

Así pues, en este proyecto se presenta una variación del actual proceso de programación. En este nuevo modelo se pretende combinar el paso de programación de vuelos con la asignación de flotas, o mejor dicho se pretende crear un algoritmo que de como *outputs* unas rutas que minimicen costes y aeronaves.

Para ello se le dirá al algoritmo los *hubo* de la aerolínea y los clientes a cubrir por cada hub. Con esto, se calculará la manera más óptima para combinar los legs entre aeropuertos.
Se considera óptima la solución que minimiza el número de aviones usados para cubrir todas las rutas de la solución y, a la vez, minimiza el coste de dicha solución.

Se ve claramente que el algoritmo y modelo desarrollado será usado como planificador de rutas a cubrir y, consecuentemente, del orden que deben seguir los vuelos. A partir de este orden, el algoritmo (en versiones futuras) podría mejorarse para que te sugiriese un horario que asegure el cumplimiento de la ruta a partir de otro input, como podrían ser los slots.

A su misma vez, y como crea rutas el algoritmo minimiza el número de aeronaves por lo que hace una primera aproximación al proceso de asignación de aeronaves y facilita este proceso, por esto se habla de un problema combinado.

Aunque en versiones futuras deberán añadirse todas las fases de este proceso, es decir asignación general y específica, ya que por el momento el único trabajo que hace de asignación de flotas es asegurarse que se asignarán el mínimo número posible de aeronaves para cubrir todos los clientes.

Dadas las condiciones y plazo de tiempo del proyecto, es importante entender que se dimensionará el proyecto y el algoritmo para poder entregar una primera fase de este modelo y planteamiento. Algunas cosas importantes que se deben considerar son:

- No se consideran condiciones de mantenimiento, ya que antes debería probarse este algoritmo en un entorno real y verificar que resulta de interés para las aerolíneas comerciales y de cargo.

- La asignación de aeronaves se hace a nivel abstracto, es decir, en ningún momento se dice qué modelo debe cubrir una ruta y, mucho menos, qué aeronave en particular. Lo que se trabaja es la minimización de aeronaves usadas para garantizar que la asignación siempre será eficiente (y más fácil) cuando en versiones futuras se mejore el algoritmo.

Por lo tanto, el problema abarca el factor de generación de itinerarios y el de *flete assignment.* El objetivo de este proceso es asignar a tantos *legs* cómo sea posible, una aeronave, mientras se optimizan ciertos objetivos y se cumplen algunas restricciones.

Gracias a este modelo se consigue plantear una nueva aproximación al problema, ya que la mayoría de modelos se basan en optimizar el proceso objetivo en cuestión y no los pasos previos a esos. Esto quiere decir que muchos modelos que quieren mejorar la asignación de flotas solo trabajan algoritmos para optimizar esa asignación, sin preocuparse si la planificación de rutas es correcta y está optimizada.

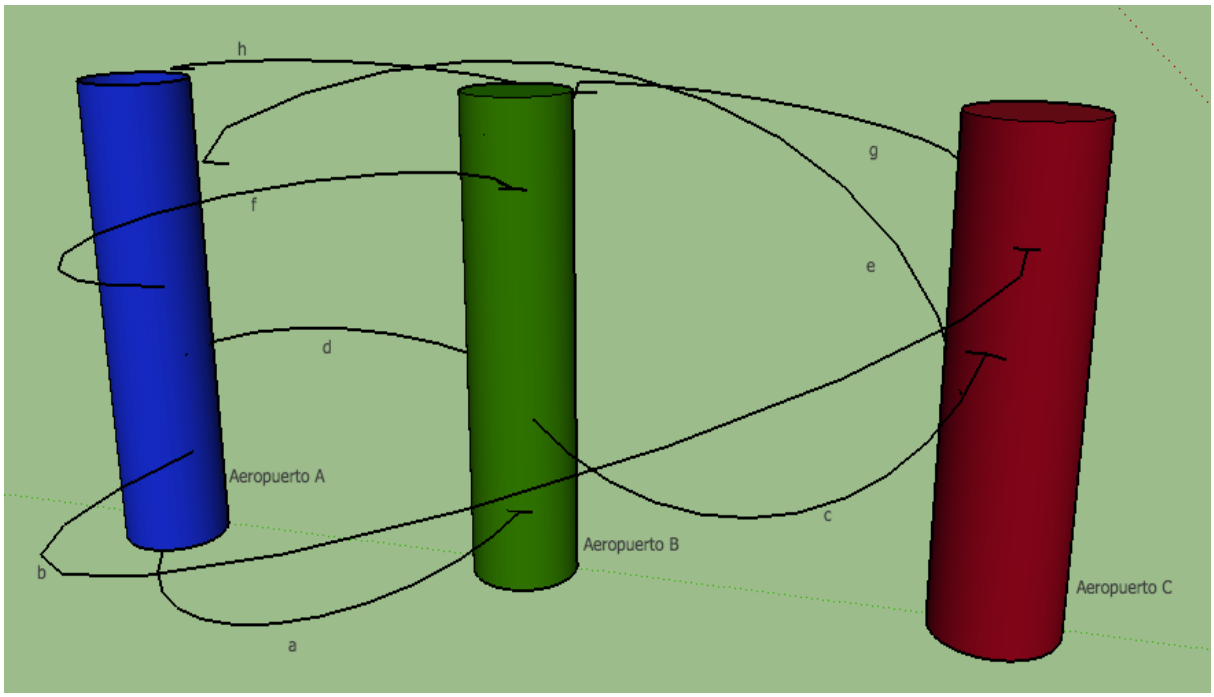


Figura 5: Esquema conceptual del problema

En la figura 5 se puede apreciar una representación conceptual del problema, donde cada barra supone un aeropuerto. Estos están unidos por los diferentes *light legs.* El objetivo es crear una secuencia de legs (líneas negras) que minimice el número de aeronaves usadas y el coste total de la solución.

Debido a que ayudará a la reducción de costes, este modelo hará que la aerolínea sea más competitiva y, además, con menos aeronaves se podrán realizar los mismos itinerarios.

Finalmente, podemos concluir a cómo este proyecto puede ayudar al sector aeronáutico:

- Ayudaría en la descongestión del espacio aéreo, ayudando así a la seguridad de las operaciones y a la reducción de la contaminación del medioambiente, sin olvidar la simplificación que sufriría la gestión aeronáutica al haber menos aviones que gestionar.

- Ayudaría a la competitividad de las aerolíneas y, por lo tanto, se podrían reducir los precios de los billetes gracias a que se minimizarán costes, lo que implicaría un acercamiento a aquellos pasajeros que son altamente flexibles al precio.

1.1.3. Gestión de la problemática

Con toda esta problemática se presenta una metodología factible. Esta consiste en desarrollar un algoritmo que permita determinar las rutas a realizar en función de unos inputs y de unas condiciones que dependen de una heurística, explicada en los siguientes puntos. Todo esto ayudará a calcular una solución factible y conseguir cumplir con los objetivos del proyecto.

Para solventar la problemática planteada se deben trabajar una serie de *inputs* como son:
- Red de aeropuertos. Aeropuertos servidores y aeropuertos clientes.
- Número de aeropuertos servidores en la red (*hubs)*.
- Valores numéricos que indican como debe trabajar el algoritmo en las ecuaciones que determinan los resultados estocásticos.

1.1.4. Planteamiento del problema

1.1.4.1 Vehicle Routing Problem

Como se verá en el apartado *Literature Review* ha resultado imposible hallar una heurística en la que basarse, así que no se ha tenido una base sobre la que trabajar y mejorar para proporcionar una mejor solución a la actualmente encontrada. Esto en parte, como se ha comentado anteriormente, es porqué nunca antes se ha planteado el problema como una combinación de las dos primeras fases de la planificación que deben realizar las aerolíneas.

Por eso y gracias al tutor del proyecto (Dr. Angel A. Juan), el proyecto se ha planteado como un problema de multidepot VRP.

VRP son las siglas de un conocido problema de logística llamado *Vehicle Routing Problem*, en el cual se intenta hallar la mejor manera en términos de coste para que desde un almacén central (depot) se suministren todos los pedidos a los clientes que se encuentran en posiciones diferentes. Figura 6.

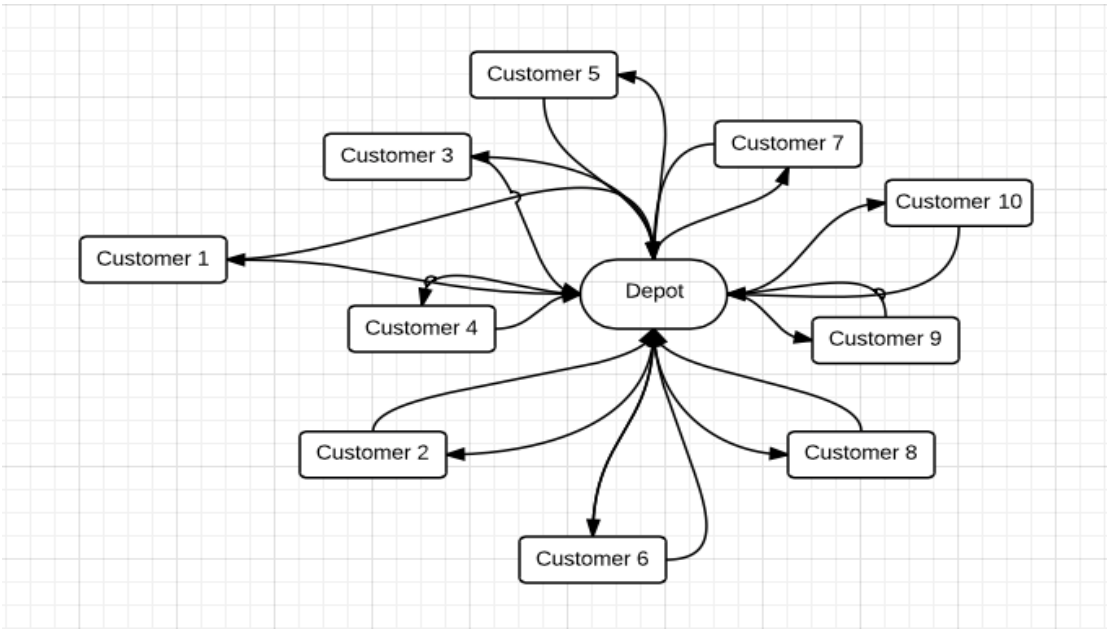


Figura 6: Dummy solution en el problema de VRP

Una solución muy interesante a este problema la presentó Clarke. Esta solución se basaba en una heurística de ahorrar costes en cada trayecto que se conectaba, es decir, inicialmente se daba una *dummy solution* (solución tonta, figura 7) al problema.

Suponiendo que tenemos infinitos camiones en el depot, podemos mandar 1 camión a cada cliente con su pedido y hacerlo volver al depot. Así se garantiza que todos quedan servidos sin problemas en términos de demanda. Esta claro que esta solución no es óptima porque presenta sobre costes, quizás un camión tiene capacidad de sobra como para satisfacer a dos clientes pero no se aprovecha porque sólo se destinará a un cliente.

Como se ve en la figura 7, la *dummy solution* hace más viajes que los necesarios. Un camión que sale del depot va hacia el cliente 1 y vuelve. Otro camión hace lo mismo con el cliente 2, y así sucesivamente.

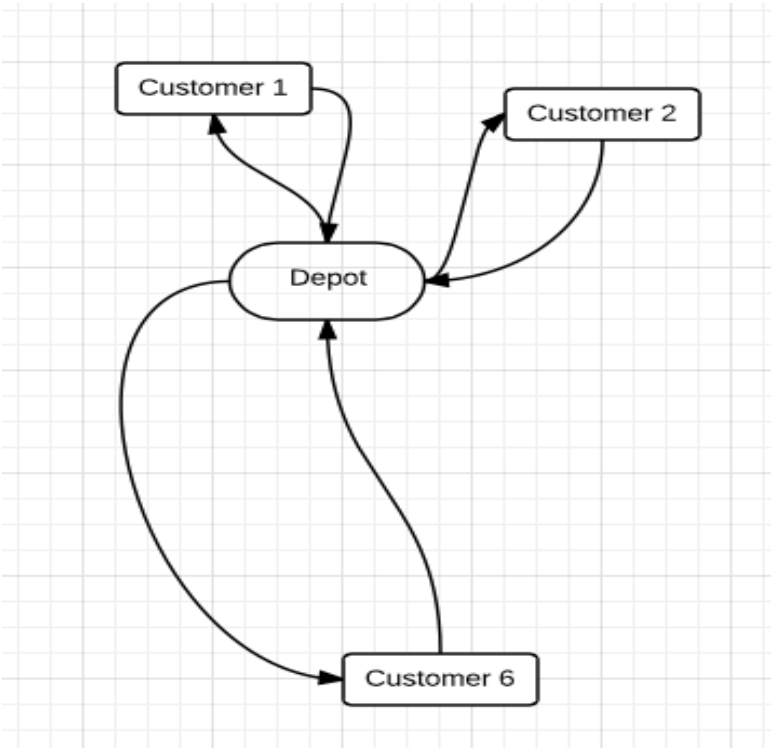


Figura 7: Dummy solution detallada

Esta claro que esta solución podría ser mucho más óptima. Supongamos que cada camión tiene una capacidad de 100, el cliente 1 tiene una demanda de 20, el cliente 2 de 80 y el cliente 3 de 70. Se ve pues que un camión podría satisfacer al cliente 1 y al 2, o al cliente 1 y al 3. Aunque por cuestiones de tiempo, proximidad y otras cuestiones es más rentable la combinación del cliente 1 y 2, aunque la conexión entre 1 y 3 también sea una solución óptima en términos de coste y de oferta/demanda. Si en el algoritmo se cuantifica el tiempo que se tarda, la solución óptima solo será las de conectar el 1 y 2, pero no el 1 y 3. El escenario que quedaría es el de la figura 8.

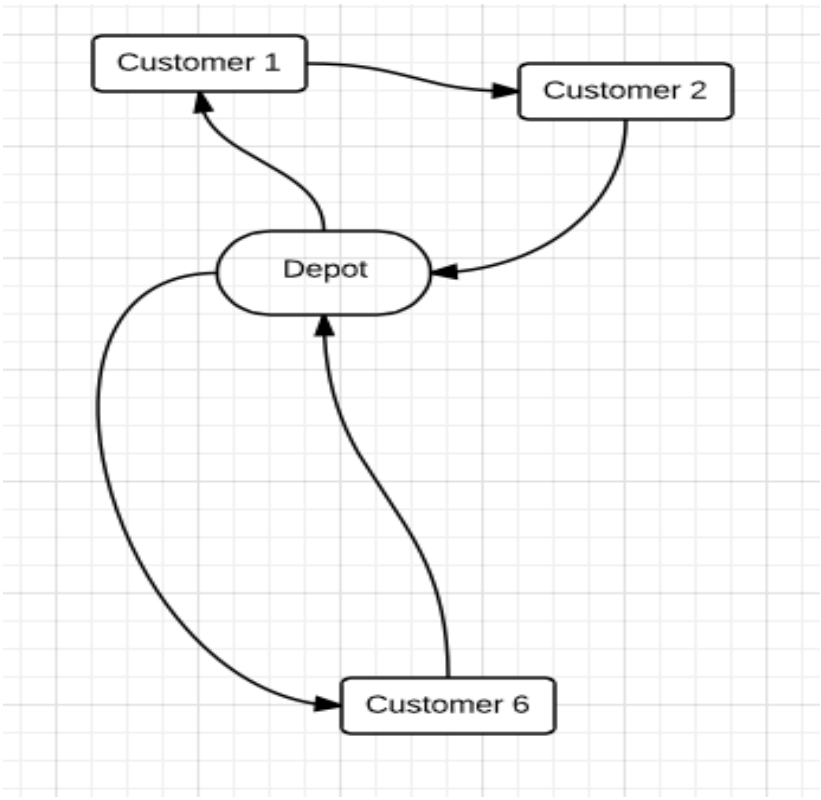


Figura 8: Resultado aplicando CSH (Cost savings heuristic)

En este escenario hay varias ventajas. La primera y más clara de ver es que en vez de 3 camiones se usan 2. Es decir, minimizamos flota. Además, maximizamos la utilización de esta. Antes el camión 1 desaprovechaba un 80% de su capacidad, el camión 2 un 20% y el camión 3 un 30%.

En este nuevo escenario el camión 1 y 2 se fusionan aprovechando en un 100% su capacidad, por lo que es más eficiente. El camión 3 sigue con las mismas estadísticas.

Pero no solo son cuestiones de maximizar el uso y ser más eficientes, sino que también son cuestiones de coste. Al usar un camión menos, nuestra flota puede ser más pequeña, teniendo menos activos y por lo tanto menores costes. Al ser más eficientes conseguimos menores costes operacionales. A su misma vez, la heurística de Clarke determina que si ir de un cliente a otro te cuesta menos que la vuelta y la ida a los diferentes clientes, en realidad estas ahorrando costes y, por lo tanto, podrías satisfacer ambos clientes con un sólo camión. Sin olvidar restricciones como que la demanda de ambos clientes sea más pequeña o igual a la capacidad de tu flota de vehículos.

1.1.4.2 Multidepot Air Fleet Assignment Problem

Extrapolando este planteamiento al problema de la asignación de aeronaves, podemos decir que estamos trabajando un problema de VRP pero con multidepots. Es decir, en vez de que las aeronaves salgan de un solo punto para satisfacer a los clientes, saldrán de varios puntos (aeropuertos hub) para satisfacer a otros aeropuertos (clientes). Se puede apreciar una aproximación simple de este sistema en la figura 9.

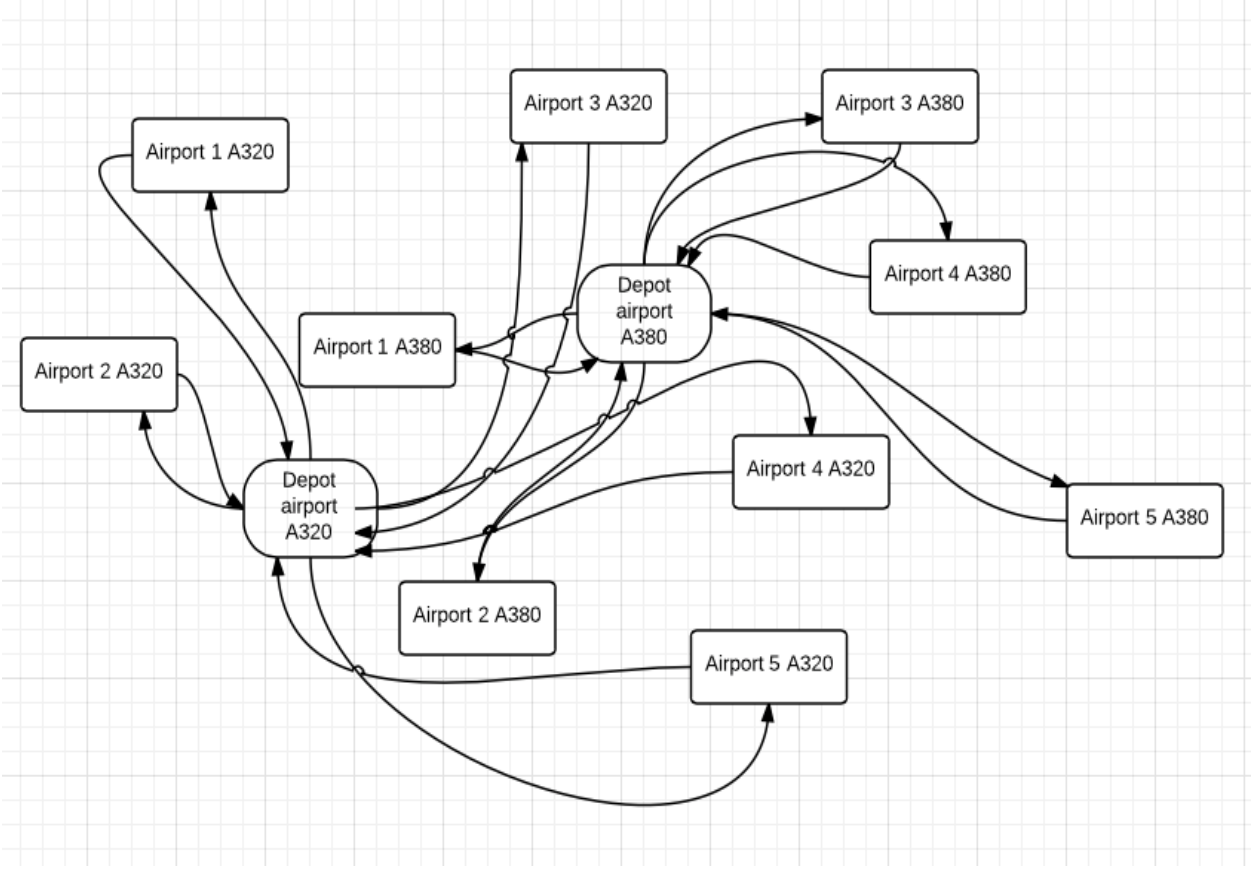


Figura 9: Representación de una red basada en MultiVRP

El problema pues se traduce en que se tendrán varios *depots* que deberán satisfacer a diferentes clientes. Por lo que, igual que en el problema anterior, se generará en primera instancia una solución tonta (*dummy solution*) donde, suponiendo que se tiene una flota infinita, se servirá a cada aeropuerto de destino (cliente) con un avión que irá y volverá del depot al cliente. A partir de aquí y con la heurística de *savings,* explicada en el punto 1.1.4.3, se decidirá en qué momentos sale más rentable ir de un aeropuerto cliente a otro con un solo avión, en vez de ir con dos aviones a cada uno de los clientes.

Por lo que el proceso se basa en una optimización constante de la solución *dummy* a partir de la heurística definida por Clarke. Una solución aproximada de optimizar esta dummy solution se encuentra en la figura 10.

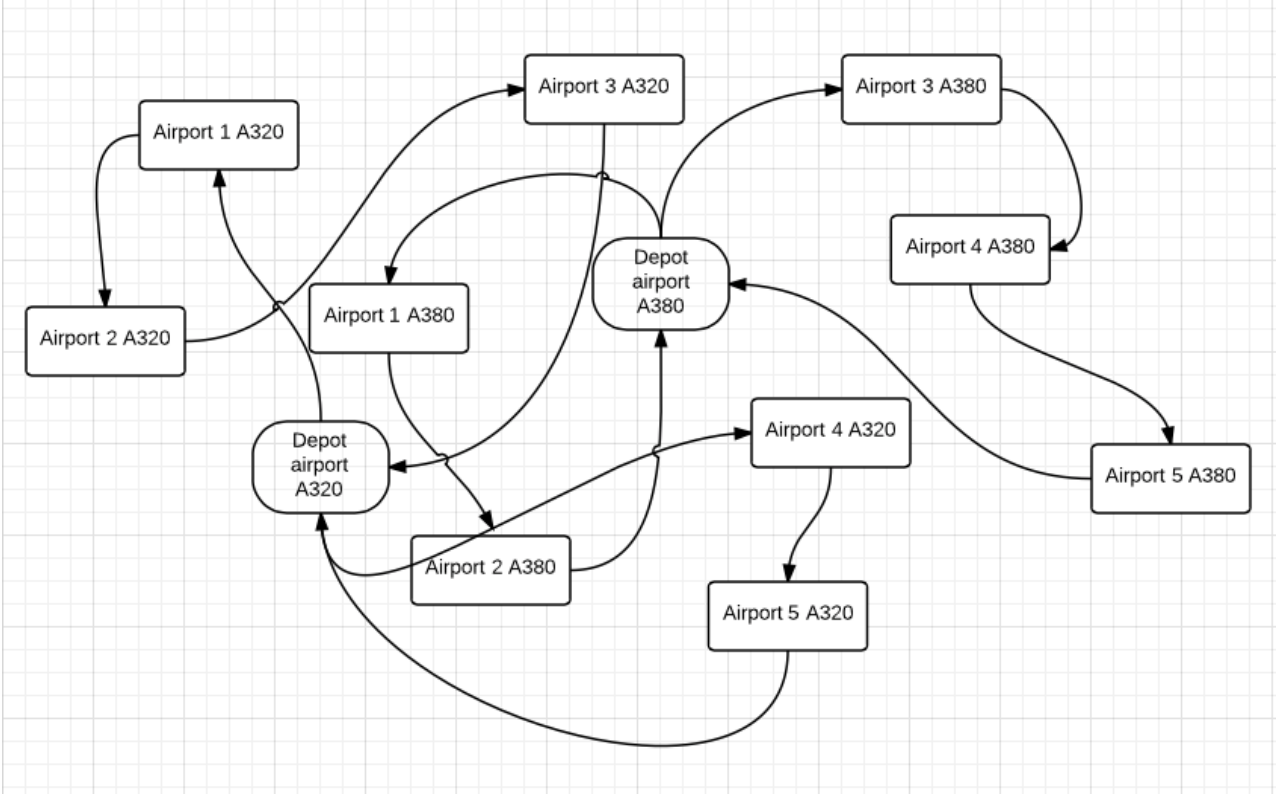


Figura 10: Optimizando el MultiVRP con CSH.

Si se compara la figura 9 con la 10, se ve claramente un menor número de viajes. Con esto se consigue una optimización de flota (en número), una maximización del uso de la misma y se minimizan los costes operacionales. Las ventajas de este método son las mismas que las de aplicar la *Cost Saving Heuristics (CSH)* al problema de VRP.

Así pues, con esta aproximación se presentan dos tipos de aeropuertos:
- Depot: Aeropuertos donde se encuentran las aeronaves al inicio de un ciclo. Cada ciclo empieza y finaliza en el mismo depot.
- Cliente: Aeropuertos intermediarios.

En este planteamiento se deben considerar distintas observaciones relevantes para entender como se pretende llegar hasta una solución optima y la dificultad que presenta la misma:

- Se trabajarán tantos problemas de *Vehicle Routing Problem* como aeropuertos *depot* tenga la red.
Es importante este matiz, pues se trabajará la solución por flota, es decir por capas. No todas las flotas pueden volar a los mismos aeropuertos, por eso se definirán inputs para cada tipo de flota. Esto se traduce en que, a pesar de que se trabaje por ejemplo con 20 aeropuertos, habrá flotas que sólo operen en 5 aeropuertos, y las demás operen en muchos más. Un ejemplo es que un Airbus A380 no puede aterrizar en cualquier aeropuerto, por lo que la red de aeropuertos que puede servir es mucho más pequeña.

Si por ejemplo nuestra aerolínea opera localmente en Inglaterra y en España, pero no hacemos conexiones entre estos paises, un input será el *hub* de Inglaterra con los clientes que este debe cubrir, y otro input será el de España con sus respectivos clientes.

Lo mismo sucedería si desde España se va a todo el mundo, pero no a Inglaterra. El *hub* de España tendría sus inputs con los clientes a servir, pero seguramente la flota que se asignaría a esta aeronave sería de A380 si vuela a destinos muy lejanos. En cambio, en el input de Inglaterra las distancias equivaldrían a un A320. Así pues existe una parte manual en el algoritmo, y es la definición correcta de inputs factibles operacional y legalmente.

- También se debe entender que en un inicio no se trabajará con demanda y capacidad en números, sino que se definirá una red de aeropuertos. Un input está compuesto por el *hub* que sirve a los clientes que aparecen en ese mismo documento .txt.

Es decir, si el primer nombre del fichero es Barcelona y luego aparecen Madrid y Sevilla, quiere decir que el *hub,* es decir el aeropuerto servidor, es Barcelona y los otros son clientes que se deben servir sin importarnos la demanda y la oferta que tengamos. Si aparecen en el input es que nos interesa ofertarlos y que hay suficiente demanda para ofertar y volar hacia esos destinos.

- El coste de cada vuelo será calculado por el algoritmo a partir de un fichero de inputs que determina la posición por coordenadas de cada aeropuerto, por lo que, en esta versión el coste es igual a la distancia.

- Se habla de ciclos en el problema de asignación de aeronaves porque una vez que una aeronave sale de un depot estará en constante vuelo (sin contar tiempos de *turnaround* y tiempos no operativos por paros de los aeropuertos) hasta llegar otra vez al *depot* donde puede permanecer parado por restricciones de mantenimiento. Según algunos estudios los ciclos deben situarse en un conjunto de entre tres y seis legs. De esta manera se minimiza el impacto a esa ruta si se sufre algún retraso, fallo de motor o cualquier otro error. Ya que dicho error impactara en menos aeropuertos. Este algoritmo trabajará con ciclos de seis aeropuertos.

1.1.4.3 Algoritmo y heurística de *savings*

Se usará, por lo tanto, una heurística conocida como *Cost Saving Heuristic (CSH).*
En este apartado se presenta un algoritmo en formato de pseudocódigo que explica la lógica detrás de dicha heurística:

Step 1. Savings computation

- Compute the savings $sij=ci0+c0j-cij$ for $i,j=1,\ldots,n$ and $i\neq j$.
- Order the savings with a probabilistic equation.

Step 2. Best feasible merge (Parallel version)

Starting from the top of the savings list, execute the following:

- Given a saving *sij*, determine whether there exist two routes that can feasibility be merged:
  - One starting with (0,*j*)
  - One ending with (*i*,0)
- Combine these two routes by deleting (0,*j*) and (*i*,0) and introducing (*i*,*j*).

Step 3. Route Extension (Sequential version)

- Consider in turn each route (0,*i*,…,*j*,0).
- Determine the first saving *ski* or *sjl* that can feasibly be used to merge the current route with another route ending with (*k*,0) or starting with (0,*l*).
- Implement the merge and repeat this operation to the current route.
- If not feasible merge exists, consider the next route and reapply the same operations.
- Stop when not route merge is feasible.

El algoritmo se divide en tres pasos.

En el primero se calcula lo que se ahorra en el trayecto si vas de un cliente a otro con un mismo avión, y desde el *hub* correspondiente. Esto lo hace a través de sumar los costes de ir a un cliente más los de volver del otro cliente, restándole los costes de ir de un cliente a otro. Si el resultado es positivo es que esa ruta te interesa unirla entre clientes. Se asume que el coste de ir a un cliente es igual al de volver a un cliente.

Supongamos que ir al cliente 1 cuesta 20 y volver cuesta 20. Ir al cliente 2 cuesta 30 y volver cuesta lo mismo. En total, si usamos la *dummy solution* se obtendrán costes por valor de: 20*2+30*2=100.
Lo que hace la heurística en el primer paso es calcular cuanto nos ahorraríamos si uniéramos esos dos clientes. Por ejemplo, si ir de uno a otro cuesta 10, el ahorro sería de:
20+30-10=50-10=40. En total ahorraríamos 40 unidades. Esto se calcula de sumar los costes de los viajes que no haríamos (una ida y una vuelta a cada uno de los clientes) y restarle el coste de ir entre los clientes (lo que pagamos para hacer una conexión).

Este proceso lo hace para todos los nodos hasta tener una lista ordenada de ahorros, lo que permite establecer un orden de preferencia. Claro está que al aplicar uno, otro puede quedar inservible.

Seguidamente, en el paso dos se inicia la unión de rutas. Se mira según el orden de la lista qué rutas nos interesa unir y, si es factible unirlas, se eliminan un camino de ida de un nodo y otro de vuelta del otro nodo al que uniremos, y se aplica el viaje entre nodos.

El tercer y último considera si en cada ruta unida aún se podrían incorporar más nodos. Para eso se debe comprobar según la lista de *savings* cual nos interesa unir, y verificar que es factible unirlas.

1.2. Objetivos generales y parciales del proyecto

1.2.1. Objetivos generales

Apoyado en las guías bibliográficas y en las técnicas desarrolladas previamente por otros expertos en el sector, se pretende **desarrollar un algoritmo que permita la correcta y óptima creación de rutas y, consecuentemente, minimice la asignación de aeronaves a los itinerarios. Estas rutas deben maximizar el uso de las aeronaves en estas trayectorias, por lo que se estarán minimizando el número de aeronaves requeridas para cubrir un itinerario.**
A su misma vez, estas rutas deben asegurar una minimización de costes respecto la *dummy solution*.

Este algoritmo permitirá la ejecución de una heurística que evaluará en qué momentos conviene conectar dos aeropuertos para minimizar los costes operacionales. Con esto se espera obtener un resultado más acurado y más óptimo de la secuencia de legs que debe formar cada ruta.

1.2.2. Objetivos parciales

Con el fin de cumplir el objetivo general del proyecto se deben definir una secuencia de objetivos parciales, que aportarán importante información en la consecución del objetivo global.

1. Estudiar el problema de creación de rutas y asignación de aeronaves: En la fase inicial del proyecto, se deberán estudiar el conjunto de técnicas usadas en la creación de rutas y en la asignación de aeronaves, con el fin de observar los puntos fuertes y las carencias de este, además de ver qué importantes factores les afectan. Un correcto estudio del proceso ayudará en las posteriores fases, puesto que al maximizar el conocimiento del proceso se evitará caer en errores simples.

2. Definición del proceso para solucionar el problema: En esta fase se aplicarán los conocimientos adquiridos en la fase previa conjuntamente con algunos conocimientos desarrollados por expertos en esta temática, para así crear un nuevo modelo que plantee una nueva manera de solucionar el problema.

3. Desarrollo de un algoritmo*:*

   Este objetivo parcial es la clave del éxito del proyecto y en la que se basará una gran parte del peso de este. Exactamente, consistirá en la creación de un algoritmo que compute y calcule las rutas a volar por las aeronaves en función de una serie de inputs y que, además, contemple un conjunto de restricciones que, en caso de no ser cubiertas, implicarían que no se está simulando el sistema real en el que, posteriormente, se podría aplicar la solución obtenida.

   Este algoritmo, a través de una red de aeropuertos cómo principales inputs, calculará una solución factible al problema explicado.

4. Diseñar y ejecutar un conjunto de experimentos que permitan evaluar el potencial de las técnicas usadas: A través de un determinado número de experimentos y modificando ciertos inputs, comprobaremos la potencia de las técnicas usadas y la mejora que podría experimentar el sistema real, consiguiendo así evaluar si es una buena herramienta para el susodicho sistema y extrayendo una serie de conclusiones.

SECCIÓN 2: BASE TEÓRICA Y EVOLUCIÓN DEL PROBLEMA

2.1 Marco conceptual

Para entender realmente el problema se debe establecer un marco conceptual que describa en qué consiste este proceso y qué actividades tiene asociadas.

El conjunto de actividades que componen el proceso de planificación de una compañía aérea generan el pilar central de su estrategia al fijar el conjunto de mercados a los que dirigirse, con qué frecuencia servirlos y en qué horarios hacerlo. Por supuesto, todo esto contempla unas serie de restricciones tanto internas y externas que tiene la aerolínea.

Es importante entender que la bibliografía se ha basado en artículos de *Air fleet assignment*, pues nadie ha planteado el problema aportando una mejora a las fases previas. Estos artículos han ayudado a entender las metodologías, modelos y aproximaciones usadas, consiguiendo así dar un nuevo enfoque al problema a través de este problema.

2.1.1. Planificación del programa de vuelos

A partir de los siguientes datos: análisis histórico de rutas, cambios en el entorno competitivo y en la capacidad de los mercados, los planes de crecimiento de la compañía aérea y otros criterios y previsiones estratégicas, se definen un conjunto de itinerarios a operar. Esto implica definir los aeropuertos que estamos interesados a cubrir, además de, posteriormente, definir horas de salida, llegada y frecuencias de vuelos.

El resultado de esta actividad define el plan comercial de la aerolínea, por lo que es la base en la que se basan los principales ingresos de la aerolínea y, además, define la competitividad de la empresa. Esta es una actividad que se lleva a cabo con una antelación de entre 18 y 12 meses antes de la salida de los vuelos.

Como se ha mencionado anteriormente, tanta antelación en las previsiones hace que en el día de operaciones la demanda real difiera mucho de la prevista, por lo que puede hacer que un vuelo que era óptimo operacional y económicamente para la empresa, ahora sea lo contrario.

La programación de vuelos es el elemento fundamental sobre el que se sustentan el resto de actividades de planificación de cualquier compañía aérea.

2.1.2. Asignación de flotas

Una vez se ha definido el conjunto de vuelos a operar, hay que asignar un tipo de avión a cada vuelo. El objetivo, actualmente, es maximizar los ingresos de la aerolínea, aunque en este proyecto se trabajará una concepción diferente, la de minimizar los costes a través de la maximización del uso de aeronaves.
El objetivo real es buscar una asignación óptima de capacidad (número de asientos) en cada vuelo, en relación a la demanda de pasajeros prevista. Aunque en este proyecto no se trabajará esta perspectiva, sino que se trabajará la de a partir de programar las rutas asegurar que se minimiza la flota necesaria a asignar, facilitando así dicha tarea.

Se puede decir que la asignación que se trabaja en este proyecto es como un punto intermedio entre las fases de programación de vuelos y asignación de flotas.

El problema de asignación de flotas real contempla, además, restricciones como que la flota está limitada a los tipos de aeronave de que dispone la compañía y también a la necesidad de conservar un balance de cada tipo de avión en cada aeropuerto base. Este último, es un factor muy relevante de cara a garantizar que el programa de vuelos sea operativamente factible, ya que permite garantizar la disponibilidad de la misma cantidad de cada tipo de avión en cada uno de los aeropuertos en los que la compañía aérea tiene una base.
En cambio, con la heurística planteada y explicada anteriormente, este algoritmo trabaja siempre sobre una flota infinita y, simplemente, procura minimizarla.

2.1.3. Planificación del mantenimiento y matriculación de vuelos

Una vez ya se ha asignado el tipo de avión que debe operar cada vuelo, se debe asignar qué avión exactamente operará cada vuelo.
Así pues, se contemplan restricciones como las necesidades de mantenimiento y revisiones obligatórias para cada avión, así como los tiempos de escala necesarios.

En esta fase se consigue determinar qué avión matriculado vuela cada ruta, cumpliendo así con las restricciones mencionadas.
Esta es la última fase del proceso, y a menudo se realiza una semana antes del vuelo.

Esta última parte no se trabajará tampoco en este algoritmo.

2.1.4. Planificación de tripulaciones

Esta actividad tiene por objetivo asignar las tripulaciones (tanto auxiliares como técnicos) que han de operar cada uno de los vuelos planificados en la etapa de programación de vuelos, de forma que se minimicen los costes.

Le afectan una gran cantidad de restricciones, tanto operacionales como legales, que dificultan realmente una solución óptima y por lo cual muchos investigadores han trabajado en este problema.
Para llevar a cabo esta actividad se deben tener en cuenta aspectos como los períodos de actividad máximo que regulan los convenios colectivos, los descansos obligatorios, periodos no lectivos o de formación, certificaciones de tipo de aeronave, bases...

Esta fase se lleva a cabo alrededor de un mes antes del día de operación.
En este proyecto no se trata esta fase.

2.2. Revisión de la literatura

2.2.1. Evolución histórica del problema

Con el claro propósito de conseguir los objetivos planteados en este proyecto y un correcto estudio de la problemática es necesario disponer de una visión cronológica sobre cual ha estado el origen del problema, las metodologías usadas a lo largo del estudio hecho por otros investigadores y en que estado se encuentra actualmente.

Son muchos los investigadores que han planteado el problema de la programación de vuelos y sus subsiguientes fases derivadas. Consecuentemente, el problema ha sido tratado con diferentes puntos de vista a lo largo de la duración de este, pero últimamente se ha apostado muy fuerte por un cambio en el planteamiento y en las técnicas usadas, simplemente con el fin de aprovechar técnicas recientes.

Para encontrar las primeras referencias al problema de la programación de vuelos, podemos retroceder hasta un artículo denominado “Airline Scheduling: An Overview” publicado el año 1984 por Etschmaier y Mathaisel (Etschmaier y Mathaisel (1984)). En este artículo ya se habla de un problema emergente, tratado por los diferentes operadores aéreos con tal de ajustar la capacidad con la demanda.

Posteriormente, Abara, J. en el año 1989 en su articulo denominado “Applying integer linear programming to the Fleet Assignment Problem (FAM)” evalúa por primera vez el problema desde un punto de vista matemático y puramente computacional. Para ello usa técnicas de MIP (Mixed integer programming), con el fin de asignar únicamente un avión a cada salto y evaluando restricciones en el modelo matemático como son, por ejemplo, el número total de aviones disponibles, capacidad, demanda, y otros factores relevantes. Además, ya empezó a contemplar en sus restricciones las políticas de mantenimiento de las aerolíneas.

En otras palabras, se puede decir que Abara fue el precursor de tratar el problema computacionalmente, y muchos investigadores posteriores han seguido optimizando el modelo y metodología presentada por él.
Vemos pues que Abara trabajó puramente el problema de asignación de aeronaves, sin contemplar fases previas. Es decir, intentaba solucionar el problema sin intentar optimizar las fases previas.

El primer cambio sustancial en el planteamiento de la problemática se produjo diez años después, planteado por Loachim, concretamente en el año 1999, y enfocaba el problema de la asignación de flota con ventanas de tiempo. Básicamente, este enfoque añadía restricciones al modelo para sincronizar los tiempos de salida de un vuelo en distintos días de la semana. El propósito de estas restricciones era el de facilitar las tareas del enrutamiento de aeronaves. El cambio viene en que Loachim empieza a retroceder a fases anteriores de la asignación de vuelos. Es decir, se da cuenta que para asegurar una buena asignación de aeronaves es importante hacer una buena planificación de rutas, aunque no trabaja al 100% esta fase. Simplemente manipula los horarios creados en esta fase.

Cuatro años después, Belanger mejoró un poco más este modelo, además de evaluar ventanas de tiempo, evaluó que los ingresos dependían del tiempo, es decir el ingreso de un vuelo depende de la hora en la que vuela. Además, cambió la metodología usada a un algoritmo de branch-and-price, demostrando que era más óptima esta metodología que no la de branch and bound, ya que consiguió reducir el número de aeronaves usadas.

Luego en el año 2004, Jay M. Rosenberg planteó un modelo totalmente distinto a lo visto hasta el momento. Las aerolíneas siempre se basan en el escenario más optimista, y el que rara vez ocurre. Un vuelo no se cancela. El problema de esa concepción es que si se cancela un vuelo, también se cancela el ciclo completo que lo conlleva. Es decir, si se vuela de Madrid a Barcelona, de Barcelona a Paris y de Paris a Madrid, pero se cancela el primero de ellos, los otros tres también se cancelan. Esto afecta a todo el sistema, pues no solo se debe recolocar los

pasajeros del primer vuelo, sino también los de los otros. Por supuesto, se aprecia que también se rompe la asignación de la flota de los tres *legs*.

El modelo desarrollado, pues, se basa en trabajar en ciclos cortos de vuelos. Esto permitirá minimizar las disrupciones. No es lo mismo tener que cancelar un ciclo de 3 vuelos que un ciclo de 8 vuelos. Los afectados serán muchos más y, consecuentemente, la aerolínea y su planificación se ve más mermada.
Por otra parte, otro detalle que se contempla es la *Hub connectivity,* si un ciclo empieza en un hub y termina en otro hub, y otro ciclo es inverso a este, cuando uno de esos vuelos falle, la disrupción alterará ambos ciclos. Para evitar eso, el autor presenta la idea de que los ciclos deben empezar y terminar en el mismo hub, esto ayuda a mantener el equilibrio de la red y, consecuentemente, si falta un vuelo de un ciclo, ese hub puede cubrir los otros vuelos, por lo que minimiza el impacto y da alternativas.

Como se verá más adelante, estas ideas han sido aplicadas en el modelo que se desarrolla a lo largo de este proyecto, sobretodo la del *Hub isolation.* Esta, básicamente, es la que determina que cada ciclo debería empezar y finalizar en un mismo hub, aislando así los ciclos entre hubs y evitando interdependencia entre ciclos aparentemente independientes. Por lo tanto, se puede afirmar que el cambio de concepción que plantea Rosenberg es el de que si se quiere solucionar el problema no siempre debe trabajarse sobre el escenario optimista.

En ese mismo año, los investigadores Lohatepanont y Barnhart plantearon el problema de la programación de vuelos y la asignación de flota conjuntamente. Aquí se ve la relevancia que tuvo el modelo de Belanger, ya que demostraba que no se podía tratar la asignación sin evaluar los horarios e itinerarios que se ejecutaban, pues afectaban la demanda e incluso otros factores. Esta ha sido la única vez en la que se han tratado las dos fases conjuntamente.

Su hipótesis era que si se quitaba un vuelo en la programación de vuelos, la demanda de los vuelos restantes programados cambiaba.

El punto más importante a considerar y que iba más allá respecto el modelo de Belanger era que la demanda de pasajeros no solo se veía afectada por un simple vuelo, sino que también por el conjunto de saltos que debía coger ese pasajero hasta llegar a su destino final.

Por lo tanto, puede pasar que para la demanda de cada pasajero, un simple cambio en uno de los vuelos que debe coger para llegar hasta su destino afecte a los saltos restantes.

Por lo general, el problema de la programación de vuelos ha sido solucionado por un período de 90 días. En este proceso tan largo, un gran conjunto de eventos inesperados puedes suceder (retrasos, congestión de aeropuertos, dificultades técnicas) y la demanda puede cambiar.
Por lo tanto, una programación que es óptima bajo unos ciertos supuestos puede transformarse en sub óptima en frente de otro escenario, o incluso no factible operacionalmente. Es muy interesante con tal de dotar al modelo de una mayor robustez evaluar los factores de incertidumbre en esos inputs que pueden presentarla.

Por eso en el año 2005, un grupo de investigadores liderados por Venkata L. ejecutaron un experimento basado en computación estadística, con el fin de acercarse a una solución realista del problema del FAM.
Para ello, desarrollaron el concepto de “Demand driven Dispatch (D3)” con el fin de cuadrar la demanda de un avión cerca de la fecha de salida del vuelo. Este proceso usaba dos fases de toma de decisiones:

- La primera se desarrollaba 90 días antes del día de salida, justamente cuando la programación de vuelos se publica. En esta fecha se asignaban los aviones que poseían un mismo modelo de *cockpit* a la tabla de tiempo de la aerolínea, ya que estos eran los que podían ser manipulados por una misma tripulación.

- La segunda fase se lleva a cabo 2 semanas antes del vuelo, cuando un alto porcentaje de la demanda se ha realizado. En esta fase se usan técnicas de ***swapping***.

Un ejemplo de este proceso sería que para un vuelo en la fase 1 le asignamos el Boeing 757 y el Boeing 767, que tienen el mismo *cockpit* y por eso cualquier tripulación que sea apta para uno de los dos podrá llevar el otro.
Posteriormente, en la segunda fase vemos que la demanda realizada es más elevada que la esperada, entonces en ese caso dejaremos asignado a ese vuelo el Boeing 767, ya que presenta una mayor capacidad. De esta manera facilitamos las tareas de asignación y podemos ajustar cerca del día de operación, minimizando el efecto de la incertidumbre que todos los otros modelos no evaluaban de ninguna manera.

Este planteamiento que desarrollaron los investigadores liderados por Venkata otorga una buena idea de lo que puede hacerse posteriormente con un algoritmo de estas características.

Un año después, en el 2006, Hanif, D. Sherali y Xiaomei Zhu añadieron un grado más de dificultad en la resolución del problema. Ellos enfocaron su trabajo en un entorno aún más realista, incorporando a su modelo la demanda estocástica de pasajeros. Para ello, utilizan el conocido *network effect,* este básicamente indica que dependiendo de la conectividad entre diferentes vuelos la demanda varía (en un modelo de *hub and spoke*). Esta idea se basa en que la demanda real se ve afectada por el conjunto de los vuelos y no sólo por el vuelo que cogerá el pasajero en cuestión. Por ejemplo, si hay 5 vuelos que se dirigen a nuestro Hub, los cuales presentan un *load factor* del 50%, no se puede suponer que el vuelo que sale del Hub hacia otro aeropuerto llegará a tener un 100% de *load factor* aunque nuestras previsiones así lo indiquen.

En otras palabras, hay interdependencia entre los diferentes *legs* y eso afecta al conjunto de la demanda de cada vuelo. Por eso no se puede trabajar con valores

esperados de demanda, sino que se debe trabajar con valores próximos y, por lo tanto, en los cálculos se debe poder demostrar la variabilidad de ese valor.

Para representar todas estas observaciones en su modelo, los autores consideraron varios mercados potenciales, es decir, escenarios, para dar la mayor flexibilidad posible al proceso de asignación de aeronaves.

Así pues, siguieron un proceso parecido al ya existente hasta el momento, pero aportando una mayor flexibilidad al modelo a través de la contemplación estocástica de la demanda.
Para ello usaban 2 fases, la primera contemplaba diferentes escenarios y los solucionaba. La segunda fase era una fase de optimización para cada escenario en la que se llevaba la solución hasta sus límites, cada escenario contemplaba la asignación de una flota.

Los resultados fueron muy interesantes, pues mejoraron un 1,7% los beneficios. Lo que se traduce en 120 millones de dólares al año. Los tiempos de computación también eran razonables por los equipos informáticos de aquel entonces. Tardaban entre 4 y 12 horas a solucionarlo. Lógicamente, el modelo presentaba muchas restricciones y un volumen de información muy grande, por lo que el tiempo de computación no es nada exagerado, más si se contempla la mejora económica producida. Por ejemplo, se contemplaron 900 *legs* y 1814 itinerarios , con un total de más de 250 aviones.

De este modelo podemos extraer la importancia de trabajar en ambientes estocásticos, ya que se pueden dar las peores situaciones o las mejores, optimizando aún más los resultados. Esta investigación ayuda a que se valore como una futura línea de trabajo muy importante la de convertir este proyecto en estocástico. Es decir, identificando las distribuciones estadísticas de los distintos inputs e incorporándolos al algoritmo.

Posteriormente, uno de los más reconocidos investigadores en este ámbito, Barnhart, realizó un artículo el año 2012 llamado “Quantitive problem solving methods in the airline industry”.

Él planteo la dificultad real que sostenía la resolución de un problema de estas características, y es que las soluciones rentables requieren previsión de las condiciones generales del mercado: precio del combustible y de las tripulaciones, así como el nivel y la naturaleza de la competencia.

Las aerolíneas abordan muchos temas de programación (asignación de aviones y tripulaciones a un vuelo, enrutamiento de aviones a las bases de mantenimiento) con técnicas de optimización combinatoria a gran escala. El tamaño de las aerolíneas de hoy en día hace que cada vez sea más difícil este problema.
Por ejemplo, las grandes compañías nacionales de Estados Unidos operan más de 3.000 vuelos al día con 600 o más aeronaves y pueden incluir más de 300 ciudades, sirviendo a más de 10.000 mercados únicos.

El artículo planteaba una solución sólo al problema de la generación de itinerarios, y no evaluaba la flota asignada a cada salto.
En primer lugar, un algoritmo de generación de itinerarios se utiliza para construir itinerarios entre cada par de aeropuertos, utilizando datos del programa de la compañía aérea.

Para un día determinado, un par de aeropuertos puede tener una alta frecuencia de vuelos, cada uno de los saltos ofrece a los pasajeros una posible manera de viajar entre los aeropuertos. Aunque la lógica utilizada para construir itinerarios difiere a través de las líneas aéreas, en los algoritmos generales se incluye la lógica de la distancia para eliminar itinerarios irrazonables y horas de conexión mínimas y máximas, para garantizar que las conexiones no realistas no estén permitidas. Además, los itinerarios suelen ser generados para cada día de la semana, para así evaluar que entre los días de la semana hay diferencias entre los vuelos ofertados.

Una vez generado el conjunto de itinerarios que conectan un par de aeropuertos, un algoritmo que evalúa el mercado predice el porcentaje de viajeros que eligen cada itinerario en un par de aeropuertos.

A continuación, la demanda en cada itinerario se determina multiplicando el porcentaje de los viajeros que se esperaban en cada itinerario por el tamaño del mercado previsto, o el número de pasajeros que viajan entre un par de aeropuertos.
Como se puede apreciar, la evolución entre metodologías se ha derivado de los avances científico-tecnológicos, que dotaban de unos mejores conocimientos a los investigadores. Además, desde un inicio se han unificado las problemáticas de generación de itinerarios, es decir, determina aeropuertos, frecuencias y horas de salida y llegada, con el problema de la asignación de vuelos. Aunque cómo en este último caso, algunos investigadores han decidido afrontar uno de los problemas para intentar llegar a la solución óptima.

Esto cómo demuestra Barnhart es normal, pues dado el gran tamaño de las aerolíneas de hoy en día, el esfuerzo computacional requerido para solventar el problema es muy grande, por lo que evaluar en un sistema real la solución se hace muy difícil.

2.2.2 Conclusiones de la bibliografía

Con toda esta bibliografía se han podido concluir distintas cosas que ayudan al desarrollo del proyecto:

- El problema real a solucionar depende de muchas variables y, además, está formado por cuatro grandes pasos. Todo esto hace que sea imposible solucionar todos los pasos conjuntamente. Es clave, pues, entender que hasta que la tecnología no mejore no se podrá solucionar el problema real. Al inicio, se solucionaba cada paso por separado. A medida que avanza la tecnología es más fácil añadir factores al problema y acercarse a una solución real, además de podernos ayudar a través de crear distintos planteamientos al problema.

- Es importante que, en medida de lo posible, la solución se plantee desde un inicio, es decir, no se intente solucionar la asignación de flotas sin haber optimizado la programación de rutas/itinerarios, ya que cada fase depende de la anterior. Es decir, la planificación estratégica de una aerolínea sufre un efecto cascada.

- El modelo planteado por este proyecto es distinto a todo lo actualmente planteado, por lo que es imposible comparar los resultados con otros proyectos.

- En el futuro se puede mejorar este proyecto añadiendo factores estocásticos a los inputs, para alterar los resultados y no trabajar en escenarios cerrados.

Hace falta mencionar que al tener una limitación temporal mucho más estricta que las investigaciones de estos artículos, además de que tampoco se ha tenido el soporte de ninguna base de datos o aerolínea que permita probar con *inputs* realistas y verificar que en redes de aeropuertos reales funciona la metodología estudiada.
Se ha tenido que redimensionar el problema para poder aportar una primera solución aproximada. Así pues, el éxito del proyecto depende de que cumpla con los objetivos previstos y mejore los resultados de la solución *dummy*.

SECCIÓN 3: METODOLOGÍA Y ALGORITMO

3.1. Metodología

Con el fin de cumplir con los objetivos definidos con anterioridad es necesario definir una metodología de resolución del problema planteado.
La metodología que se seguirá a lo largo del proyecto empleará técnicas de *biased randomization*, por lo que se obtendrán unos resultados que serán diferentes cada vez.

Este método no es muy distinto a una metodología determinista. Los dos reciben unos *inputs,* los cuales son gestionados por unas operaciones algorítmicas, y luego se obtienen unos *outputs* con los resultados. La diferencia se halla en que las técnicas de *biased randomization* permiten dotar a los resultados de cierta aleatoriedad, así pues se simulan distintos escenarios, lo que permite valorar distintas soluciones al mismo problema.

En este proyecto la *biased randomization* se implementa en el punto del algoritmo en el que se selecciona con qué *saving* empezar a gestionar la solución. El simple hecho de que cada *saving* no es combinable con todos los demás hace que dependiendo del que escojamos primero, se eliminen ciertas combinaciones y, por lo tanto, ciertas soluciones factibles. Este *saving* que escogemos depende de una ecuación matemática (explicada más adelante), por lo que en cada problema de *single depot* que se trata dentro de una ejecución, el *saving* escogido está en una posición distinta de la lista. Esto quiere decir que si tenemos 3 *depots* en el problema, sabemos que se solucionarán 3 problemas de *single depot*, por lo que en el primer *depot* la ecuación puede dar como resultado 1, en cambio, en el segundo *depot* puede dar 4, y en el tercero el valor puede ser 2.

Al mismo tiempo, es distinto en cada ejecución. Por ejemplo, en la primera ejecución el resultado obtenido de dicha ecuación puede ser 2, por lo que se escogerá el segundo valor de la lista, y en la segunda ejecución puede ser 2 o cualquier otro, por lo que la solución puede ser igual o distinta.

Además, se debe entender que una vez se ha incorporado el *saving* seleccionado a la ruta (si se puede incorporar), este *saving* se elimina de la lista para que cuando la ecuación que calcula el nuevo índice se ejecute, no pueda volver a coger el mismo. Así vemos que esta ecuación es llamada cada vez que se tiene que coger un *saving* de la lista.

La ecuación tiene la siguiente forma: Math.log(r.nextDouble()) / Math.log(1 - beta))

El parámetro beta de la misma es definido por uno de los ficheros que sirven como *input* al algoritmo. Este puede ir de 0 a 1. El otro parámetro llamado r.nextDouble() es un método nativo de java que permite escoger un número al azar.

Un ejemplo numérico sería el siguiente: ((Log(0.5034))/(Log (1-0.15)))= -0.2980/-0.07=4.22
Este 4.22 redondeado a entero se transforma en un 4.
Ahora con este número calculamos el módulo respecto el tamaño total de la lista, en java es la siguiente ecuación: (valor%tamaño.lista) —> 4%28=4.

La conclusión es que escogeremos el cuarto *saving* de la lista.

En cambio, si el valor de beta se acerca a 1, por ejemplo beta=0.99, el resultado sería el siguiente:
((Log(0.5034))/(Log (1-0.99)))=-0.2980/-2=0.149. Redondeado es un 0.
0%28=0. En este caso escogeríamos el primer *saving.*

Se observa que cuanto más cerca este el valor de beta a 1, menos efecto de *biased randomization* sufre el algoritmo. Es decir, el valor escogido siempre será similar y, además, será el primero. Por el contrario, cuanto más cercano a 0, más probabilidad hay de alejarnos de ese valor y explorar un conjunto de soluciones que empiecen por escoger un *saving* que ahorra menos. Esto no se traduce en que la solución será peor, ya que quizás la combinación final es más rentable que no la que se puede obtener si se escoge el primer *saving* siempre.

Es importante entender que se ha decidido aplicar estas técnicas en este proyecto porque ayudaba a simular distintos escenarios, es decir, soluciones más buenas al problema y otras más malas, pero al fin y al cabo soluciones.
Este paso no deja de ser una introducción y una pre preparación al uso de técnicas *simheuristics*, que son las que pueden ayudar a dotar realmente al algoritmo de estocasticidad, y las cuales deberían incorporarse a un futuro a este algoritmo.

3.2 Algoritmo

En este apartado se desarrollarán en forma de pseudocódigo las partes más importantes del algoritmo realizado.

Además, también se explicará el formato de los *inputs, outputs*, y algunos aspectos claves para entender el algoritmo en su totalidad.

3.2.1 Inputs

En total hay 5 tipos distintos de inputs.

El más sencillo es denominado random.txt. Es un fichero que solo determina si el algoritmo es ejecutado en su versión determinista o en una versión *random*.
Un 1 dentro del fichero quiere decir que el algoritmo se ejecutará de manera random y, por lo tanto, cada vez que se ejecute se obtendrá una solución distinta a pesar de tener siempre los mismos inputs.
Este random se determina mediante una ecuación que atribuye una probabilidad a cada *saving*, esta probabilidad representa la probabilidad de que se escoja un *saving* enfrente de otro, la fórmula ha sido explicada anteriormente.
Se debe recordar que al escoger uno, se imposibilita que se escojan muchos otros, ya que sólo se puede ir a cada aeropuerto una vez desde cada aeropuerto servidor

(*depot*), de esta manera existen bastantes combinaciones las cuales dependen del número de *depots* y clientes.

El segundo fichero es el de number_depots.txt, este es el que define cuantos depots introduciremos en el algoritmo. El número mínimo es 1, por lo tanto el problema sería uno de *single depot*, por el contrario sería *multi depot*.

Otro fichero muy importante es el de beta.txt. Este solo cobra relevancia si el fichero random.txt tiene un 1. El fichero beta determina el valor de una de las variables que se utilizan en la ecuación que define las probabilidades de escoger un *saving* enfrente de otro. El valor de beta va entre 0 y 1. Como valor inicial trabaja con 0,1.

Por último, tenemos los dos últimos tipos de ficheros. Si el problema es single depot, solo habrá dos ficheros de estas características. Si en cambio se trabaja con 2 depots, habrá 4 ficheros, si es con 3 habrá 6 ficheros y así consecutivamente.

El primero de estos ficheros, llamado 0_input_depot.txt, es el que incorpora las coordenadas de los aeropuertos. Las dos primeras coordenadas del fichero corresponden al *depot* y las siguientes a las de los clientes que debe servir dicho *depot*.
El fichero 0_input_names.txt trabaja en el mismo formato que el anterior, pero en vez de coordenadas incorpora los nombres, para así poder identificar en los outputs los aeropuertos de manera más fácil.

Si el problema debe solucionarse para dos depots, los ficheros para este tendrán los siguientes nombres:

- 0_input_depot.txt
- 0_input_names.txt

En cambio, los ficheros referentes al segundo *depot* tendría este nombre:

- 1_input_depot.txt
- 1_input_names.txt

En otras palabras, lo único que cambia entre ficheros que hacen referencia a distintos *depots* es el número con el que empieza el fichero.

3.2.2 Outputs

Una vez que se introducen los inputs en el formato correcto en la carpeta correspondiente y se ejecuta el algoritmo, conseguimos dos tipos distintos de outputs.

El primero hace referencia a la solución *dummy*, el fichero correspondiente se llama DummySolution.txt
Este incorpora los siguientes aspectos:

-Rutas *dummys* a servir (Depot —> Cliente —> Depot)
-Coste de cada ruta
-Coste total de las rutas
-Número de aviones usados para cubrir estas rutas.
-Tiempo de ejecución para conseguir la solución.

El segundo fichero, HeuristicSolution.txt, incorpora los mismos aspectos que el anterior, pero las rutas pasan a tener el siguiente formato:

Depot—> Cliente 1—> Cliente 2—> Cliente 3 —> Cliente 4—> Depot

Aunque no todas son así, ya que si no interesa unir algún aeropuerto a ninguna ruta este puede quedar aislando quedando el formato *dummy*:

Depot —> Cliente —> Depot

Vemos pues, que la solución optimizada puede incorporar la solución dummy en su interior. Pues la solución heurística es una optimización o mejora de la solución *dummy*.

3.2.3 Pseudocódigo

Una vez que sabemos qué debemos introducir en el algoritmo y qué resultados nos dará este, toca entender como funciona y como está estructurado.

En el anexo número 1 se puede encontrar el algoritmo en su totalidad y en formato normal, es decir, sin pseudocódigo.

Sin embargo, para facilitar el entendimiento de todo el proyecto, aquí desarrollaremos distintos apartados del algoritmo en formato de pseudocódigo.

En el algoritmo hay 14 clases. La gran mayoría son clases que dan cuerpo al algoritmo, es decir, representan los objetos que son necesarios para formar y plantear el problema. Estas clases son muy cortas y no presentan ningún tipo de complejidad, por lo que se explicará su usabilidad, y no en formato de pseudocódigo.

Clases de estructuración del algoritmo

Entre este tipo de clases podemos encontrar la clase Inputs.java, la cuál guarda los valores de los ficheros en la estructura de variables que se han creído oportunas.

Esta clase está estructurada en distintos objetos que son clases del mismo paquete. Por ejemplo, la clase Node.java define las variables que formarán los objetos del tipo Node, que serán los aeropuertos con nombre, coordenadas e ID.

Route.java es una clase que incorpora un conjunto de *edges* (más adelante explicados). Esta permite que se defina un objeto del tipo Route, que será la ruta dummy, es decir la ruta simple que luego se deberá optimizar.

Asignaciones.java es la clase en la cual se asigna la solución obtenida mediante la heurística, incluyendo el coste de dicha ruta y una lista de nodos, es decir, de aeropuertos.

La clase Solution.java es una clase que permite, una vez realizados los cálculos, crear un objeto que almacene la solución, es decir, la ruta y los valores representativos de la eficiencia de estos. Por ejemplo, el coste y una ID.

Finalmente, tenemos la clase Outputs.java. Esta es la clase que permite convertir las soluciones obtenidas por los algoritmos en ficheros de extensión .txt, para poder analizarlas a posteriori.

Clases de preparación de los cálculos del algoritmo.

En este otro tipo de clases, tenemos una que es de obtención de valores desde los ficheros .txt generados como *inputs.* Concretamente, esta se llama InputsReader.java.
A parte, hay una clase denominada InputsManager.java. En esta lo que se hace es llamar las clases de cálculo correspondiente que explicaremos en formato de pseudocódigo más adelante. Además, esta clase lo que genera son las listas de *savings* y de *edges* que se requerirán para hacer los cálculos. Los *savings* son conexiones entre dos clientes y un depot. En estos se indica cuanto te ahorras al conectar los tres entre ellos y no entre parejas.

En cambio, los *edges* son conexiones entre dos aeropuertos y el coste que implica moverse entre ellos.

El cálculo de los savings lo hace otra clase, concretamente la clase EdgeSavings.java, aquí se explica en pseudocódigo:

Step 1: Save the information

- Coge de uno de los savings el nodo depot y los dos clientes que se conectan con él.
- Guarda las coordenadas X y Y de cada uno de esos aeropuertos.

Step 2: Cost calculation

- Calcula el coste del depot al cliente 1 (coste 1)

  La fórmula usada es: (Math.sqrt((Xd - X1)*(Xd - X1) + (Yd - Y1)*(Yd - Y1))
- Calcula el coste del depot al cliente 2 (coste 2)
- Calcula el coste del cliente 1 al cliente 2 (coste 3)

Step 3: Saving calculation

- Calcula cuanto te ahorras -> coste 1 + coste 2 - coste 3;
- Guarda el valor en el saving correspondiente de la lista creada anteriormente en InputsManager.java

El cálculo de los *edges* lo hace la clase Edge.java, en formato de pseudocódigo sería lo siguiente:

Step 1: Save the information

- Guarda las coordenadas X y Y de ambos aeropuertos.

Step 2: Cost calculation

- Calcula el coste para desplazarse de un aeropuerto al otro.

  La fórmula usada es: Math.sqrt((X2 - X1) * (X2 - X1) + (Y2 - Y1) * (Y2 - Y1))
- Guarda el valor obtenido en la lista de edges.

Además, en esta misma clase se puede encontrar el método que calcula los edges de la solución dummy.

En formato de pseudocódigo el algoritmo hace lo siguiente:

- Mientras queden nodos:
    - Coge el *depot* i y sus coordenadas
    - Coge sus clientes
        - Mientras queden clientes:
            - En la posición x del string: Guarda como origen el aeropuerto *depot* y como destino el cliente x
            - Calcula su coste
            - Incrementa x
            - Cuando hayan finalizado los clientes incrementa variable i

Finalmente, la última clase a analizar es RandsCWS.java, en la cual se hacen los cálculos correspondientes para saber si interesa unir *savings* y *edges,* para conseguir minimizar costes y aeronaves en una ruta que sirva a todos los clientes desde el *depot* correspondiente. El pseudocódigo es:

- Mientras queden nodos en la lista:
    - Copia todos los *inputs* para poder manipularlos
        - Mientras queden savings:
            - Calcula el valor del índice i ->Math.log(r.nextDouble()) / Math.log(1 - beta))
            - Coge el *savings* de la lista en posición i y guarda los aeropuertos que lo forman.
            - Si contador = 0
                - Crea una ruta guardando estos aeropuertos en ella.
            - Si contador no es 0
                - Comprueba si interesa hacer un *merge* (unir) estos aeropuertos la ruta aún abierta.
                - Si interesa hacer el *merge,* añade los aeropuertos a la ruta.
                - Si no interesa, cierra la ruta.

        - Borra todos los aeropuertos y los savings que ya han sido usados para no repetirlos.
    - Cuando solo quede un nodo aeropuerto:
        - Mira si nos interesa añadirlo a la ruta actual, si interesa únelo.
        - Si no interesa, este nodo se cubrirá en una ruta dummy.
    - Guarda la ruta creada.
    - Incrementa el valor de contador y calcula el nuevo valor de i.
    - Calcula el coste de todas las rutas creadas y guárdalo.

El método que comprueba si interesa hacer un merge es el siguiente:

- Coge el nodo cliente 1 y el cliente 2.
- Copia la ruta actual y añádele ambos clientes.
- Calcula el coste de la nueva ruta en la variable: coste_ruta_heuristic
- Coge el *depot* de la ruta
    - Mientras queden clientes:
        - Coge el cliente x
        - Calcula el coste de ir desde *depot* al cliente x y su vuelta
        - Súmalo a la variable: coste_ruta_dummy
        - Incrementa x
    - Si coste_ruta_dummy >= coste_ruta_heuristic
        - Boolean merge=true;
    - Sino
        - Boolean merge=false;
    - Si el tamaño de la ruta heurística >=6 (Se priorizan los ciclos cortos)
        - Boolean merge=false;
    - Sino
        - Boolean merge=true;
    - Return merge

Como es lógico, en el algoritmo se busca que las rutas no tengan más de 6 *legs* en su interior. Los ciclos cortos son mejores como se ha verificado con algunos estudios de la bibliografía. Esto es así puesto que si hay un fallo o un retraso solo se propaga a pocos vuelos, minimizando las indemnizaciones a pagar. Así mismo, cuando se mira si interesa unir por costes, siempre nos quedamos con la ruta más barata o, en caso del mismo coste, escogemos la que minimiza las aeronaves. De esta manera se cumplen con los objetivos de minimización de costes y de aeronaves usadas.

SECCIÓN 4: EXPERIMENTOS Y ANÁLISIS DE RESULTADOS

4.1. Experimentos: documentación de resultados

4.1.1 Introducción

Una vez descrita la metodología de trabajo del proyecto, además de haber explicado a través de pseudocódigo las partes más significantes del algoritmo, se debe analizar el rendimiento del algoritmo desarrollado. Para ello usaremos una serie de inputs que permita analizar la eficiencia y la potencia del algoritmo, sin olvidar la calidad de dichas soluciones.

Para ello, los experimentos se basarán en la comparación de la obtención de resultados entre una maquina y un humano. Para ello, se ha ejecutado con 3 inputs distintos el algoritmo, demostrando que a pesar de incrementar el tamaño del problema y, consecuentemente, la complejidad, los tiempos de ejecución son más que razonables.

4.1.2 Resultados

4.1.2.1. Tiempos de ejecución

Los primeros resultados obtenidos y que permiten ver la eficiencia del algoritmo son los tiempos de ejecución del algoritmo. Sobretodo se observa la eficiencia del algoritmo en frente de una solución manual, si lo comparamos con el tiempo de obtención de un resultado a mano.

En cuanto a tiempos de ejecución por parte del algoritmo:

- Ha solucionado el problema con inputs de 3 depots y 2 clientes que servir en cada depot en 12 milisegundos. (Número 1 en la figura 13 línea roja)

- En 12 milisegundos también ha solucionado el problema con inputs de 3 depots, pero dónde uno de ellos servía 8 clientes y los otros servían solo 2 clientes. (Número 2 en la figura 13 línea roja)

- En cambio, para solucionar un problema de 3 depots, dónde 2 depots servían a 6 clientes y el otro sólo servía 4 clientes, el algoritmo ha hallado una solución en 10 milisegundos. (Número 3 en la figura 13 línea roja)

- La solución más rápida la ha hecho en 9 milisegundos, y corresponde a un problema de 3 depots y 4 clientes por depot. (Número 4 en la figura 13 línea roja)

Por el contrario, y para intentar verificar la eficiencia del algoritmo de solución en frente de una persona, se han solucionado a mano los dos problemas de menor complejidad obteniendo los siguientes resultados:

- Para el problema de 3 depots y 2 clientes, por cada depot se ha tardado 5 minutos y 52 segundos. (Número 1 en la figura 13 línea azul)

- Si la dificultad la aumentamos a 3 depots, dónde cada uno sirva 4 clientes, se obtiene un tiempo de ejecución a mano de 34 minutos y 53 segundos. (Número 2 en la figura 13 línea azul)

Es importante entender que la persona ha realizado sólo un par de ejecuciones a mano, puesto que, observando los resultados, la diferencia entre los tiempos de ejecución es más que notable, a pesar de que la complejidad usada por el humano no llega a ser tan alta como la del ordenador, el cual llega a trabajar hasta 8 clientes por depot sin incrementar el tiempo de ejecución.

Para ver la diferencia de tiempo en milisegundos entre el ordenador y el humano podemos apreciar la gráfica comparativa de los tiempos de calculo de debajo:

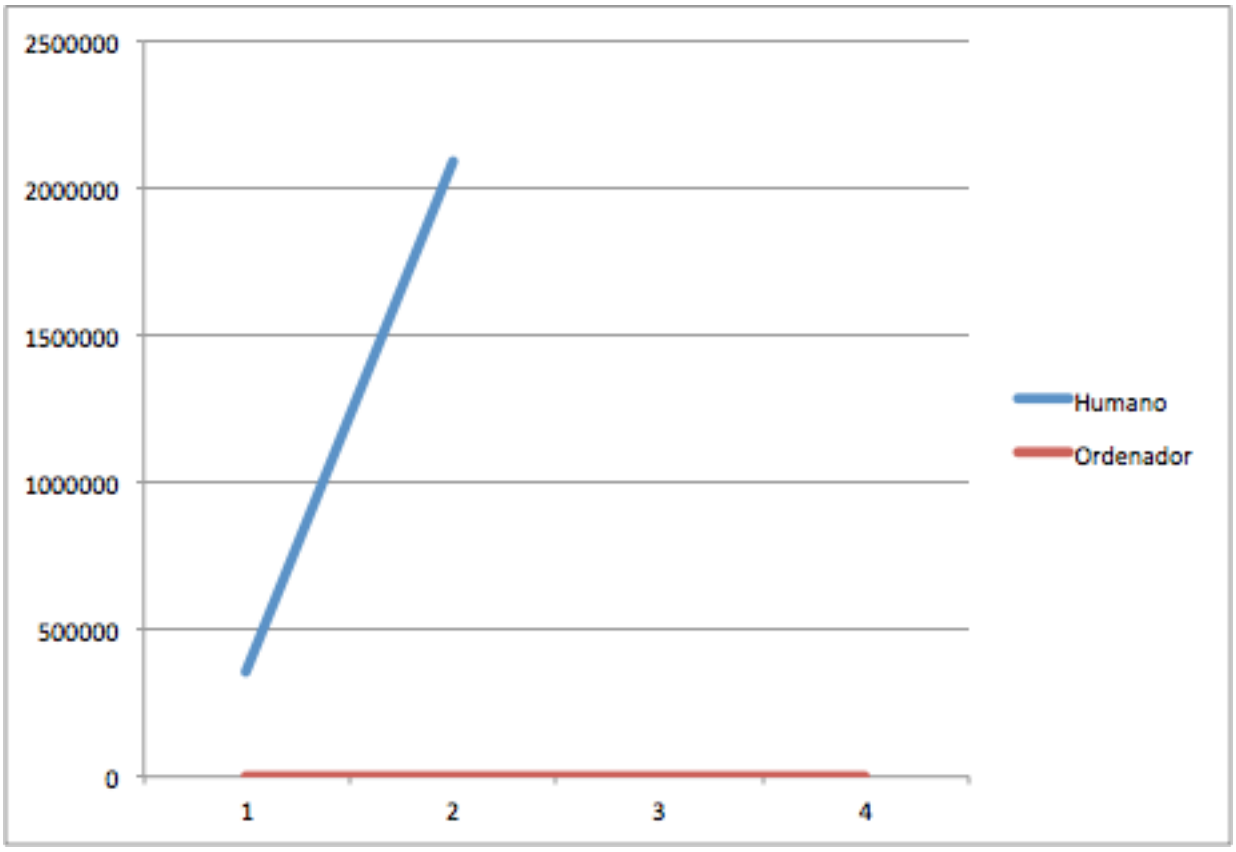


Figura 11: Comparación de tiempos de ejecución

Con la gráfica anterior, podemos observar que a pesar del incremento de dificultad de los inputs, el algoritmo lo ejecuta sin problema y sin incrementar el tiempo. De hecho, llega a disminuirlo y, por lo tanto, demuestra que la potencia de resolución del algoritmo es mucho mayor a la de resolución a mano por parte de un humano, puesto que este último en un simple incremento de dificultad sufre un crecimiento gigantesco y que hace inviable probar de solucionar a mano (de manera “rápida”) algunos de los inputs solucionados por el algoritmo.

4.1.2.2. Verificación de resultados

Para saber si el algoritmo funciona correctamente, debemos ejecutar con el ordenador el algoritmo con unos inputs que luego sean solucionados a mano. El resultado debería ser similar, no exacto, pues el algoritmo usa técnicas para dotar aleatoriedad a la solución.

Así pues, se ha trabajado con la solución factible más sencilla, unos inputs de 3*2. Esto quiere decir que hay 3 nodos depots que deben servir 2 clientes, cada uno de ellos. En este caso, la solución debería ser exacta, pues al sólo haber dos clientes por cada *depot,* solo existe un saving factible y, finalmente, una simple combinación que sería la solución.

La solución *dummy* obtenida con ordenador es:

---------- DUMMY SOLUTION ----------
----------RUTA 1 ----------
Coste ruta: 10.878924834476944
Barcelona --> Lisboa --> Barcelona
----------RUTA 2 ----------
Coste ruta: 18.799002535716905
Barcelona --> Milan --> Barcelona
----------RUTA 3 ----------
Coste ruta: 10.878924834476944
Lisboa --> Barcelona --> Lisboa
----------RUTA 4 ----------
Coste ruta: 14.280236540045422
Lisboa --> Milan --> Lisboa
----------RUTA 5 ----------
Coste ruta: 18.799002535716905
Milan --> Barcelona --> Milan
----------RUTA 6 ----------
Coste ruta: 14.280236540045422
Milan --> Lisboa --> Milan

El número de aviones usado es: 6
Coste total de la solución: 87.91632782047854
Se ha tardado 7.0 milisegundos en encontrar la solución

Y la solución heurística es:

---------- HEURISTIC SOLUTION ----------
----------RUTA 1 ----------
Coste de la ruta: 21.979081955119636
Barcelona --> Lisboa --> Milan --> Barcelona
----------RUTA 2 ----------
Coste de la ruta: 21.979081955119636
Lisboa --> Barcelona --> Milan --> Lisboa
----------RUTA 3 ----------
Coste de la ruta: 21.979081955119636
Milan --> Barcelona --> Lisboa --> Milan

El número de aviones usado es: 3
Coste total de la solución: 65.9372458653589
Se ha tardado 3.0 milisegundos en encontrar la solución

Además, se puede concluir que se ha reducido el número de aeronaves usadas en un 50% y el coste en un 25%.

Por el contrario, las soluciones correspondientes a mano son:

Solución dummy:

Barcelona --> Lisboa --> Barcelona
Barcelona --> Milan --> Barcelona
Lisboa --> Barcelona --> Lisboa
Lisboa --> Milan --> Lisboa
Milan --> Barcelona --> Milan
Milan --> Lisboa --> Milan

Coste total: 87,88
Tiempo: 2 minutos y 7 segundos

Solución heurística:

Barcelona --> Lisboa --> Milan --> Barcelona
Lisboa --> Barcelona --> Milan --> Lisboa
Milan --> Barcelona --> Lisboa --> Milan

Coste total: 65,91
Tiempo: 5 minutos y 52 segundos

Una vez visto esto, se verifica que la consecución de resultados con el algoritmo es el esperado y se puede incrementar la complejidad de inputs para el algoritmo, y ver las soluciones obtenidas.

Está solución dibujada gráficamente sería:

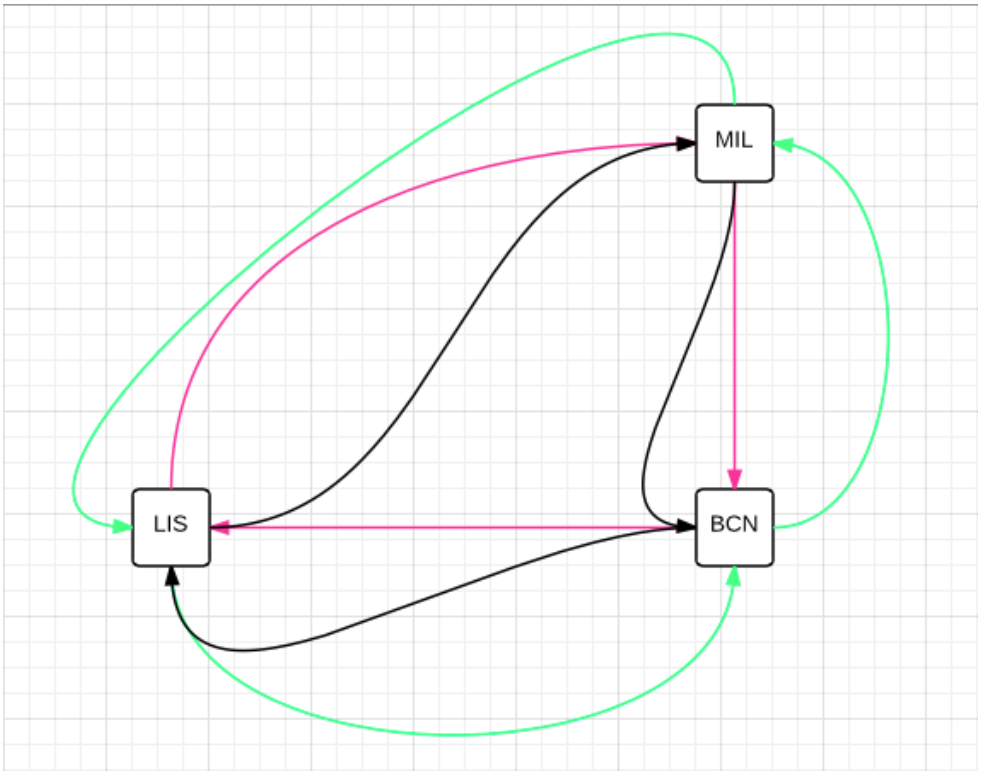


Figura 12: Solución al problema 3*2

4.1.2.3. Soluciones finales

Finalmente, y ampliando la complejidad de los inputs, se consiguen los siguientes resultados ejecutando el algoritmo.

Solución de un problema multi depot (3 *depots*) en los que dos *depots* sirven a dos clientes y el otro *depot* sirve a 8 clientes:

---------- DUMMY SOLUTION ----------
----------RUTA 1 ----------
Coste ruta: 10.878924834476944
Barcelona --> Lisboa --> Barcelona
----------RUTA 2 ----------
Coste ruta: 18.799002535716905
Barcelona --> Milan --> Barcelona
----------RUTA 3 ----------
Coste ruta: 30.563823023830498
Barcelona --> Paris --> Barcelona
----------RUTA 4 ----------
Coste ruta: 20.888807652813743
Barcelona --> Londres --> Barcelona
----------RUTA 5 ----------
Coste ruta: 11.67549242700649
Barcelona --> Madrid --> Barcelona
----------RUTA 6 ----------
Coste ruta: 29.299776939610606
Barcelona --> Dublín --> Barcelona
----------RUTA 7 ----------
Coste ruta: 21.65828476191328
Barcelona --> Vigo --> Barcelona
----------RUTA 8 ----------
Coste ruta: 22.04025539021759

```
 Barcelona -->  Frankfurt -->  Barcelona
----------------------------------RUTA 9 --------------------------
Coste ruta: 10.878924834476944
 Lisboa -->  Barcelona -->  Lisboa
----------------------------------RUTA 10 --------------------------
Coste ruta: 14.280236540045422
 Lisboa -->  Milan -->  Lisboa
----------------------------------RUTA 11 --------------------------
Coste ruta: 18.799002535716905
 Milan -->  Barcelona -->  Milan
----------------------------------RUTA 12 --------------------------
Coste ruta: 14.280236540045422
 Milan -->  Lisboa -->  Milan
--------------------------------------------------------------------------------
El número de aviones usado es: 12
Coste total de la solución: 224.04276801587076
Se ha tardado 7.0 milisegundos en encontrar la solución


 ---------------------------- HEURISTIC SOLUTION -------------------------
----------------------------------RUTA 1 --------------------------
Coste de la ruta: 52.75720073274139
 Barcelona -->  Lisboa -->  Paris -->  Londres -->  Madrid -->  Barcelona
----------------------------------RUTA 2 --------------------------
Coste de la ruta: 49.62416303891774
 Barcelona -->  Milan -->  Frankfurt -->  Dublín -->  Vigo -->  Barcelona
----------------------------------RUTA 3 --------------------------
Coste de la ruta: 21.979081955119636
 Lisboa -->  Barcelona -->  Milan -->  Lisboa
----------------------------------RUTA 4 --------------------------
Coste de la ruta: 21.979081955119636
 Milan -->  Barcelona -->  Lisboa -->  Milan
--------------------------------------------------------------------------------
```

El número de aviones usado es: 4

Coste total de la solución: 146.3395276818984

Se ha tardado 3.0 milisegundos en encontrar la solución

Vemos pues, que existe una reducción del coste del 34,7% y del número de aviones en un 66%.

Gráficamente la solución completa queda así:

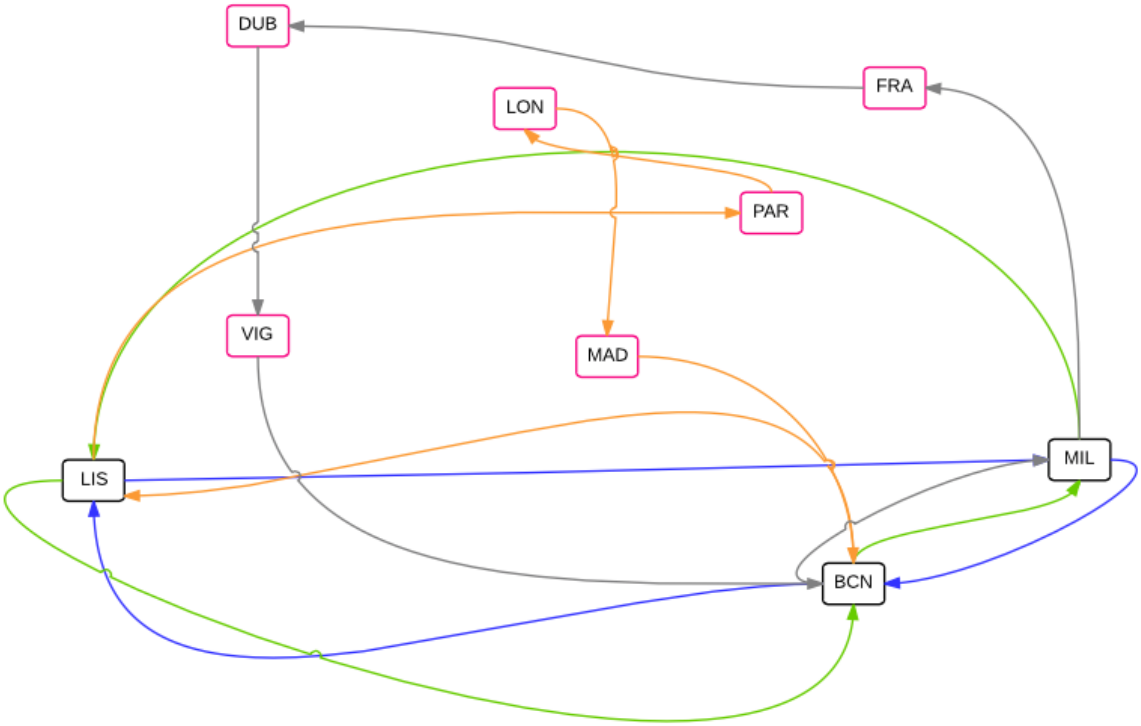


Figura 13: Solución completa al 2*2 y 1*8

De manera más detallada se pueden apreciar las siguientes rutas como solución definitiva:

Ruta 1:

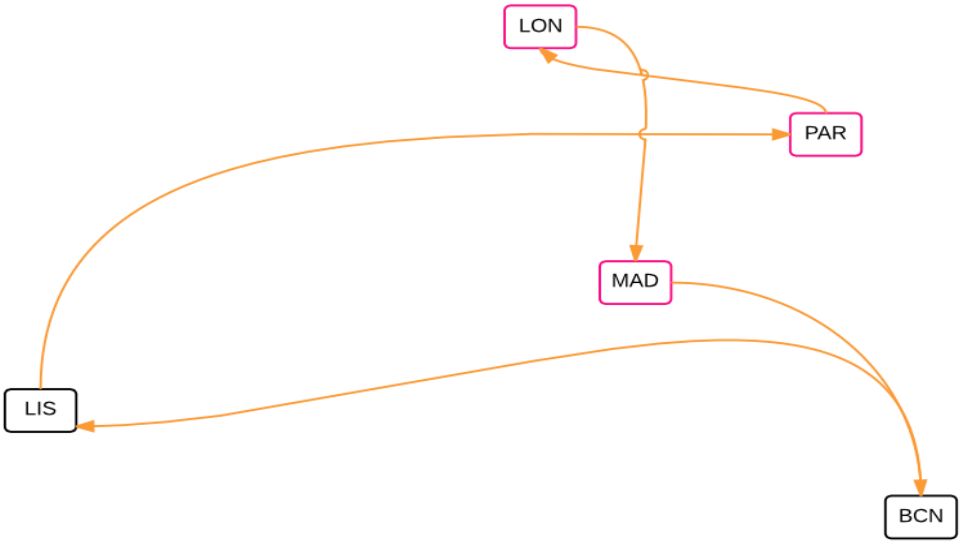


Figura 14: Ruta 1 detallada de la solución

Ruta 2:

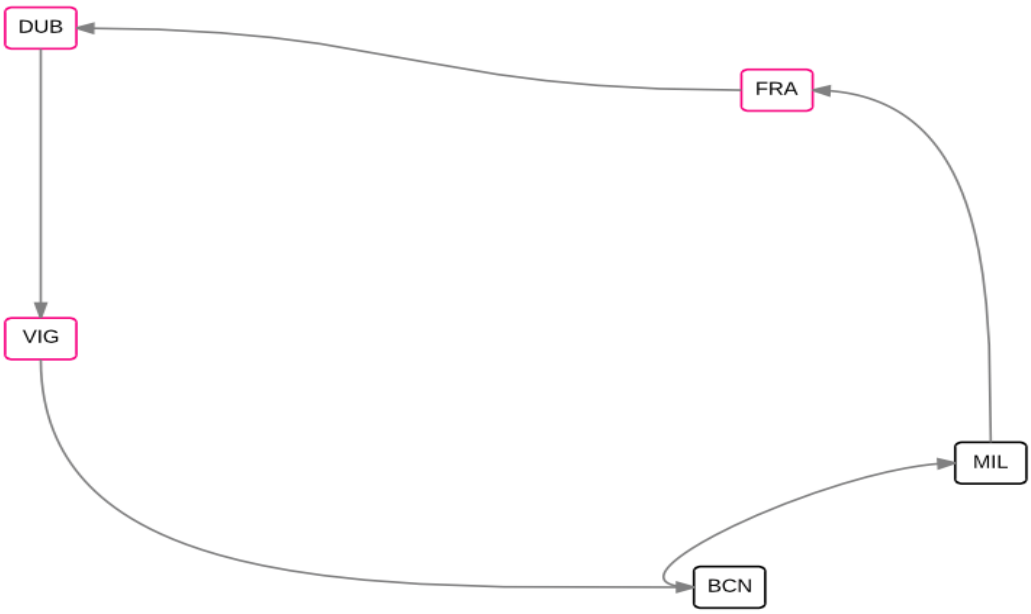


Figura 15: Ruta 2 detallada de la solución

Ruta 3:

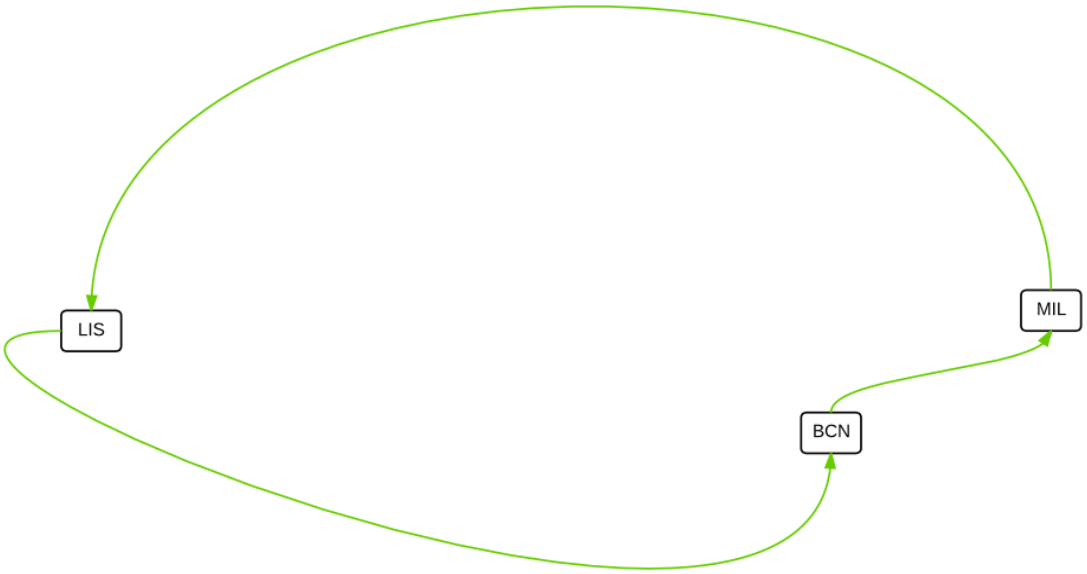


Figura 16: Ruta 3 detallada de la solución

Ruta 4:

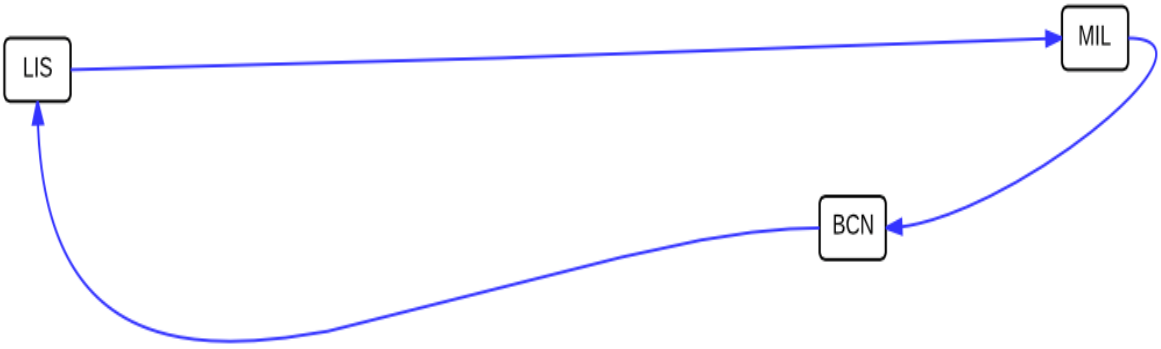


Figura 17: Ruta 4 detallada de la solución

Solución de un problema multi *depot* (3 *depots*), en los que dos *depots* sirven a seis clientes y el otro *depot* sirve a cuatro clientes:

---------- DUMMY SOLUTION ----------
----------RUTA 1 ----------
Coste ruta: 10.878924834476944
Barcelona --> Lisboa --> Barcelona
----------RUTA 2 ----------
Coste ruta: 18.799002535716905
Barcelona --> Milan --> Barcelona
----------RUTA 3 ----------
Coste ruta: 30.563823023830498
Barcelona --> Paris --> Barcelona
----------RUTA 4 ----------
Coste ruta: 20.888807652813743
Barcelona --> Londres --> Barcelona
----------RUTA 5 ----------
Coste ruta: 11.67549242700649
Barcelona --> Madrid --> Barcelona
----------RUTA 6 ----------
Coste ruta: 22.04025539021759
Barcelona --> Frankfurt --> Barcelona
----------RUTA 7 ----------
Coste ruta: 10.878924834476944
Lisboa --> Barcelona --> Lisboa
----------RUTA 8 ----------
Coste ruta: 14.280236540045422
Lisboa --> Milan --> Lisboa
----------RUTA 9 ----------
Coste ruta: 37.571401750429644
Lisboa --> Dublín --> Lisboa
----------RUTA 10 ----------
Coste ruta: 32.52475769750253

Lisboa --> Vigo --> Lisboa

----------RUTA 11 ----------

Coste ruta: 18.799002535716905

Milan --> Barcelona --> Milan

----------RUTA 12 ----------

Coste ruta: 14.280236540045422

Milan --> Lisboa --> Milan

----------RUTA 13 ----------

Coste ruta: 23.238621049898406

Milan --> Zaragoza --> Milan

----------RUTA 14 ----------

Coste ruta: 12.883332454258609

Milan --> Roma --> Milan

----------RUTA 15 ----------

Coste ruta: 4.988090254578137

Milan --> Múnich --> Milan

----------RUTA 16 ----------

Coste ruta: 59.24350983045527

Milan --> Moscú --> Milan

El número de aviones usado es: 16

Coste total de la solución: 343.53441935146947

Se ha tardado 5.0 milisegundos en encontrar la solución

---------- HEURISTIC SOLUTION ----------

----------RUTA 1 ----------

Coste de la ruta: 54.15763520317266

Barcelona --> Milan --> Frankfurt --> Paris --> Londres --> Barcelona

----------RUTA 2 ----------

Coste de la ruta: 22.430245017398484

Barcelona --> Lisboa --> Madrid --> Barcelona

----------RUTA 3 ----------

Coste de la ruta: 71.543547186328

Lisboa --> Barcelona --> Dublin --> Milan --> Vigo --> Lisboa

----------RUTA 4 ----------

Coste de la ruta: 31.96328741518763

Milan --> Barcelona --> Zaragoza --> Lisboa --> Munich --> Milan

----------RUTA 5 ----------

Coste de la ruta: 64.75084135772802

Milan --> Roma --> Moscú --> Milan

El número de aviones usado es: 5

Coste total de la solución: 244.84555617981482

Se ha tardado 2.0 milisegundos en encontrar la solución

En este caso, la reducción del coste es del 28,7% y del número de aviones del 68,8%.

De manera gráfica, la solución completa es:

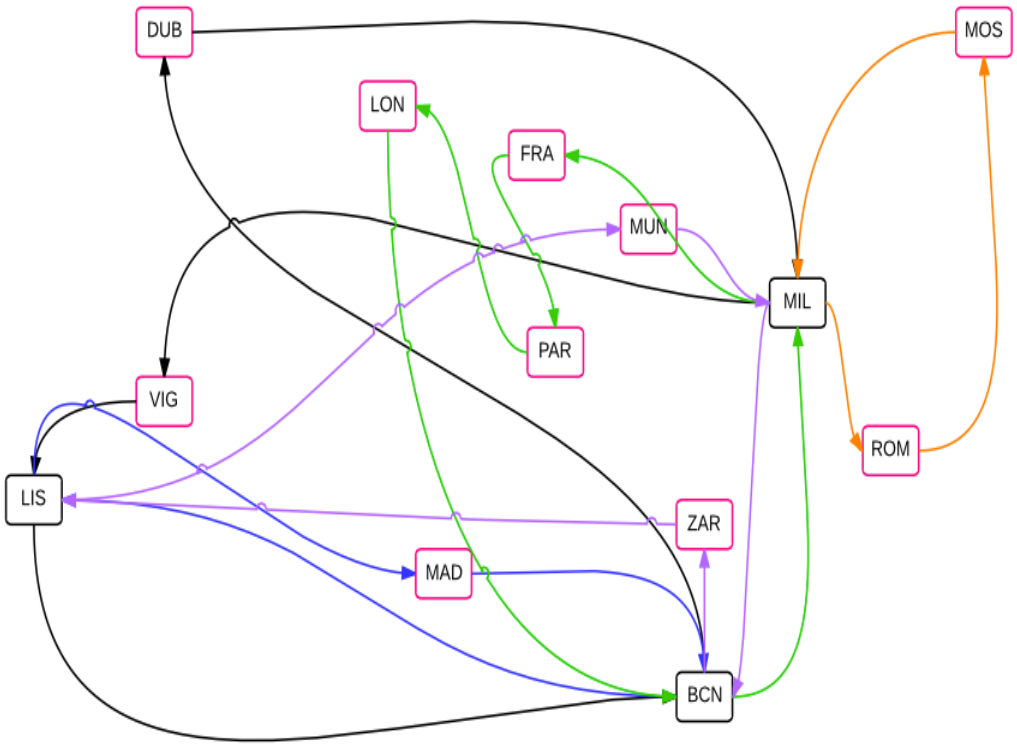


Figura 18: Solución completa 2*6 y 1*4

De manera más detallada se pueden apreciar las siguientes rutas especificas:

Ruta 1:

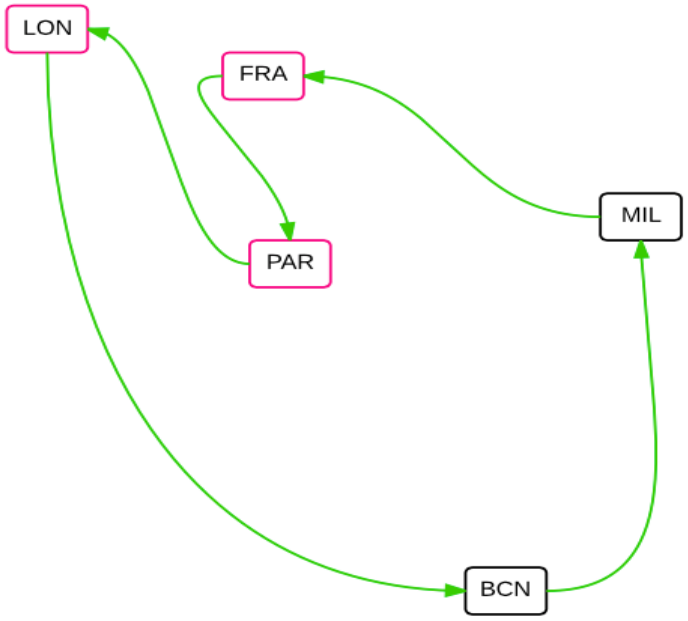


Figura 19: Ruta 1 detallada de la solución

Ruta 2:

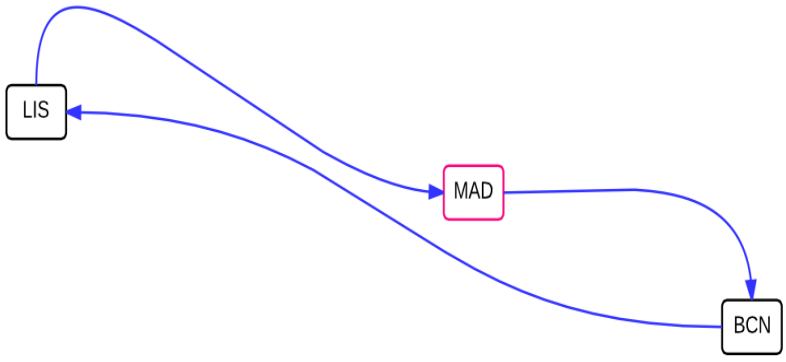


Figura 20: Ruta 2 detallada de la solución

Ruta 3:

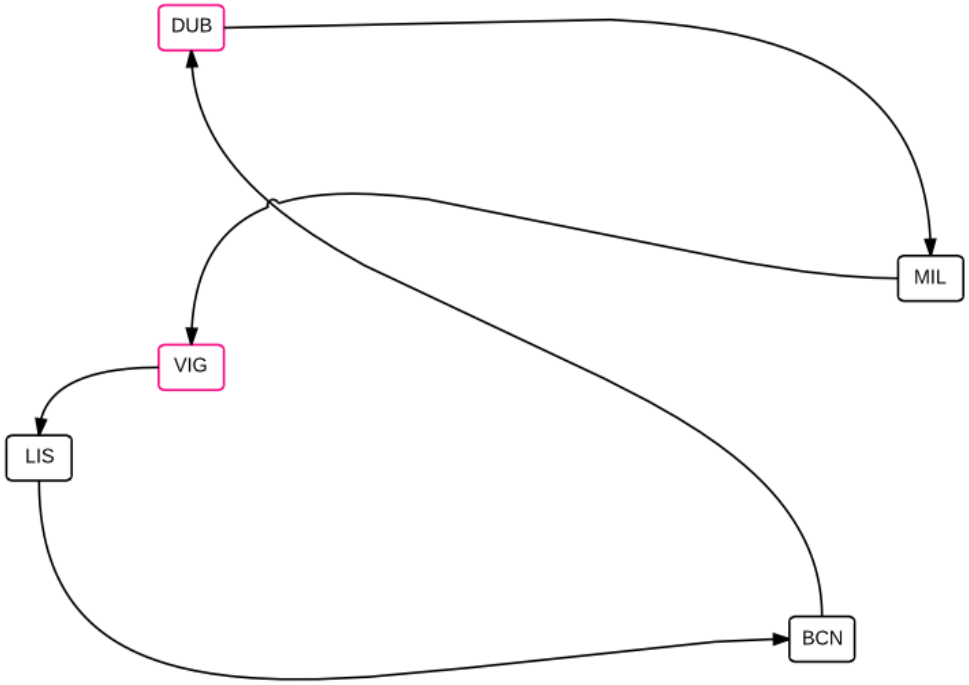


Figura 21: Ruta 3 detallada de la solución

Ruta 4

Figura 22: Ruta 4 detallada de la solución

Ruta 5:

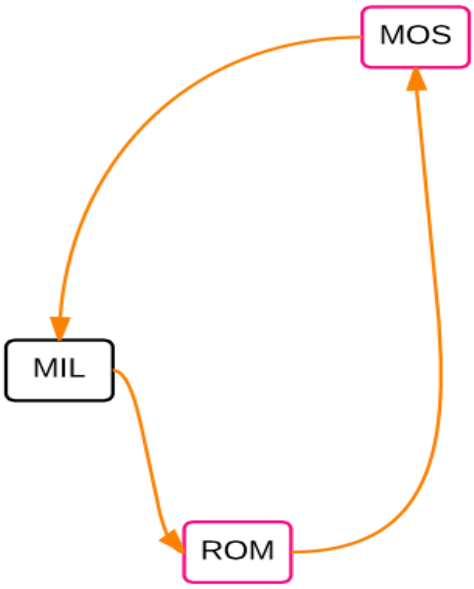


Figura 23: Ruta 5 detallada de la solución

4.1.2.4. Soluciones *biased randomization*

Para demostrar la aleatoriedad del algoritmo se realizarán varias ejecuciones con los mismos inputs y se verificará la diferencia de resultados entre estas ejecuciones.
Con los inputs de 3 depots, en los que 1 de esos depots sirve a 8 clientes y los otros depots sólo a 2, se obtienen los siguientes resultados haciendo varias ejecuciones:

Solución Dummy 1	Solución 2 Dummy
----------RUTA 1 ----------	----------RUTA 1 ----------
Coste ruta: 10.878924834476944	Coste ruta: 10.878924834476944
Barcelona --> Lisboa --> Barcelona	Barcelona --> Lisboa --> Barcelona
----------RUTA 2 ----------	----------RUTA 2 ----------
Coste ruta: 18.799002535716905	Coste ruta: 18.799002535716905
Barcelona --> Milan --> Barcelona	Barcelona --> Milan --> Barcelona
----------RUTA 3 ----------	----------RUTA 3 ----------
Coste ruta: 30.563823023830498	Coste ruta: 30.563823023830498
Barcelona --> Paris --> Barcelona	Barcelona --> Paris --> Barcelona
----------RUTA 4 ----------	----------RUTA 4 ----------
Coste ruta: 20.888807652813743	Coste ruta: 20.888807652813743
Barcelona --> Londres --> Barcelona	Barcelona --> Londres --> Barcelona
----------RUTA 5 ----------	----------RUTA 5 ----------
Coste ruta: 11.67549242700649	Coste ruta: 11.67549242700649
Barcelona --> Madrid --> Barcelona	Barcelona --> Madrid --> Barcelona
----------RUTA 6 ----------	----------RUTA 6 ----------
Coste ruta: 29.299776939610606	Coste ruta: 29.299776939610606
Barcelona --> Dublín --> Barcelona	Barcelona --> Dublín --> Barcelona
----------RUTA 7 ----------	----------RUTA 7 ----------
Coste ruta: 21.65828476191328	Coste ruta: 21.65828476191328
Barcelona --> Vigo --> Barcelona	Barcelona --> Vigo --> Barcelona
----------RUTA 8 ----------	----------RUTA 8 ----------
Coste ruta: 22.04025539021759	Coste ruta: 22.04025539021759

Barcelona --> Frankfurt --> Barcelona ----------RUTA 9 ---------- ---------- Coste ruta: 10.878924834476944 Lisboa --> Barcelona --> Lisboa ----------RUTA 10 ---------- ---------- Coste ruta: 14.280236540045422 Lisboa --> Milan --> Lisboa ----------RUTA 11 ---------- ---------- Coste ruta: 18.799002535716905 Milan --> Barcelona --> Milan ----------RUTA 12 ---------- ---------- Coste ruta: 14.280236540045422 Milan --> Lisboa --> Milan ---------- ---------- El numero de aviones usado es:12 Coste total de la solución: 224.04276801587076	Barcelona --> Frankfurt --> Barcelona ----------RUTA 9 ---------- ---------- Coste ruta: 10.878924834476944 Lisboa --> Barcelona --> Lisboa ----------RUTA 10 ---------- ---------- Coste ruta: 14.280236540045422 Lisboa --> Milan --> Lisboa ----------RUTA 11 ---------- ---------- Coste ruta: 18.799002535716905 Milan --> Barcelona --> Milan ----------RUTA 12 ---------- ---------- Coste ruta: 14.280236540045422 Milan --> Lisboa --> Milan ---------- ---------- El numero de aviones usado es:12 Coste total de la solución: 224.04276801587076

Solución 1 Heurística	Solución 2 Heurística
----------RUTA 1 ---------- ---------- Coste de la ruta: 52.75720073274139 Barcelona --> Lisboa --> Paris --> Londres --> Madrid --> Barcelona ----------RUTA 2 ---------- ---------- Coste de la ruta: 49.62416303891774 Barcelona --> Milan --> Frankfurt --> Dublín --> Vigo --> Barcelona ----------RUTA 3 ---------- ---------- Coste de la ruta: 21.979081955119636 Lisboa --> Barcelona --> Milan --> Lisboa ----------RUTA 4 ---------- ---------- Coste de la ruta: 21.979081955119636 Milan --> Barcelona --> Lisboa --> Milan ---------- ---------- El numero de aviones usado es:4 Coste total de la solución: 146.3395276818984	----------RUTA 1 ---------- ---------- Coste de la ruta: 77.85777979743841 Barcelona --> Paris --> Londres --> Vigo --> Frankfurt --> Barcelona ----------RUTA 2 ---------- ---------- Coste de la ruta: 55.0903734226081 Barcelona --> Lisboa --> Milan --> Madrid --> Dublín --> Barcelona ----------RUTA 3 ---------- ---------- Coste de la ruta: 21.979081955119636 Lisboa --> Barcelona --> Milan --> Lisboa ----------RUTA 4 ---------- ---------- Coste de la ruta: 21.979081955119636 Milan --> Barcelona --> Lisboa --> Milan ---------- ---------- El numero de aviones usado es:4 Coste total de la solución: 176.90631713028577

Se aprecia que las soluciones dummy de ambas ejecuciones son idénticas, pues como es lógico todos los *depots* sirven a sus clientes y en el mismo orden, pues sólo hay una solución *dummy* factible.

Por el contrario, la solución heurística consigue minimizar el número de aeronaves en 4 en ambas ejecuciones, pero las rutas son distintas en cada una de las soluciones, por lo que el coste total de la solución es diferente.

En este caso además se puede afirmar que la solución heurística número 1 es mejor (más barata) que la solución 2, aunque aún existen combinaciones factibles que no han sido extraídas con sólo 2 ejecuciones y que, por lo tanto, podrían mejorar la solución 1.

Con todo esto se verifica que se ha implementado correctamente la ecuación que determina la aleatoriedad del algoritmo y, consecuentemente, que con distintas ejecuciones se pueden conseguir distintas soluciones. Es importante entender que cuanto más grande sea el tamaño de los inputs, mayor número de combinaciones existirá, haciendo que existan una gran variedad de soluciones distintas, aunque todas mejorarán el coste y la usabilidad de las aeronaves respecto la solución *dummy.*

4.1.2.5. Análisis de resultados

Los resultados obtenidos son teóricamente aceptables, sobretodo si lo comparamos con la solución a mano, que demuestra que no hay errores de cálculo y que, además, el algoritmo es más eficiente en tiempo.

Numéricamente, la solución además es muy interesante, pues reduce los costes operacionales de una aerolínea alrededor de un 30% y el número de aviones usados entre un 50% y un 70%, basándonos en que el planteamiento para cubrir todos los clientes es usar la solución *dummy*.

Ahora bien, una cuestión importante a analizar es si en el mundo aeronáutico actual, los resultados son correctos y son realistas. Para ello, se ha contactado con una persona que trabaja en el Departamento de Estrategia y Planificación de Rutas de Volotea.

Después de una sesión informativa con él, en la que se le ha introducido al problema que se quería tratar, a la metodología usada y, finalmente, a las soluciones obtenidas, se ha conseguido identificar una serie de conceptos que faltan por trabajar para conseguir implementar este algoritmo en una aerolínea comercial.

4.1.3 Conceptos a trabajar:

1. No siempre se quieren unir dos aeropuertos a pesar de que estos se sirvan desde un mismo cliente. Un ejemplo es: Desde el *depot* Barcelona queremos ir a Lisboa y a Milán, pero no queremos ir desde Lisboa a Milan o viceversa.
Por ello, se ha visto que para solucionar esto se debería añadir un fichero de excepciones al algoritmo. Este fichero incluiría las conexiones entre aeropuertos prohibidas.

Además, también es importante el orden en el que conectamos clientes, ya que cada vez más el sector aeronáutico y el *core business* de las aerolíneas reside en los vuelos feeders que a veces hacen las propias aerolíneas. Si conectas en el orden incorrecto los vuelos, no aprovechas los fedeers, eliminando así la mayor parte del beneficio de una aerolínea, ya que el vuelo principal, que es el que te da más beneficios, se queda sin el número de pasajeros necesarios. Así pues, también sería posible añadir un fichero que incorpore las rutas feeders para obligar al algoritmo a que antes de cierto leg, deben haberse ejecutado como mínimo algunas de esas rutas feeders para garantizar la viabilidad económica operacional.

2. Se deben incorporar los cruces que hacen las aerolíneas comerciales por cuestiones de mantenimiento: Se debería incorporar una condición que oblige a finalizar cada ruta en uno de los dos aeropuertos que pueden realizar el mantenimiento de la aeronave. Así pues se necesitaría un fichero de *depots* que pueden realizar el mantenimiento.
Se observa que el modelo actual choca con esta idea, pues el aeropuerto de inicio también es el de finalización de dicha ruta y no se evalúa si puede realizar mantenimiento.

Con este concepto se conseguiría añadir las condiciones de mantenimiento al algoritmo, aunque requeriría de un estudio previo para saber qué aeropuertos son los que sirven como aeropuerto de mantenimiento.

Otra idea muy distinta sería la de que el algoritmo identificase cuando hacer el mantenimiento. Para esto sería necesario incorporar las leyes aeronáuticas actuales al algoritmo y contabilizar las horas que usa cada aeronave en cada trayecto para saber cuando se acerca a los límites legales. Una vez se acerce a estos que cierre la ruta en un aeropuerto que sea *depot* y a la vez que pueda hacer mantenimiento.

3. Finalmente, se planteó una tercera y última cuestión. Qué sucedería en caso de que un avión fallase en medio de una ruta programada?

Esto es un factor muy importante y que se produce frecuentemente. Para ello, en el algoritmo se debería incorporar una fórmula estadística que modelase el comportamiento de los fallos de motor y otros componentes de las aeronaves, para definir cada cuantas horas de vuelo suele haber un fallo.

Esto permitiría considerar este factor en el algoritmo y que, cuando se produjese un fallo, el algoritmo contabilizase que se utiliza una aeronave extra para acabar de cubrir esa ruta, además de los costes de mandarla hasta el aeropuerto donde se ha detectado el fallo del anterior.

A pesar de estas posibles mejoras, se observa que el algoritmo presenta una primera aproximación interesante para definir el orden en que se deben servir los legs para cubrir la totalidad de los aeropuertos que debe servir la aerolínea desde cada *hub* de la misma, minimizando coste total de operación y uso de aeronaves. Aunque carece de muchos factores realistas e importantes que ayudarían a definir el comportamiento real de las aeronaves en dichas rutas calculadas, y que ayudarían a trabajar al algoritmo en un sistema completamente aleatorio que ayudaría a la aerolínea a la planificación de la operativa.

Sin embargo, el empleado de Volotea coincide en que, a pesar de las carencias del algoritmo para el sector comercial y que en la planificación actual de una aerolínea si que son considerados, el algoritmo es una buena versión alfa y que, además, es una muy buena aproximación a problemas realistas en el sector del cargo aéreo.

La diferencia reside en que el sector del cargo no plantea limitaciones como las comentadas en el punto 1 de los conceptos a trabajar. Ya que desde el aeropuerto de destino, requieren servir un conjunto de clientes, y da igual cómo los conectes mientras los sirvas. Del mismo modo, no es tan importante el orden en que se conecten los clientes siempre y cuando cumplas con los horarios de entrega.

Este último punto nos ayuda a identificar dónde reside el principal potencial de este algoritmo. El potencial de este reside en que puede ser útil (en el estado actual) puramente para el sector cargo como algoritmo de planificación.

Esto quiere decir que una compañía de cargo aéreo, como puede ser DHL, MRW, Correos o cualquier otro, puede usar este algoritmo para definir en qué orden servir a los distintos clientes y como conectar entre clientes, para minimizar el coste de dicha ruta y minimizar las aeronaves usadas.

Esto se traduce en que el algoritmo ayuda a la competitividad de las compañías de cargo aéreas, minimizando sus costes operativos y consiguiendo que tengan el mínimo numero de aeronaves en propiedad o renting. Ahora bien, que este algoritmo sirva como planificador quiere decir que los horarios de entrega o de llegada a cada aeropuerto cliente se definirán según el orden y las conexiones que determine el algoritmo. Por lo que podemos definir que este algoritmo puede ser considerado como un algoritmo de programación de vuelos en el sector del cargo aéreo, definiendo así la pauta operativa de este tipo de compañías.

SECCIÓN 5: CONCLUSIONES Y TRABAJO FUTURO

5.1 Conclusiones

A lo largo del proyecto se ha realizado un algoritmo que permita a las aerolíneas calcular la ruta más óptima en cuanto a coste y usabilidad de aviones para servir un número de aeropuertos clientes desde un aeropuerto hub. Se puede decir pues, que es un algoritmo que trata la fase de *route scheduling* y *air fleet assignment*, pues plantea con qué rutas minimizas el número de aeronaves necesarias a asignar para facilitar la tarea de AFA(*air fleet assignment)*.

Además, este algoritmo se ha llevado un paso más allá, se ha pretendido que sea un algoritmo que solucione este problema en su versión multidepot. Esto quiere decir que no sólo soluciona la demanda de los clientes desde un *hub,* sino que lo soluciona para diferentes *hubs* y los diferentes clientes que cada uno de ellos debe servir.

Para conseguirlo, primero de todo se ha realizado un estudio detallado de la bibliografía, para conseguir el conocimiento necesario de los diferentes planteamientos posibles para el problema.

Una vez con esto, se ha realizado una introducción al problema y al proyecto, para luego explicar un marco conceptual que permite entender la metodología actual y la situación actual de las aerolíneas en situaciones similares a las que trata este proyecto.

Posteriormente, se ha definido la metodología usada, además de las técnicas en las que se basa dicha metodología, sin olvidar la explicación de las partes más importantes del algoritmo con pseudocódigo. Finalmente, se han verificado los resultados obtenidos con esta y se han analizado.

Con el análisis de estos resultados se han obtenido las siguientes conclusiones:

- Es clara la necesidad de estudio del problema por parte de las aerolíneas. El modelo aquí planteado ayuda a optimizar desde un inicio la planificación de las aerolíneas, por lo que una mejora constante de este algoritmo y una ampliación del alcance de este, puede ayudar a mejorar y optimizar todo el proceso de planificación, además de ayudar a la automatización de toda esta fase de la planificación.

La optimización de costes que la resolución de este problema puede conseguir es muy alta, hablamos de alrededor del 30% respecto la solución *dummy*. Sin olvidar la usabilidad de las aeronaves, que se maximiza en gran medida y, consecuentemente, se minimiza la necesidad de aeronaves entre un 50% y un 70%. Todo esto implica una disminución de los costes fijos y variables de la aerolínea y, en consecuencia, una mayor facilidad de conseguir beneficios por parte de esta. Por lo que los objetivos del proyecto han estado satisfactoriamente cumplidos.

- Por otro lado, para incrementar la usabilidad real del algoritmo se debe mejorar el mismo con factores que ahora mismo se olvidan. Ejemplo de ello: las demandas de cada *leg*, las capacidades de cada aeronave, la limitación del número de aeronaves que posee la aerolínea, condiciones de mantenimiento, conexiones prohibidas por cuestiones de la aerolínea y otros vistos anteriormente, aunque por cuestiones de tiempo, el alcance del problema es más que satisfactorio.

- Sin embargo, el primer paso se ha conseguido exitosamente, pues se han podido implementar con éxito en el algoritmo las técnicas de aleatoriedad. Buena prueba de ellos es que si se trabaja con los mismos *inputs* una y otra vez, se obtiene una solución diferente.

- Con todo esto se puede afirmar que los resultados obtenidos con la aplicación del algoritmo corresponden a los resultados correctos y coherentes que se deberían obtener, aunque aún no están listos para incoporarse a la operativa diaria de una aerolínea por falta de factores que deben estudiarse. Así pues, este algoritmo produce el primer granito de arena en una montaña que aún se debe construir.

Lo que de momento aporta este es una primera fase, que permite ver la eficiencia de dicho algoritmo en frente de un humano en la resolución de un problema de estas características. Así pues, se da a entender que en el caso de aproximar el problema a una situación real, el algoritmo aún será mucho más eficiente que no la solución que pueda obtener un humano. Por lo tanto, se abre una pregunta, ¿Se puede semiautomatizar la planificación de las rutas a través de un algoritmo?

-En respuesta a la pregunta se puede decir que el algoritmo puede plantearse como un planificador de rutas. Es decir, este te da una secuencia de *legs* que minimizan costes y número de aeronaves usadas para cubrir todos los clientes. Por lo tanto, esta secuencia que nos da puede ser usada para planificar los horarios de la actividad aeronáutica de la aerolínea, sobretodo para el sector de cargo aéreo, pues el orden de los *legs* en la solución nos podría definir los horarios y el orden en que se deben servir los clientes para minimizar costes y aeronaves.

5.2 Futuras líneas de trabajo

En este proyecto se ha presentado un nuevo modelo para solucionar el problema de la planificación de rutas y de la asignación de aeronaves. A través de dicho modelo se ha conseguido dotar de un giro a los planteamientos realizados hasta ahora por otros investigadores, y así aportar una solución innovadora.

Lógicamente, se ha presentado una primera versión del algoritmo, el cual deja una gran cantidad de vertientes que analizar y mejorar para obtener, consecuentemente, unos mejores resultados.

Evaluando pues el trabajo realizado en este proyecto y las conclusiones que se han podido extraer del mismo, se enumeran acto seguido las principales líneas de trabajo futuro que pueden dar continuidad al presente proyecto:

1. Aplicar técnicas *simheuristics* para conseguir una aplicación más realista del problema dentro de un entorno estocástico (cómo el real), una de las primeras líneas de trabajo sería estudiar las distribuciones estadísticas que modelan el comportamiento de los inputs, que aquí no se han incorporado (demanda, oferta y otros), y añadirlas al modelo para hacer que el algoritmo trabaje en un entorno realista.

Con esto se conseguiría una representación más real del sistema, ya que por ejemplo entre muchos otros, la demanda es variable en función de lo cerca que estamos del día de operación. Otro ejemplo es que la duración de un vuelo, generalmente, no es la prevista, sino que sufre desviaciones respecto el valor esperado. Todo esto no siempre se ve alterado por variables conocidas y controlables, sino que es alterado por factores no controlables y/o desconocidos. Lo mismo sucede con otras variables que siguen ciertas distribuciones estadísticas. Por lo tanto, los resultados de la computación serán más fiables para la aplicación real gracias a la similitud de comportamiento que guardara el modelo con el sistema real.

2. Aprovechando el potencial de las técnicas de simheuristics se debería intentar modelar el comportamiento de los *delays*. Mayoritariamente, una causa que genera muchos retrasos y que es difícil de predecir es la meteorología.
Es decir, el factor estocástico que supone un cambio climático inesperado en el día de operaciones y que puede alterar la programación de manera muy notable, afectando a los horarios de los vuelos o incluso a que un ciclo de una ruta termine abruptamente. Esto influiría pues en que una ruta quizás se debería separar en dos, o que incluso una ruta puntualmente no se pueda realizar y se deba quitar un aeropuerto (debido a la meteorología se pueden llegar a cerrar aeropuertos durante horas).

Por eso es importante que el proceso de modelado refleje con precisión como modelar los *delays*. No importa el porque hay un *delay*, es decir, no necesitamos modelar un cambio meteorológico o un imprevisto técnico, sino cómo modelar la probabilidad de que se produzca un retraso y como afectaría esto a la planificación.

Así se conseguiría ejecutar una simulación que mostrase unos resultados que han evaluado factores que pueden afectar la planificación de vuelos a través de retrasos, independientemente del motivo. Además de indicar de esta manera cuando un ciclo puede sufrir un corte y se debe pensar una posible solución.

3. Incorporar la opción de bloquear ciertas conexiones entre aeropuertos, los cuales, a pesar de tener que servirse desde un mismo *depot,* no deben ser conectados entre ellos por cuestiones operacionales y comerciales de la aerolínea.

Así pues, también sería posible añadir la opción de marcar las rutas *feeders* para obligar al algoritmo a que antes de cierto leg, deben haberse ejecutado como mínimo algunas de esas rutas *feeders* para garantizar la viabilidad económica operacional.

Esto es importante, pues muchas aerolíneas basan su actividad comercial en una ruta que les reporta muchos beneficios y que les compensa las pérdidas de las rutas *feeders* que alimentan a esa ruta tan exitosa. Si no se hace en el orden correcto, todas las rutas presentarían pérdidas económicas, por lo que, a pesar de ahorrarse costes y aeronaves, a la aerolínea no le interesaría esta solución.

4. Incorporar el concepto del fallo de los aviones. Para ello se debería modelar con una distribución estadística apropiada los tiempos de fallo de los componentes de las aeronaves, para saber cuando estas fallan. Esto permitiría simular en el algoritmo cuando puede fallar una aeronave en una ruta específica, para así prever cuando puede ser necesario tener preparado un avión y desde dónde debería ser enviado este avión de sustitución a la ruta que ha sufrido el fallo. Esto ayudaría a sumar ese coste al coste de la solución y valorar si realmente sigue saliendo más rentable.

Tanto la línea de trabajo futuro número 3 y la número 4 ayudarían a que el algoritmo tuviese más usabilidad para su aplicación real en el sector aeronáutico comercial. Aunque también mejoraría aún más su aplicación en el sector de carga aérea.

5. Añadir al algoritmo las condiciones de mantenimiento a las que están sujetas las aerolíneas. Bien añadiendo la contabilización de las horas de vuelo de cada aeronave y cuando le toca mantenimiento, para así planificar cuando parar este avión en un aeropuerto dónde puede hacerse el mantenimiento, o bien haciendo que las rutas siempre acaben en aeropuertos dónde poder realizar el mantenimiento.

Quizás el problema deba replantearse para poder incorporar esta línea de trabajo. Actualmente, las rutas siempre empiezan y terminan en un mismo aeropuerto, por lo que para asegurar esta condición, quizás debería incorporarse que cada X rutas el aeropuerto de destino sea diferente al de salida y así poder hacer el mantenimiento y, posteriormente, retomar el ciclo normal.

SECCIÓN 6: BIBLIOGRAFÍA

6.1 Bibliografía

Cheng,F., J Gulding. 2014 "A hybrid optimization-simulation approach for itinerary generation" - World simulation conference.

Barnhart 2012 - "Quantitative Problem Solving Methods in the Airline Industry". Springer, Ltd., 2012. Print.

Bazargan 2010 - "Airline Operations and Scheduling". Ashgate Publishing, Ltd., 2010. Print.

Abara, J. 1989. “Applying Integer Linear Programming to the Fleet Assignment Problem,” Interfaces 19: 20-38.

Barnhart, C., A. Cohn. 2004. “Airline Schedule Planning: Accomplishments and Opportunities.” Manufacturing & Service Operations Management 6(1): 3-22.

Cheng, F., J. Gulding, B. Baszczewski, and R. Galaviz. 2011. “An Optimization Model for Sample Day Selection in NAS-wide Modeling Studies".

Jacobs, T., B. Smith, and E. Johnson. 2008. “Incorporating Network Flow Effects into the Airline Fleet Assignment Process,” Transportation Science 42(4): 514-529.

Rexing, B., C. Barnhart, T. Kniker, A. Jarrah, and N. Krishnamurthy. 2000. “Airline Fleet Assignment with Time Windows”, Transportation.

Ozdemir, Y., Basligil, H., Baglan, S. 2012. "A large scale integer linear programming to the daily fleet assignment problem: a case in turkey".

Venkata, L.P., Rosenberg, J.M., Chen, Victoria., Smith, B.C. 2005. "A statistical computer experiments approach to airline fleet assignment”, University of Texas.

Lan, S. 2003. "Planning for robust airline operations: Optimizing Aircraft Routings and Flight departure times to achieve minimum passenger disruptions”, Massachusetts Institute of Technology.

Catherine, M., Mora-Camino, F. 2014. "Airline fleet assignment: a state of the art".

Angel A. Juan , Javier Faulin , Scott E. Grasman , Markus Rabe , Gonçalo Figueira. "A review of simheuristics: extending metaheuristics to deal with stochastic combinatorial optimization problems ".

Hanif D. Sherali *, Ebru K. Bish, Xiaomei Zhu. "Airline fleet assignment concepts, models, and algorithms"

Cacchiani, Valentina. "A heuristic approach for an integrated fleet-assignment, aircraft-routing and crew-pairing problem"

Chang. 2001. "An integrated approach to flight schedule and fleet assignment"

Belanger, N. 2006. "Weekly airline fleet assignment with homogeneity".

Fisher, M. 1979. "A generalized assignment heuristic for vehicle routing".

Lohatepanont, M. 2002. "Airline fleet assignment and schedule design. Models and algorithms".

Wang. 2010. "Application of simulated annealing algorithm in airline fleet assignment problem"

Talluri. 2001. "Swapping Applications in a Daily Airline Fleet Assignment"

Shenari. 2005. "Airline fleet assignment concepts, models, and algorithms"

Mancel. 2006. "Airline fleet assignment- a state of the art"

Hanif, D. Sherali. 2006. "Two-Stage Fleet Assignment Model Considering Stochastic Passenger Demands"

Jay M. Rosenberg. 2004. "A robustfleet-assignment model with hub isolation and short cycles"

Firma del autor:

Miquel Rius Carmona