PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

ALGORITMO GENÉTICO PARA LA ASIGNACIÓN DE TIPO DE AVIONES A VUELOS

Tesis para optar el Título de **Ingeniero Informático,** que presenta el bachiller:

Víctor Gabriel Avalos Aguilar

ASESOR: Rony Cueva Moscoso

Lima, febrero de 2015

RESUMEN

El continuo crecimiento del comercio mundial ha ocasionado un incremento constante en la demanda de vuelos comerciales. Las aerolíneas se han visto en la necesidad de diversificar sus flotas de aeronaves y aumentar el número de las mismas para satisfacer la creciente demanda. La variedad de tipos de avión, la creciente cantidad de vuelos y un mayor número de aeronaves disponibles han complicado el proceso mediante el cual se asigna un avión específico a atender un vuelo programado. Ante esta nueva realidad se ha visto un creciente número de investigaciones dedicadas a diseñar algoritmos capaces de obtener una buena asignación vuelo-avión utilizando la menor cantidad de recursos.

Los algoritmos planteados han ido subiendo en complejidad a medida que ha pasado el tiempo. Los primeros que fueron planteados eran denominados algoritmos exactos, estos podían obtener la respuesta optima, pero requerían de mucho tiempo y poder de procesamiento. Luego se hizo uso de algoritmos heurísticos, como el GRASP, el cual entregaban una solución buena, que posiblemente no sea la óptima, pero su consumo de recursos era menor. En la actualidad se han diseñado varios algoritmos meta-heurísticos que permiten obtener una mejor solución que los heurísticos haciendo mejoras continuas a la solución obtenida hasta que se cumplan ciertas condiciones de parada.

El objetivo de este proyecto es diseñar un algoritmo genético que minimize los costos en la asignación avión-vuelo y a la vez maximice los posibles beneficios a obtener. Para cumplir con este objetivo se hará un estudio de los conceptos asociados a la asignación de tipos de aeronaves a vuelos y se recopilarán datos reales de previas asignaciones hechas por aerolíneas que están presentes en el mercado peruano. El producto final será un algoritmo genético diseñado y calibrado para obtener soluciones que sean válidas para el actual contexto nacional.

FACULTAD DE
CIENCIAS E INGENIERÍA
ESPECIALIDAD DE
INGENIERÍA INFORMÁTICA

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO INFORMÁTICO

TÍTULO: **ALGORITMO GENÉTICO PARA LA ASIGNACIÓN DE TIPO DE AVIONES A VUELOS**

ÁREA: Ciencias de la computación
PROPONENTE: Ing. Rony Cueva Moscoso
ASESOR: Ing. Rony Cueva Moscoso
ALUMNO: Víctor Gabriel Avalos Aguilar
CÓDIGO: 20084520
TEMA N°: 570
FECHA: 15/12/2014

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
Facultad de Ciencias e Ingeniería
08 ABR. 2015
MIGUEL MEJIA PUENTE
DECANO

DESCRIPCIÓN

Debido al crecimiento continuo de la demanda por vuelos domésticos a nivel nacional las aerolíneas se han visto obligadas a aumentar la frecuencia de los vuelos y por consiguientes han tenido que aumentar su flota de aviones, a veces utilizando nuevos modelos de aeronaves.

Este incremento en la variedad de modelos de aeronaves que maneja una aerolínea y en la cantidad de vuelos ofrecidos complica la realización del proceso denominado "asignación de flotas en aerolíneas". Este proceso se encarga de asignar a cada vuelo un tipo de aeronave para que lo atienda minimizando el costo total del vuelo y maximizando la posible ganancia. Para la minimización se toma en cuenta principalmente el uso de combustible, que representa la mayor parte del costo de operación. Para la maximización se toma en cuenta la demanda de cada ruta y lo cantidad de pasajeros por cada tipo de avión.
Debido a la complejidad del proceso se ha hecho uso de algoritmos de asignación a lo largo del tiempo. La creciente complejidad del problema ha requerido el uso de algoritmos meta-heurísticos los cuales otorgan una respuesta buena, probablemente no la más óptima, en un corto tiempo.

El algoritmo genético a desarrollar permitirá obtener una buena asignación de flotas en un tiempo relativamente corto, maximizando el beneficio obtenido por cada vuelo mediante la reducción de los costos asociados a su atención y priorizando las rutas con mayor demanda.

OBJETIVO GENERAL

El objetivo del Proyecto es desarrollar un algoritmo genético que optimice la asignación de tipo de aviones con los vuelos programados de una aerolínea maximizando el beneficio obtenido.

Av. Universitaria 1801 Apartado Postal 1761 Teléfono:

OBJETIVOS ESPECÍFICOS

Los objetivos específicos del presente proyecto son:

- Diseñar las estructuras de datos requeridas para cada uno de los algoritmos a desarrollar.
- Definir la función objetivo que tome en cuenta las restricciones propias del proceso de asignación de flotas en aerolíneas.
- Implementar el algoritmo GRASP que será utilizado para la generación de la población inicial y en la comparación de resultados.
- Diseñar el pseudocódigo algoritmo genético utilizado en la solución del problema de asignación de flotas a vuelos tomando en cuenta las restricciones propias del problema.
- Seleccionar y realizar la experimentación numérica adecuada entre el algoritmo GRASP y el genético para comparar sus tiempos ejecución y la aptitud de sus soluciones.
- Desarrollar una aplicación que contenga los algoritmos implementados, que permite ejecutarlos y mostrar los resultados de cada uno.

ALCANCE

El algoritmo genético dará una solución para la asignación de flotas en aerolíneas tomando en cuenta las siguientes restricciones: tiempos muertos, disponibilidad de aeronaves, etapas de vuelo, conexión de flujo y la distancia máxima que puede recorrer cada tipo de flota.

Además la data de prueba estará basada en el contexto nacional por que la calibración del alfa para el algoritmo GRASP y otros parámetros como número de iteraciones para ambos algoritmos y el porcentaje de casamiento en el algoritmo genético reflejarán la realidad nacional. También, de acuerdo a la definición del proceso de asignación de flotas en aerolíneas el algoritmo debe recibir como datos de entrada la lista de vuelos de las aerolíneas y los tipos de flota.

Por último, el desempeño del algoritmo puede verse afectado por las características del hardware y software de la computadora en la que se ejecute, esto es debido a la complejidad del mismo algoritmo y al número de iteración que se deberán realizar.

Máximo: 100 páginas

DEDICATORIA

A mi padre Víctor por inculcar en mí la pasión por la tecnología, a mi madre Norka por apoyarme todo momento, a mi padrino Alejando y mi madrina Netly por ser grandes ejemplos a seguir y a toda mi familia, la cual me ha enseñado varias lecciones de la vida, gracias por estar siempre a mi lado y brindarme la fortaleza para seguir adelante.

AGRADECIENTOS

Quiero agradecer muy especialmente al Ing. Rony Cueva por guiarme, apoyarme y aconsejarme durante todas las etapas de este proyecto. A mis amigos César y Ángel, con quienes colaboramos en el transcurso del desarrollo de nuestros proyectos.

Índice general

Índice de imágenes

Índice de tablas

Índice de fórmulas

CAPÍTULO 1: GENERALIDADES

1 Problemática

El continuo crecimiento del comercio internacional que acompaña al proceso de globalización ocasiona un incremento en la demanda de vuelos internacionales, pero también se incrementa la demanda de vuelos domésticos, especialmente en países desarrollados. La falta de transporte aéreo, así como de cualquier otra entrada en el sistema económico, puede influir negativamente en el crecimiento económico, así como también lo pueden hacer la falta o inexistencia de servicios de transporte aéreo apropiados. [OECD, 2008]

La demanda de vuelos comerciales domésticos e internacionales ha ido creciendo constantemente, incluso en los últimos años, en los cuales las principales potencias económicas atravesaron una época de crisis [BOEING, 2013]. Esta situación obliga a las empresas a intentar ofrecer el mejor servicio al menor costo posible para hacerlo más rentable que sus competidores directos. Una de las posibles soluciones a este problema es maximizar el uso de todos sus activos. Siendo las aerolíneas importantes para dinamizar la economía, la maximización del uso de los activos es de interés también de los aeropuertos, los cuales poseen una gama de recursos disponibles pero en cantidad limitada, por ejemplo el tiempo que cada avión puede utilizar la infraestructura del aeropuerto es asignado con exactitud, cualquier retraso puede hacer que el avión pierda el derecho a usar ese recurso en el tiempo especificado. El trabajo conjunto de las aerolíneas con los aeropuertos facilita un uso eficiente de los recursos de ambas entidades y contribuye al crecimiento económico del país.

En este contexto donde las aerolíneas necesitan reducir sus costos se han desarrollado nuevos modelos de negocio, uno de estos es el denominado *aerolíneas de bajo costo.* Este tipo de aerolínea busca optimizar su estructura de costos para poder ofrecer servicios de transporte aéreo de carga a un precio mucho menor que las aerolíneas tradicionales [GROSS, SCHRÖDER, 2012]. Esto le obliga a tercerizar algunos de sus servicios y optimizar el uso de sus recursos en general, entre los más importantes tenemos:

- Aeronaves
- Tripulación
- Espacio de tiempo asignado en un aeropuerto (slot)

Cada uno de estos recursos cuenta con uno o varios procesos de planificación en el cual es asignado a un vuelo, en el caso de las aeronaves uno de estos procesos se llama asignación de flotas en aerolíneas *(airline fleet assignment)*. En este proceso se decide qué avión deberá operar cada uno de los vuelos pertenecientes a un plan de vuelos ya definido tomando en cuenta los costos operacionales de la asignación. [GONZÁLEZ, 2012]. Esta asignación debe tener en cuenta otros factores como limitantes [SHERALI, BISH, ZHU, 2006]:

- Tiempos muertos:
  - Mantenimiento en línea: Realizado generalmente después de cada vuelo.
  - Mantenimiento general: Realizado después de un tiempo especificado por el fabricante del avión.
- Número de aviones disponibles.
- Número de viajes que se deben hacer para ir de un aeropuerto a otro.
- Conexión de flujo.
- Distancia máxima que puede recorrer la aeronave.
- La posición de la nave al final y al comienzo del día.

Estos factores deben ser tomados en cuenta para evitar efectos adversos en la empresa. Uno de estos efectos es la asignación de una aeronave con menor capacidad de la óptima, lo que resulta en la pérdida de los clientes por falta de espacio. Otro efecto ocurre cuando se asigna una aeronave con demasiada capacidad para el vuelo, lo que genera la existencia de asientos no vendidos, aumentando los costos de operación. Por último, también se debe tomar en cuenta el consumo de combustible de la aeronave, el cual es él componente principal del costo de operación. Todos estos efectos tienen una influencia negativa en las finanzas de la empresa y peor aún en su imagen hacia los clientes, donde la última, en un ambiente donde se han homogenizado los precios, puede ser un factor decisivo en la elección de la aerolínea donde el cliente elija viajar.

El resultado del *airline fleet assignment* es generalmente un itinerario de vuelos para cada avión, donde se indica el aeropuerto de origen y de destino, acá también se especifican los tiempos de inactividad y de mantenimiento que tendrá el avión, en la imagen 1 se puede observar un ejemplo gráfico simple de un itinerario.

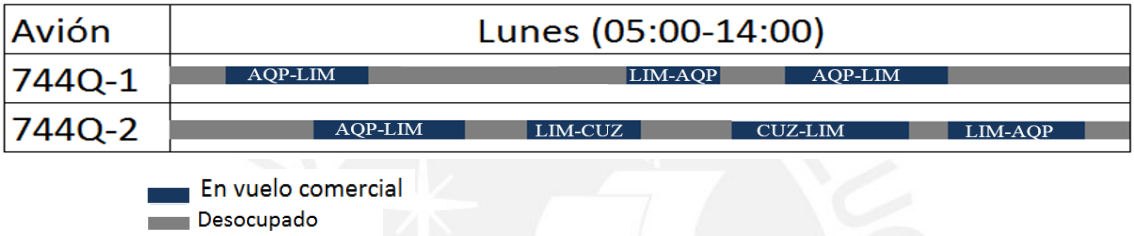


Imagen 1: Ejemplo de itinerario de aeronaves. Imagen de autoría propia.

La complejidad de este proceso depende principalmente de la cantidad de aviones y las posibles interconexiones entre vuelos. Por ello a medida que ha avanzado el tiempo se ha pasado de utilizar algoritmos de programación lineal, a algoritmos exactos y últimamente a algoritmos heurísticos y meta-heurísticos. La principal razón para el cambio de tipo de algoritmo a lo largo del tiempo se centra en el uso de recursos y del tiempo de ejecución de estos. Esto se debe al constante incremento de la demanda de servicios de transporte aéreo, que influye en el aumento en la cantidad de vuelos y esto conlleva a necesitar más aviones disponibles. Esta situación se puede observar tanto en Europa y en China, en donde, además, el modelo de aerolíneas de bajo costo se está empezando a crecer más rápido que el negocio de las aerolíneas tradicionales [ATM, 2013] [STUFF, 2014].

Ante esta realidad los algoritmos meta-heurísticos, se han presentado como la mejor opción al presentar un balance entre una buena solución, el tiempo requerido para obtenerla y la utilización de recursos. Aunque la situación en el Perú no sea tan compleja como en Europa o Asia, se puede hacer uso de algoritmos del mismo nivel para obtener soluciones buenas que se acerquen a la óptima en corto tiempo y además estar preparados para un posible crecimiento en la oferta y demanda de vuelos, situación muy probable debido al continuo crecimiento económico del Perú en los últimos 5 años.

Para este problema se diseñará un algoritmo meta-heurístico que genere una asignación de tipo de aviones a los vuelos programados de una aerolínea maximizando el beneficio obtenido. Para lograr este objetivo se minimizará el costo de operación, representado por el consumo de combustible, y maximizará la posible ganancia priorizando la asignación de aviones con mayor capacidad a vuelos de alta demanda, tomando en cuenta las características propias del avión y los factores limitantes mencionados anteriormente a excepción de los tiempos muertos por mantenimientos generales.

1.1 Objetivo general

Desarrollar un algoritmo genético que optimice la asignación de tipo de aviones a los vuelos programados de una aerolínea maximizando el beneficio obtenido.

1.2 Objetivos específicos

Con el fin de soportar el objetivo general se han planteado los siguientes objetivos específicos:

- Objetivo 1: Diseñar las estructuras de datos requeridas para cada uno de los algoritmos a desarrollar.
- Objetivo 2: Definir la función objetivo que tome en cuenta las restricciones propias del proceso de asignación de flotas en aerolíneas.
- Objetivo 3: Implementar el algoritmo GRASP que será utilizado para la generación de la población inicial y en la comparación de resultados.
- Objetivo 4: Diseñar el seudocódigo del algoritmo genético utilizado en la solución del problema de asignación de flotas a vuelos tomando en cuenta las restricciones propias del problema.
- Objetivo 5: Seleccionar y realizar la experimentación numérica adecuada entre el algoritmo GRASP y el genético para comparar sus tiempos de ejecución y la aptitud de sus soluciones.
- Objetivo 6: Desarrollar una aplicación que contenga los algoritmos implementados, que permite ejecutarlos y mostrar los resultados de cada uno.

1.3 Resultados esperados

- Resultado 1 para el objetivo 1: Documento que muestre el diseño de las estructuras utilizadas para representar los datos de los algoritmos y la solución del problema.
- Resultado 1 para el objetivo 2: Documento que muestre el diseño de la función objetivo y las restricciones.
- Resultado 1 para objetivo 3: Algoritmo GRASP-Construcción adaptado al problema y usado como algoritmo de experimentación para la generación de la población inicial y la comparación de resultados.
- Resultado 1 para objetivo 4: Documento que muestre el diseño del seudocódigo para el algoritmo genético adaptado al problema y usado para la solución del mismo y la comparación de resultados.
- Resultado 1 para el objetivo 5: Generar datos de entrada para la ejecución de los algoritmos genético y GRASP.
- Resultado 2 para objetivo 5: Documento detallado que muestre los tipos de prueba que fueron utilizados en el desarrollo de los resultados de la experimentación numérica, así como la interpretación de los resultados obtenidos de los algoritmos.

- Resultado 1 para objetivo 6: Aplicación que permita ejecutar los algoritmos de experimentación y de estudio y muestre los resultados obtenidos en ambos casos.

2 Herramientas, métodos, metodologías y procedimientos

2.1 Introducción

En esta sección se presentaran los resultados esperados con las herramientas, métodos y procedimientos que se emplearán en su desarrollo. Luego se procederá a explicar a detalle cada una de las metodologías a emplear en la gestión del proyecto y del desarrollo del producto.

2.2 Mapeo

Tabla 1: Mapeo de resultados esperados a herramientas y metodologías

Resultados esperado	Herramientas y metodologías a usarse
RE1 para OE1: Documento que muestre el diseño de las estructuras utilizadas para representar los datos de los algoritmos y la solución del problema.	**Microsoft Word 2013** facilitará la elaboración de documentos. **Extreme Programming** es una metodología de desarrollo que está más enfocada en la programación que en la documentación
RE1 para OE2: Documento que muestre el diseño de la función objetivo.	**Metodología del algoritmo genético** es una metodología que permitirá guiar la implementación de la función fitness. **Microsoft Word 2013** facilitará la elaboración de fórmulas, también como la descripción de cada variable presente en estas.
RE1 para OE3: Algoritmo GRASP-Construcción adaptado al problema y usado como algoritmo de experimentación para la generación de la población inicial y la comparación de resultados.	**Extreme Programming** es una metodología de desarrollo que está más enfocada en la programación que en la documentación. **Metodología del algoritmo GRASP** es una metodología que permitirá guiar la implementación del algoritmo. **Netbeans** es un entorno de desarrollo el cual posee varias herramientas que facilitan la codificación en varios lenguajes de programación. **Java** es un lenguaje de programación orientado a objetos que facilitará el uso de varios hilos de ejecución así como el manejo de memoria.
RE1 para OE4: Documento que muestre el diseño del seudocódigo para el algoritmo genético.	**Metodología del algoritmo genético** es una metodología que permitirá guiar la implementación de la función fitness. **Microsoft Word 2013** permitirá la elaboración del seudocódigo, facilitando herramientas de diseño.
RE1 para el OE5: Aplicación que permita generar de manera aleatoria datos de entrada para la ejecución de los algoritmos genético y GRASP.	**Extreme Programming** es una metodología de desarrollo que está más enfocada en la programación que en la documentación. **Netbeans** es un entorno de desarrollo pensado para elaborar programas, el cual posee varias herramientas que facilitan la codificación en varios lenguajes de programación.

Resultados esperado	Herramientas y metodologías a usarse
	Java es un lenguaje de programación orientado a objetos que facilitará el uso de varios hilos de ejecución.
RE2 para OE5: Documento detallado que muestre los métodos estadísticos que fueron utilizados en el desarrollo de los resultados de la experimentación numérica, así como la interpretación de los resultados obtenidos de los algoritmos.	**ANOVA (Análisis de la varianza)** es un test estadístico usado para la comparación de las medias entre dos tratamientos**.** **Microsoft Excel 2013** permitirá la elaboración de gráficos y tablas de resultados, además de constar con fórmulas que facilitarán la realización de las pruebas. **Microsoft Word 2013** permitirá la redacción de un análisis de los resultados obtenidos de la prueba, además permite importar tablas y gráficos de Microsoft Excel 2013.
RE1 para OE6: Aplicación que permita la ejecutar los algoritmos de experimentación y de estudio y muestre los resultados obtenidos en ambos casos.	**Extreme Programming** es una metodología de desarrollo que está más enfocada en la programación que en la documentación. **Netbeans** es un entorno de desarrollo pensado para elaborar programas, el cual posee varias herramientas que facilitan la codificación en varios lenguajes de programación. **Java** es un lenguaje de programación orientado a objetos que facilitará el uso de varios hilos de ejecución así como el manejo de memoria.

2.3 Herramientas

a) Netbeans

Netbeans es un IDE (Integrated Development Environment) desarrollado para escribir, compilar y ejecutar programas o aplicaciones basadas en java para ordenadores, dispositivos móviles y aplicaciones web. También posee herramientas para el desarrollo en los lenguajes de programación PHP y C/C++. Para esto, el IDE cuenta con una interfaz que permite editar código de una manera rápida, así como la sencilla y eficiente administración de proyectos de programación que se tienen [NETBEANS, 2013].

Debido a que esta herramienta permite administrar los proyectos de manera sencilla y eficiente se justifica su uso. Otra de las razones es que facilita la edición de código con funcionalidades como autocompletar, la cual ayudará en la implementación del algoritmo. Finalmente, el uso de esta herramienta está relacionada al lenguaje de programación Java a usar en el proyecto, ya que este IDE fue desarrollado y optimizado para el desarrollo de programas basados en JAVA.

b) Java

Java es un lenguaje de programación orientado de objetos. Este lenguaje es simple, fácil de aprender, distribuido, interpretado, sólido, seguro, portable y multiplataforma (funciona en varios sistemas operativos y en las principales plataformas de hardware) [JAVA, 2013].

Debido a que este lenguaje es multiplataforma y es adecuado para el IDE Netbeans, se justifica su elección. Otra razón para la elección de este lenguaje es el conocimiento que se posee del mismo.

c) Microsoft Excel 2013

Microsoft Excel 2013 es una aplicación perteneciente a Microsoft Office 2013. Es un programa que consiste principalmente en hojas de cálculo pero que en sus versiones más actuales es útil para el manejo de herramientas estadísticas. Estas funcionalidades permiten ahorrar tiempo, simplificar el trabajo y aumentar la productividad. Brinda herramientas que facilitan el desarrollo de análisis estadísticos así como la posibilidad de mostrar los resultados en gráficos dinámicos. Cuando se utilice una de estas herramientas, se deberá proporcionar los datos y parámetros para cada análisis y la herramienta utilizará las funciones de macros estadísticas o técnicas correspondientes para realizar los cálculos y mostrar los resultados en una tabla[MICROSOFT OFFICE, 2013].

En su versión 2013 es parte del paquete office 365, el cual permite guardar y editar documentos desde ONEDRIVE, un servicio de Microsoft que te permite guardar tus archivos en la nube.

Debido a las funcionalidades mencionadas anteriormente se usará el Excel tanto para la generación de data de prueba, usando como base data real, así como para la ejecución de las pruebas de la experimentación numérica. Se usarán las funciones de Excel para generar data semi-aleatoria variando la data real obtenida. Para la experimentación numérica se recopilarán los datos en tablas de Excel y mediante sus funciones estadísticas se aplicarán los métodos que se definirán más adelante. Por último, la facilidad para guardar el trabajo en la nube, lo cual permitirá tener una copia segura del trabajo realizado, justifican la elección de esta herramienta.

2.4 Metodologías

2.4.1 Relacionadas a la gestión del proyecto

a) PMBOK

El PMBOK es la suma del conocimiento dentro de la profesión de manejo de proyectos. EL PMBOK completo contiene prácticas tradicionales que son generalmente usadas, así como también conocimiento sobre prácticas avanzadas e innovadoras que poseen un uso más limitado.

La metodología PMBOK se usará para la gestión de integración del proyecto, debido a que es ampliamente aceptada por el estándar de proyectos. Se debe tener en cuenta que, debido a la naturaleza del presente proyecto de fin de carrera, sólo se hará uso de partes de la metodología que puedan ser aplicables y se adapten mejor a las necesidades del mismo. [PMBOK, 2009]

Como se mencionó se hará uso solo de las siguientes áreas:

- **Gestión de la integración del proyecto**
  En esta área se incluyen los procesos requeridos para asegurar que los varios elementos del proyecto están propiamente coordinados. Para esta área se desarrollará el acta de constitución del proyecto, se gestionará, se ejecutará y se cerrará el proyecto.

- **Gestión del tiempo del proyecto**
  Esta área proyecto contiene los procesos necesarios para asegurar que este se cumpla en el tiempo establecido. Para esta área se definirán las actividades a realizar así como la secuencia en la que se efectuarán, se estimará la duración de cada una y se creará un calendario del proyecto.

- **Gestión de la calidad del proyecto**
  Esta área del proyecto contiene los procesos necesarios para asegurar que cumpla con las necesidades para las cuales se está desarrollando. Para asegurar la calidad del proyecto se determinarán los criterios de aceptación y se hará un monitoreo constante de la calidad en el desarrollo y en los entregables del mismo.

- **Gestión de las comunicaciones del proyecto**
  Esta área incluye los procesos necesarios para asegurar la generación, colección, diseminación y almacenamiento de la información del proyecto. Para esta área se realizará un planeamiento de las comunicaciones de manera que se facilite la distribución de la información del proyecto y se pueda realizar un intercambio de ideas y opiniones entre las personas involucradas en mismo.

- **Gestión de los riesgos del proyecto**
  Esta área contiene los procesos necesarios para identificar, analizar y responder a los riesgos del proyecto. Para esta área se identificarán los riesgos del proyecto, su posible impacto en el mismo y un plan de contingencia o prevención para cada riesgo identificado.

2.4.2 Relacionadas al desarrollo del proyecto

a) Extreme Programming (XP)

XP es una metodología de desarrollo que se enfoca la aplicación correcta de técnicas de programación, comunicación clara y trabajo en equipo. Entre los principales valores del XP tenemos [BECK, ANDRES, 2004]:

- Comunicación
- Simplicidad
- Retroalimentación
- Coraje
- Respeto

Debido a que el proyecto se realizará solo por una persona se harán uso de algunos principios de esta metodología. Estos principios son: coraje y simplicidad. Los demás serán omitidos debido a que se centran más en el trabajo con los clientes, los cuales no existen en este proyecto de fin de carrera. Por último XP valora más el correcto desarrollo del producto que la documentación.

La razón para el uso de esta metodología se centra en la necesidad de enfocarse más en la programación que en la documentación. Esta metodología será usada en la implementación de la herramienta de carga masiva y la implementación de los algoritmos GRASP y genético.

b) Análisis de la varianza (ANOVA)

Es una metodología mediante la cual se puede comparar dos o más muestras con varianza desconocida independientes entre sí. Esta metodología nos permitirá determinar cuál de los dos resultados obtenidos, por el algoritmo genético y GRASP, es el más óptimo.

- **Prueba de Kolmogorov-Smirnov**

"La prueba de Kolmogorov-Smirnov para una muestra es un procedimiento de bondad de ajuste, que permite medir el grado de concordancia existente entre la distribución de un conjunto de datos y una distribución teórica específica. Su objetivo es señalar si los datos provienen de una población que tiene la distribución teórica especificada, es decir, contrasta si las observaciones podrían razonablemente proceder de la distribución especificada" [UNIVERSIDAD DE VALENCIA, 2010].

La razón de usar esta prueba es asegurar que los resultados obtenidos se comportan de manera normal y así poder aplicar la prueba estadística Z para la comparación de dos muestras.

- **Prueba F de Fisher**

"Se emplea para probar si dos muestras provienen de poblaciones que poseen varianzas iguales. Esta prueba es útil para determinar si una población normal tiene mayor variación que la otra y también se aplica cuando se trata de comparar simultáneamente varias medias poblacionales. La comparación simultánea de varias medias poblacionales se conoce como análisis de varianza". [RETURETA, 2010]

La prueba F de Fisher será usada para garantizar la calidad de las soluciones ya que analiza la variación de las muestras obtenidas de ambos algoritmos, se justifica la elección de esta.

- **Prueba Z**

"Se basa en la diferencia entre las proporciones de dos muestras. Bajo la hipótesis nula, se supone que las proporciones de dos poblaciones son iguales y ya que el conjunto estimado para las proporciones se basa en esta hipótesis, se combinan las proporciones de las dos muestras para calcular una estimación global de la proporción de la población común.

Esta prueba puede ser usada para la diferencia entre las proporciones de dos muestras para determinar si existe una diferencia en la proporción de éxitos de los dos grupos (prueba de dos colas) o si un grupo tiene mayor proporción de éxitos que el otro grupo (prueba de una cola)" [BERENSON, LEVINE, KREHBIEL 2006].

La justificación para emplear esta prueba radica en que para comparar los algoritmos GRASP y Genético, se usaran 40 muestras. En caso la cantidad de muestras hubiera sido menor de 35 la prueba a elegir sería la de T-Student.

c) Metodología del algoritmo GRASP

"La Metodología GRASP para construir una solución en lugar de considerar apenas un candidato disponible para el seleccionado (criterio usado por el algoritmo goloso), crea un conjunto de candidatos del que seleccionará uno de ellos aleatoriamente. Por tal motivo, la metodología permite construir varias soluciones diferentes" [GANOZA, SOLANO 2004].

Este algoritmo ya ha sido usado en la solución del problema planteado [SHERALI, BISH, ZHU, 2006] y servirá para comparar a los algoritmos heurísticos que solo conservan la mejor solución con el algoritmo genético.

d) Metodología del algoritmo genético

Los algoritmos genéticos son descritos como algoritmos de búsqueda estocásticos basados en el mecanismo de selección natural y la genética. Estos algoritmos presentan tres principales ventajas cuando son aplicados a problemas de optimización como el que se está tratando [GEN, CHENG, LIN, 2008]:

1. Adaptabilidad: Los algoritmos genéticos pueden manejar cualquier tipo de función objetivo y cualquier tipo de restricciones

2. Robustez: El uso de operadores de evolución hace que los algoritmos genéticos sean muy efectivos en búsquedas globales.

3. Flexibilidad: Los algoritmos genéticos proveen de gran flexibilidad para hibridar con algoritmos heurísticos específicos para ciertos ambientes para obtener una implementación eficiente para un problema específico.

Es en base a estas razones que se ha elegido a este algoritmo como el principal de este proyecto.

3 Alcance

El proyecto de fin de carrera pretende reducir el costo resultante de la asignación de una aeronave a un vuelo e intentar maximizar las posibles ganancias favoreciendo la asignación de aviones con mayor capacidad a vuelos con alta demanda pero tomando en cuenta el consumo de combustible del avión.

Para lograr este objetivo se tomarán en cuenta ciertas variables, los cuales son algunos factores y limitantes de los tantos mencionados en la problemática y que pueden ser consideradas las más importantes para la adecuada asignación de aviones a vuelos. Estas variables son:

- Tiempos muertos: Un tiempo muerto representa un espacio de tiempo en el cual la aeronave no está disponible para atender a ningún vuelo, en nuestros algoritmos estos tiempos corresponderán a los procesos de mantenimiento realizados a la aeronave y el proceso de embarque realizado entre cada vuelo.

- Modelo de la aeronave: En ingles se le denomina *fleet type*. Agrupa a un conjunto de aeronaves que tengan la misma cantidad de asientos disponibles, máxima distancia que puede recorrer, tiempo máximo que debe pasar antes de un mantenimiento general a la aeronave, velocidad de crucero y consumo de combustible.

- Etapas de vuelo: Una etapa de vuelo se inicia con el despegue y termina en el aterrizaje. En la actualidad existen vuelos que poseen varias etapas, y estas influirán en la decisión de seguir o cambiar de aeronave entre diferentes etapas de vuelo.

- Disponibilidad de aeronaves: Es la cantidad de aeronaves disponibles por cada tipo de flota que limitaran la cantidad de asignaciones avión-vuelo que puedan hacer.

- Detalle del vuelo: Se tomará en cuenta la distancia, el origen, el destino y las etapas de vuelo en caso existan más de una.

El proyecto se basará en la implementación de un algoritmo genético para resolver el problema de asignación de flotas en aerolíneas. Para el desarrollo del proyecto es necesaria la implementación de un algoritmo meta-heurístico GRASP que se utilizará con el fin de obtener una población inicial para el algoritmo genético y además será usado en la experimentación numérica.

Tanto en el algoritmo de experimentación (GRASP) como en el de estudio (Algoritmo Genético) se consideraran los diversos factores mencionados. La asignación de tipo aviones a vuelos tomará en cuenta variables que realmente se utilizan en proceso de asignación de flotas en aerolíneas. Cabe resaltar que estas, y solo estas, variables serán consideradas en el desarrollo del proyecto de fin de carrera.

3.1 Limitaciones

Entre las limitaciones del proyecto podríamos encontrar:

- El tiempo de ejecución del algoritmo puede verse afectado dependiendo de las características del hardware y software de la computadora en que se ejecute. Esto se debe a la complejidad de los algoritmos y la cantidad de veces que deben ser ejecutados para obtener los resultados.

- No se tomarán en cuenta más variables a las ya establecidas (tiempos muertos, tipo de flota, etapas de vuelo, disponibilidad de aeronaves y detalle del vuelo)

- Los resultados del proyecto pueden verse afectados por la cantidad y calidad de datos de entrada. La calidad de los datos se mide en que tan cercano o parecidos son a los datos utilizados en la asignación de flotas de una aerolínea.

3.2 Riesgos

Tabla 2: Tabla de riesgos identificados.

Riesgo identificado	Impacto en el proyecto	Medidas correctivas para mitigar
Mala planificación del proyecto	• Entregables revisados a destiempo • Rechazo de entregables • Entregables no presentados	Revisar el plan del proyecto para ajustar los plazos y tiempos de modo que se pueda cumplir con los objetivos en el tiempo establecido
Pérdida parcial o total de la información del proyecto	• Entregables no presentados • Dedicar tiempo a recuperar la información perdida. • Desaprobar el curso	Utilizar 2 servicios en la nube para tener un respaldo de cada uno de los archivos. El principal será Onedrive por su conexión con Microsoft Office 2013 y el secundario será Google Drive, en el cual se hará un respaldo semanal.
Se dificulta el acceso a data real para el problema	• Se deberá generar data de prueba • Resultados erróneos o poco confiables	Obtener data real de prueba con anticipación. Buscar medios alternativos para el acceso a data real de prueba.

4 Justificativa y viabilidad del proyecto

4.1 Justificativa

Según lo visto en el presente proyecto, la asignación de flotas en aerolíneas es fundamental para el correcto uso de los recursos de la empresa, especialmente de las aeronaves. Tomando en cuenta el precio de adquisición o de arrendamiento de una aeronave, una incorrecta asignación avión-vuelo puede afectar negativamente a los beneficios económicos de la empresa así como afectar su reputación ante posibles cancelaciones de vuelos o reembolso de pasajes de vuelo. El crecimiento continuo de las rutas de aviación comercial y la variedad de tipos de aviones en la gran mayoría de aerolíneas a nivel nacional genera la necesidad de asignar correctamente cada tipo de aeronave para maximizar su utilidad. Además, las aerolíneas son un componente indispensable en el crecimiento interno de un país facilitando el contacto entre individuos separados geográficamente.

Por lo tanto, según lo analizado, se pretende implementar una herramienta informática basada en algoritmos genéticos que permita asignación avión-vuelo, de modo que reduzca el costo total de las asignaciones realizadas mediante la optimización de la relación entre los pasajeros por vuelo y el gasto de combustible del avión asignado.

La razón del empleo de algoritmos para la solución del problema radica en el hecho de que un algoritmo permite encontrar soluciones para determinados problemas en diferentes contextos (para este proyecto, la asignación de flotas en aerolíneas), a comparación de otras herramientas, como por ejemplo un sistema de información el cual está orientado más a la administración de datos e información para su posterior uso.

Por otro lado, debido a la complejidad del problema de tipo NP, el desarrollo tendrá que hacer uso de un algoritmo de aproximación, en este caso será un algoritmo genético, ya que permite obtener buenos resultados en tiempos aceptables, a comparación de métodos exactos como la investigación operativa, el cual da soluciones exactas y

óptimas pero están restringida a diversos factores o de alcance limitado. Además, el algoritmo genético se basa en el proceso genético de los organismos vivos y van acorde con los principios de la selección natural y la supervivencia de los más fuertes. Es por esto que son adecuadas al problema en cuestión, ya que imitan el proceso natural y crean soluciones a problemas del mundo real.

La mayoría de soluciones actuales a este problema vienen incluidas en un paquete de software el cual requiere se adquiera un módulo entero que contiene a la solución para el problema de asignación de flotas. Estos paquetes de software generalmente tienen precios elevados, lo cual limita su uso por aerolíneas pequeñas o medianas.

Debido a las razones presentadas, el uso de algoritmos facilitará el tiempo de respuesta generando soluciones óptimas en un tiempo aceptable. Dada la naturaleza del problema, el proyecto serviría como base para el desarrollo de futuras herramientas que intenten optimizar la asignación de flotas en otros contextos.

4.2 Viabilidad

En cuanto a la viabilidad del proyecto, se verifica que existe información referente al tema específico y a la solución. Además se dan las condiciones para desarrollar el proyecto sobre la asignación de flotas en aerolíneas, dado que se cuenta con la posibilidad de obtener la información acerca de este proceso, de las características de las aeronaves y del itinerario de vuelos de una aerolínea.

A continuación se presentan factores que influirán en la viabilidad del proyecto:

a) Viabilidad Técnica

Las herramientas a utilizar en el proyecto de fin de carrera, mencionadas anteriormente, serían principalmente el lenguaje de programación Java (versión 1.7) y el entorno de desarrollo Netbeans (versión 8.0). Además y sin ser menos importantes, el procesador de textos y hojas de cálculo Microsoft Word 2013 y Excel 2013, respectivamente.

En cuanto a la viabilidad técnica, el lenguaje de programación Java funciona con las principales plataformas de hardware y sistemas operativos. Los principales beneficios radican en la independencia de la plataforma, el alto rendimiento, fácil aprendizaje, manejo de estándares, modelo de seguridad probada, entre otros [JAVA, 2013].

Mientras que mediante el uso del IDE Netbeans hará posible depurar el código de una forma muy sencilla, realizar pruebas con valores distintos en las variables, observar el comportamiento de las variables, etc. Todo esto facilita la depuración del software a desarrollar. Los principales beneficios que ofrece el IDE son programación en diversos lenguajes, depuración de código, creación y gestión de diversos proyectos, amplia comunidad de usuarios, internacionalización del software, CVS para gestionar versiones, conexión a servidor, entre otras [NETBEANS, 2013].

La viabilidad de las metodologías de los algoritmos GRASP y Genético es asegurada debido al acceso a documentación detallada sobre la estructura básica de cada uno y a ejemplos de estos aplicados a problemas similares. Además, se tiene experiencia con el desarrollo de algoritmos meta-heurísticos adquirida a lo largo de la carrera. Los principales beneficios de estos algoritmos

se centran en el tiempo necesario para obtener una respuesta buena el cual es menor a los necesarios por los algoritmos exactos.

Por último, el uso de la metodología Extreme Progamming (XP) priorizará el desarrollo del algoritmo sobre la documentación, pero no dejará de lado esta última. El beneficio se centra en la necesidad de enfocarse más en la programación que en la documentación. Además se tiene acceso a documentación detallada sobre la metodología y se tiene experiencia trabajando con otras metodologías ágiles.

Es importante señalar que se poseen los conocimientos necesarios, para el manejo de las herramientas Java y Netbeans, ya que estas herramientas se han ido desarrollando en los últimos semestres de la carrera de Ingeniería Informática. Adicionalmente se contará con la guía tanto del asesor como de los profesores que tienen dominio del tema a presentar. Así mismo, las metodologías elegidas serán una guía y apoyo para cumplir con los objetivos del proyecto.

Finalmente, las herramientas a utilizar en el presente proyecto son de fácil acceso ya que son gratuitas, por lo que no será impedimento para el desarrollo normal del proyecto.

b) Viabilidad Temporal

Sobre el aspecto temporal, la implementación del proyecto se empezará inmediatamente después de culminar el primer curso de tesis 1, obteniendo así un mayor tiempo para culminar el proyecto.

Debido a que el curso de tesis 1 se realiza en la primera parte del ciclo regular 2014, se poseerá un periodo de tiempo extra de 1 mes, tiempo que será también utilizado para lograr un buen desarrollo del proyecto y cumplir los objetivos en los plazos establecidos.

Por lo tanto, la viabilidad temporal está asegurada para el desarrollo del proyecto en el periodo de tiempo establecido. La consideración del tiempo extra que se maneja en los meses mencionados, es importante y será de gran utilidad para la culminación del trabajo.

c) Viabilidad económica

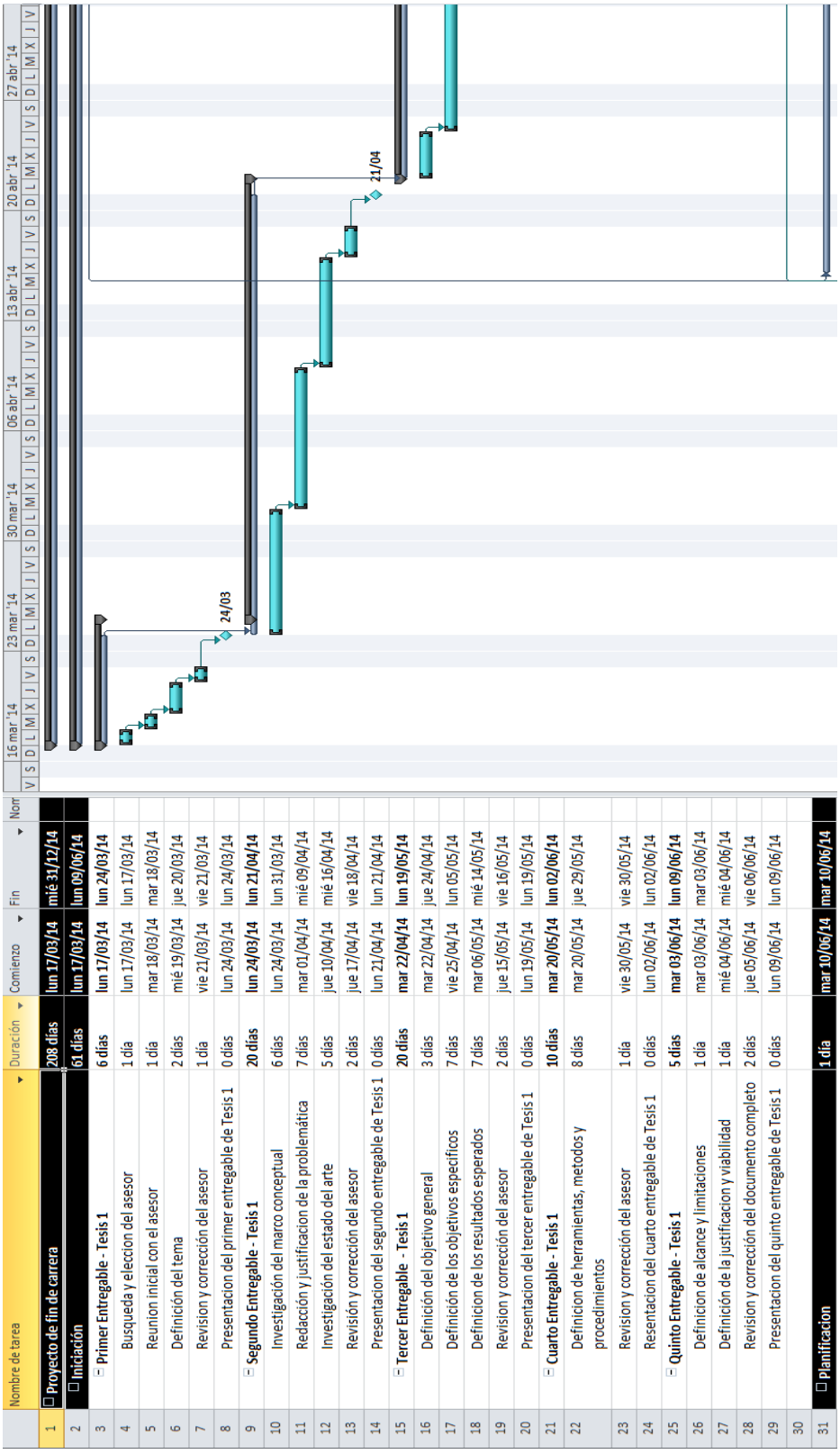
En este aspecto, el presente proyecto no tendrá gastos ya que las herramientas y material necesario son recursos gratuitos o educativos, como son los casos del IDE Netbeans y el lenguaje de programación Java.

En cuanto al costo de equipos, las computadoras utilizadas para el desarrollo del proyecto son de uso propio, de modo que no se incurrirá en gastos para la obtención de nuevos equipos o hardware. Las tecnologías y arquitecturas, que manejan estas computadoras, son las adecuadas y suficientes para el desarrollo del proyecto, debido al alto nivel de procesamiento y suficiente espacio de almacenamiento que requieren la ejecución de los algoritmos.

Por otro lado, no se contará con gastos por personal de desarrollo del proyecto debido a que el presente trabajo es de responsabilidad propia y será realizado individualmente. Aunque siempre se cuente con el apoyo del asesor para resolver algunas dudas.

5 Plan de actividades

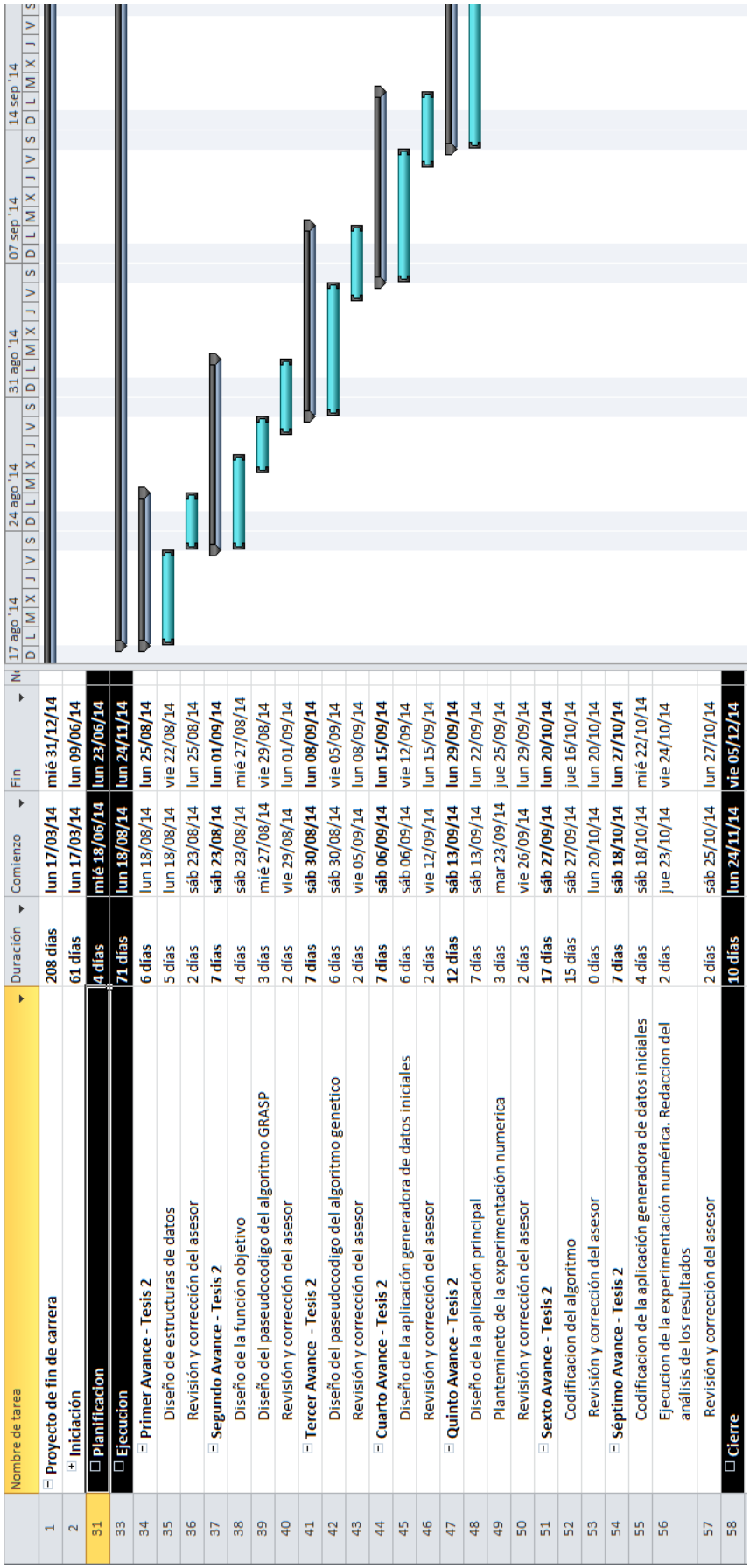
5.1 Iniciación y Planificación



	Nombre de tarea	Duración	Comienzo	Fin	Nom
1	Proyecto de fin de carrera	208 días	lun 17/03/14	mié 31/12/14	
2	Iniciación	61 días	lun 17/03/14	lun 09/06/14	
3	Primer Entregable - Tesis 1	6 días	lun 17/03/14	lun 24/03/14	
4	Busqueda y eleccion del asesor	1 día	lun 17/03/14	lun 17/03/14	
5	Reunion inicial con el asesor	1 día	mar 18/03/14	mar 18/03/14	
6	Definición del tema	2 días	mié 19/03/14	jue 20/03/14	
7	Revision y corrección del asesor	1 día	vie 21/03/14	vie 21/03/14	
8	Presentacion del primer entregable de Tesis 1	0 días	lun 24/03/14	lun 24/03/14	
9	Segundo Entregable - Tesis 1	20 días	lun 24/03/14	lun 21/04/14	
10	Investigación del marco conceptual	6 días	lun 24/03/14	lun 31/03/14	
11	Redacción y justificacion de la problemática	7 días	mar 01/04/14	mié 09/04/14	
12	Investigación del estado del arte	5 días	jue 10/04/14	mié 16/04/14	
13	Revisión y corrección del asesor	2 días	jue 17/04/14	vie 18/04/14	
14	Presentacion del segundo entregable de Tesis 1	0 días	lun 21/04/14	lun 21/04/14	
15	Tercer Entregable - Tesis 1	20 días	mar 22/04/14	lun 19/05/14	
16	Definición del objetivo general	3 días	mar 22/04/14	jue 24/04/14	
17	Definición de los objetivos específicos	7 días	vie 25/04/14	lun 05/05/14	
18	Definicion de los resultados esperados	7 días	mar 06/05/14	mié 14/05/14	
19	Revision y corrección del asesor	2 días	jue 15/05/14	vie 16/05/14	
20	Presentacion del tercer entregable de Tesis 1	0 días	lun 19/05/14	lun 19/05/14	
21	Cuarto Entregable - Tesis 1	10 días	mar 20/05/14	lun 02/06/14	
22	Definicion de herramientas, metodos y procedimientos	8 días	mar 20/05/14	jue 29/05/14	
23	Revision y corrección del asesor	1 día	vie 30/05/14	vie 30/05/14	
24	Resentacion del cuarto entregable de Tesis 1	0 días	lun 02/06/14	lun 02/06/14	
25	Quinto Entregable - Tesis 1	5 días	mar 03/06/14	lun 09/06/14	
26	Definicion de alcance y limitaciones	1 día	mar 03/06/14	mar 03/06/14	
27	Definición de la justificacion y viabilidad	1 día	mié 04/06/14	mié 04/06/14	
28	Revision y corrección del documento completo	2 días	jue 05/06/14	vie 06/06/14	
29	Presentacion del quinto entregable de Tesis 1	0 días	lun 09/06/14	lun 09/06/14	
30					
31	Planificacion	1 día	mar 10/06/14	mar 10/06/14	

Imagen 2: Cronograma de actividades parte 1. Imagen de autoría propia.

5.2 Ejecución y Cierre



	Nombre de tarea	Duración	Comienzo	Fin	Nombres de recursos
1	**Proyecto de fin de carrera**	**208 días**	**lun 17/03/14**	**mié 31/12/14**	
2	**Iniciación**	**61 días**	**lun 17/03/14**	**lun 09/06/14**	
31	**Planificacion**	**4 días**	**mié 18/06/14**	**lun 23/06/14**	
33	**Ejecucion**	**71 días**	**lun 18/08/14**	**lun 24/11/14**	
34	**Primer Avance - Tesis 2**	**6 días**	lun 18/08/14	**lun 25/08/14**	
35	Diseño de estructuras de datos	5 días	lun 18/08/14	vie 22/08/14	
36	Revisión y corrección del asesor	2 días	sáb 23/08/14	lun 25/08/14	
37	**Segundo Avance - Tesis 2**	**7 días**	**sáb 23/08/14**	**lun 01/09/14**	
38	Diseño de la función objetivo	4 días	sáb 23/08/14	mié 27/08/14	
39	Diseño del paseudocodigo del algoritmo GRASP	3 días	mié 27/08/14	vie 29/08/14	
40	Revisión y corrección del asesor	2 días	vie 29/08/14	lun 01/09/14	
41	**Tercer Avance - Tesis 2**	**7 días**	**sáb 30/08/14**	**lun 08/09/14**	
42	Diseño del paseudocodigo del algoritmo genetico	6 días	sáb 30/08/14	vie 05/09/14	
43	Revisión y corrección del asesor	2 días	vie 05/09/14	lun 08/09/14	
44	**Cuarto Avance - Tesis 2**	**7 días**	**sáb 06/09/14**	**lun 15/09/14**	
45	Diseño de la aplicación generadora de datos iniciales	6 días	sáb 06/09/14	vie 12/09/14	
46	Revisión y corrección del asesor	2 días	vie 12/09/14	lun 15/09/14	
47	**Quinto Avance - Tesis 2**	**12 días**	**sáb 13/09/14**	**lun 29/09/14**	
48	Diseño de la aplicación principal	7 días	sáb 13/09/14	lun 22/09/14	
49	Plantemineto de la experimentación numerica	3 días	mar 23/09/14	jue 25/09/14	
50	Revisión y corrección del asesor	2 días	vie 26/09/14	lun 29/09/14	
51	**Sexto Avance - Tesis 2**	**17 días**	**sáb 27/09/14**	**lun 20/10/14**	
52	Codificacion del algoritmo	15 días	sáb 27/09/14	jue 16/10/14	
53	Revisión y corrección del asesor	0 días	lun 20/10/14	lun 20/10/14	
54	**Séptimo Avance - Tesis 2**	**7 días**	**sáb 18/10/14**	**lun 27/10/14**	
55	Codificacion de la aplicación generadora de datos iniciales	4 días	sáb 18/10/14	mié 22/10/14	
56	Ejecucion de la experimentación numérica. Redaccion del análisis de los resultados	2 días	jue 23/10/14	vie 24/10/14	
57	Revisión y corrección del asesor	2 días	sáb 25/10/14	lun 27/10/14	
58	**Cierre**	**10 días**	**lun 24/11/14**	**vie 05/12/14**	

Imagen 3: Cronograma de actividades parte 2. Imagen de autoría propia.

CAPÍTULO 2: MARCO CONCEPTUAL

1 Marco teórico

1.1 Introducción

En el siguiente apartado se procederá a desarrollar los conceptos relacionados a la problemática, los cuales contienen las denominaciones claves que describen el contexto. Asimismo, se explicarán aquellas nociones relacionadas a la propuesta de solución.

1.2 Objetivo del Marco Conceptual

En esta sección, se explicarán los conceptos claves que permitan el entendimiento tanto de la industria aerocomercial y la asignación de recursos, en este caso aviones, y al problema de la asignación de flotas en aerolíneas, generalmente conocido como *airline fleet assigment problem*.

1.3 Conceptos relacionados al problema

1.3.1 Aeronavegabilidad

Se dice del estado en cual debe encontrarse la aeronave para poder realizar vuelos de manera legal. Implica ciertas condiciones no solo del avión, sino de sus tripulantes, sus pasajeros y la carga que lleve. Cada país establece sus propias reglas de aeronavegabilidad. Entre las condiciones requeridas del avión esta la realización de tareas de mantenimiento, las cuales se pueden dividir en 2 tipos [MTC, 2002]:

- Overhaul (Mantenimiento General): Proceso de revisión general, que implica desarmar ciertos componentes vitales de la aeronave, como los rotores de los aviones, hacer un examen minucioso de su estado y volver armarlos en caso no se encuentre ninguna anormalidad. Cada empresa que fabrica aviones presenta un tiempo recomendado en el cual se debe realizar un overhaul a los componentes del avión e incluso ofrecen realizar esta revisión como un servicio adicional.

- Mantenimiento de línea: Agrupa las tareas de mantenimiento realizadas antes de cada vuelo con el objetivo de asegurar la aeronavegabilidad de la aeronave. Entre estas actividades hay inspecciones manuales y análisis de los resultados de los diversos sensores presentes.

Es necesario tomar en cuenta los tiempos requeridos para estos mantenimientos para poder asegurar que la aeronave está lista para atender un nuevo vuelo.

1.3.2 Tipo de flota (fleet type)

También denominado como modelo de aeronave, agrupa a aviones que pertenezcan a un mismo tipo o modelo. Estos poseen la misma distribución de la cabina, requerimientos de tripulación, requerimientos de mantenimientos y capacidad de pasajeros y carga [SHERALI, BISH, ZHU, 2006]. Ejemplo: Airbus 319-200.

1.3.3 Familia de flota (fleet family)

Un conjunto de tipos de flota que poseen la misma configuración de la cabina y requerimientos de tripulación. Esto hace posible que la misma tripulación se pueda desempeñar en cualquiera de los aviones de una misma familia de flota [SHERALI, BISH, ZHU, 2006]. Ejemplo: Airbus 319.

1.3.4 Etapa de vuelo (fligth leg)

Una etapa de vuelo se define como un segmento del vuelo que empieza en un aeropuerto y termina en otro. Este se encarga de conectar dos puntos de parada de dentro de un vuelo, ejemplo: una etapa de vuelo abarca el viaje desde que el avión departe hasta que aterriza [SHERALI, BISH, ZHU, 2006].

1.3.5 Camino (Path)

Es una secuencia de una o más etapas de vuelo entre un origen y destino específicos, empezando cuando el avión sale del aeropuerto [SHERALI, BISH, ZHU, 2006].

1.3.6 Vuelo directo (Through-fligth)

Se denomina vuelo directo al conjunto de dos o más etapas de vuelo que son atendidas por la misma aeronave. Este tipo de vuelos son atractivos debido a que el pasajero no debe moverse del avión aunque este haga paradas intermedias [SHERALI, BISH, ZHU, 2006].

1.3.7 Tiempo de turno (turn time)

Es el tiempo mínimo que requiere una aeronave entre su aterrizaje y su siguiente despegue. Este incluye el tiempo necesario para algunas inspecciones, preparación de la aeronave para su siguiente vuelo y su desplazamiento a la pista. El tiempo de turno promedio para vuelos domésticos es de 30 a 40 minutos [SHERALI, BISH, ZHU, 2006].

1.3.8 Asignación de flotas en aerolíneas (airline fleet assigment problem)

Es un proceso durante el cual se asignan aeronaves a vuelos ya programados, minimizando el costo operativo de cada asignación y maximizando la ganancia obtenida por número de pasajeros atendidos en el vuelo. En las aerolíneas se resuelve mediante la formulación de un problema de programación entera mixta basado en la red de vuelos de la aerolínea.

Las principales restricciones que posee este problema son:

- Restricción de cobertura: Cada etapa de vuelo debe tener asignado un tipo de flota.
- Restricción de balance: Se debe mantener la conservación de flujo.
- Disponibilidad de aeronaves: No se deben asignar más aeronaves que las que se tienen disponibles.

A continuación se describirán dos de los principales modelamientos para el problema [SHERALI, BISH, ZHU, 2006]:

- **Modelo 1: Usar los arcos como representación de conexiones**

Este modelo es denominado estructura de redes de conexión; en esta red, los nodos representan puntos de tiempo cuando los vuelos arriban o departen. Además existen dos nodos imaginarios llamados fuente maestro (master source) y sumidero maestro (master sink) que simulan el principio del día y el fin de este respectivamente. En este modelo hay tres tipos de arcos que representan los diferentes tipos de conexión:

- Arcos de conexión de vuelo: Enlazan los nodos de partida con los nodos de llegada.
- Arcos de finalización: Enlazan los nodos de llegada con el nodo de sumidero maestro representando que la aeronave se mantiene en la última estación por el resto del día
- Arcos de iniciación: Enlazan el nodo fuente maestro con los nodos de partida representando que las aeronaves están presentes en la estación al inicio del día.

Cada una de las conexiones debe ser posible respetando los tiempos de llegada y de partida por lo que se requiere tomar en cuenta el tiempo de turno entre la llegada del avión y su siguiente partida.

A continuación (imagen 4) se muestra un ejemplo de la representación de los vuelos en una estructura de redes de conexión:

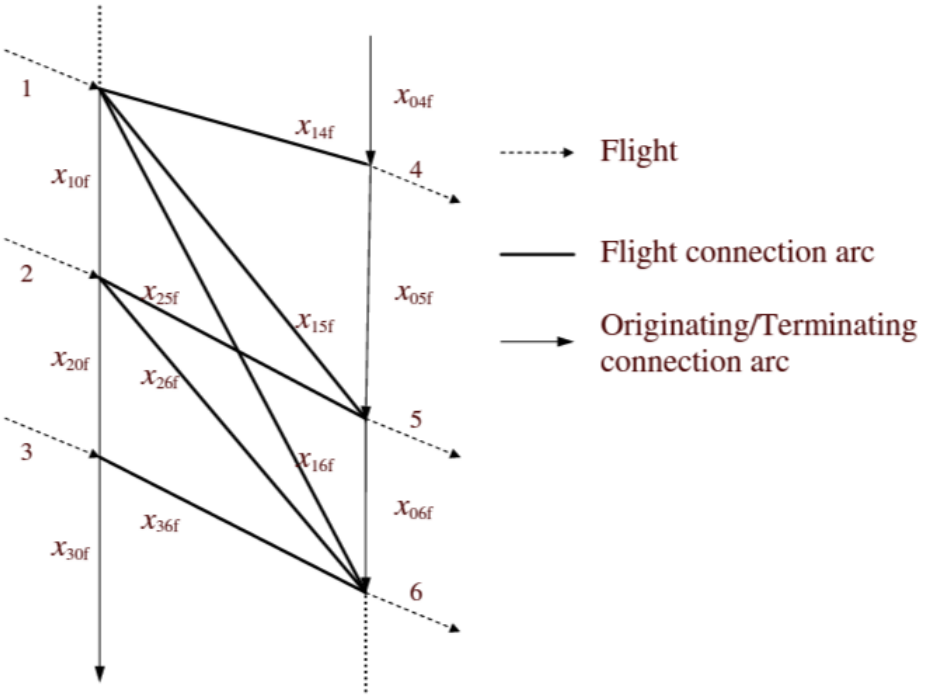


Imagen 4: Representación en forma de estructura de redes de conexión. Imagen recuperada de [SHERALI, BISH, ZHU, 2006].

Los arcos del 1 al 6 representan los vuelos, pero no son incluidos en las variables de decisión; las cuales son las variables binarias de conexión x_{ijf} que indican si el tipo de flota f cubrirá la conexión i,j, donde x_{iof} representan conexiones de finalización y x_{0jf} las conexiones de iniciación.

El programa matemático basado en las redes de conexión es el siguiente:

Maximizar:

$$\sum_{i \in L \cup \{0\}} \sum_{j \in L} \sum_{f \in F} p_{jf} x_{ijf} - c \sum_{j \in L} \sum_{f \in F} x_{ojf} \qquad 1a$$

Fórmula 1: Fórmula de maximización de beneficios en el modelo de red de conexión.

Sujeto a:

$$\sum_{i \in L \cup \{0\}} \sum_{j \in L} x_{ijf} = 1 \quad \forall j \in L, \qquad 1b$$

$$\sum_{i \in L \cup \{0\}} x_{ilf} - \sum_{j \in L \cup \{0\}} x_{ljf} = 0 \ \ \forall l \in L, f \in F, \qquad 1c$$

$$\sum_{i \in D_s} x_{0if} - \sum_{i \in A_s} x_{i0f} = 0 \ \ \forall s \in S, f \in F, \qquad 1d$$

$$\sum_{i \in D_s} x_{0if} \leq A_f \quad \forall f \in F, \qquad 1e$$

$$x \ binario \qquad 1f$$

Fórmula 2: Limitaciones en el modelo de red de conexión.

Donde:

S	Conjunto de las estaciones, indexado por s
F	Conjunto de tipos de flota, indexado por f
L	Conjunto de etapas de vuelo, indexado por i, j
D_s	Conjunto de etapas de partida para la estación $s, s \in S$
A_s	Conjunto de etapas de salida para la estación $s, s \in S$
x_{ijf}	{ 1, si la conexión i, j será cubierta por el tipo de flota f 0, en caso contrario
p_{jf}	Beneficio obtenido de usar el tipo de flota $f$al vuelo con nodo de partida j . Es una combinación de ganancia, utilización de aeronave, etc.
c	Es el costo de asignar cualquier tipo de flota a un vuelo

En este modelo la restricción de cobertura (1b) requiere que cada vuelo esté precedido por una llegada o un arco de iniciación cubierto por un tipo de flota. En el caso que no sea requerido que todos los vuelos sean atendidos se puede usar $\sum_{i \in L \cup \{0\}} \sum_{J \in L} x_{ijf} \leq 1 \quad \forall j \in L$ en vez de (1b). La restricción de balance (1c) segura el balance del flujo en cada etapa de vuelo de la red para cada tipo de flota. La restricción de balance de planeamiento (1d) asegura que el mismo número de aeronaves de cada tipo esté presente en cada estación cada noche para que la misma asignación se pueda repetir diariamente. La restricción de disponibilidad (1e) limita el número de aviones usados sea A_f, el cual representa al número de aviones disponibles por cada tipo de flota.

- **Modelo 2: Usar los arcos como representación de una etapa de vuelo**

Este modelamiento es conocido como estructura de redes tiempo-espacio, la cual, a diferencia de las redes de conexión, se centra en representar las etapas de vuelo y le deja al modelo decidir cuáles son las conexiones, siempre y cuando estas sean

posibles para las consideraciones de tiempo y espacio. Esto provee de una mayor libertad para establecer conexiones, mientras que se reducen la cantidad de variables de decisión debido a que el número de etapas de vuelo es mucho menor que el número de conexiones posibles. Sin embargo este modelo no es capaz de distinguir entre las específicas aeronaves en tierra, lo que limita su aplicación en el subsecuente problema de ruteo [SHERALI, BISH, ZHU, 2006].

La representación por redes tiempo-espacio superpone un conjunto de redes, una para cada tipo de flota. Esto permite tener tiempos de vuelo y tiempos de turno para cada tipo de flota. En caso de que estos tiempos para un grupo de tipos de flota no sea muy diferente, se puede construir una red compuesta. En la red de cada tipo, cada evento de aterrizaje o partida de un vuelo que ocurre en un tiempo específico es representado por un nodo. Con el objetivo de permitir solo conexiones factibles, el nodo de aterrizaje es puesto en el tiempo que el avión está listo para salir, es decir el tiempo de aterrizaje real más el tiempo de turno necesario.

Existen tres tipos de arco: arcos de tierra que representan a una aeronave permaneciendo en una estación por cierto periodo de tiempo, arcos de vuelo representando las etapas de vuelo, y arcos de madrugada que conectan los últimos eventos del día con los primeros eventos del día. A continuación se muestra un ejemplo de la representación de los vuelos en una estructura de redes de tiempo-espacio:

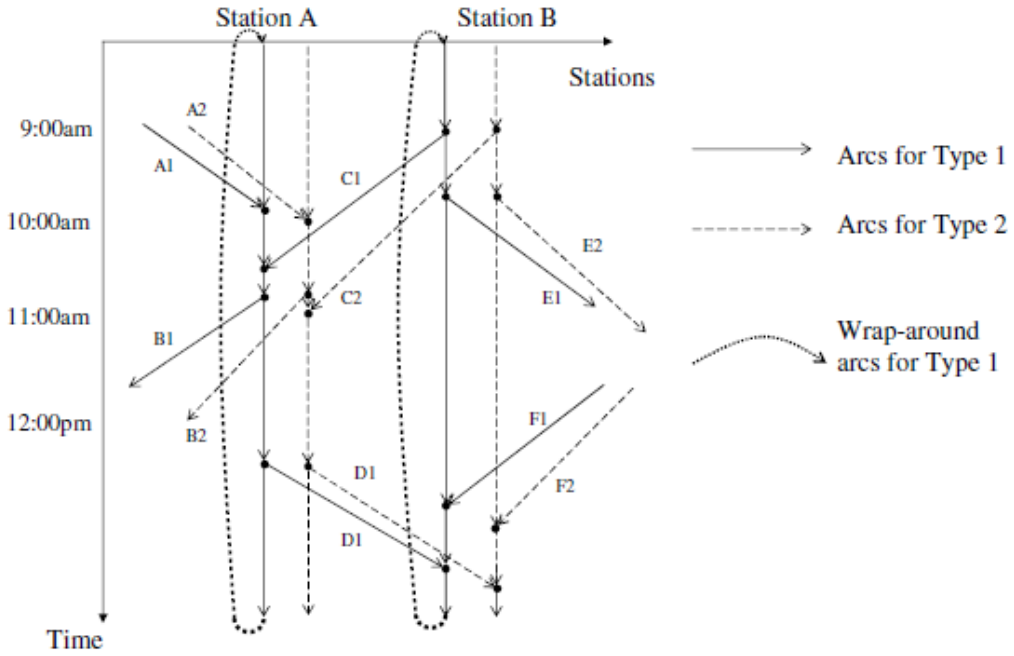


Imagen 5 : Representación en forma de estructura de redes de tiempo-espacio. Imagen recuperada de [SHERALI, BISH, ZHU, 2006].

Las flechas inclinadas representan los arcos de vuelo, las flechas verticales representan los arcos de tierra y las flechas curvas representan los arcos de madrugada. En la figura se pueden observar al par de arcos A_1 y A_2 que representan al vuelo A siendo atendido por los tipos 1 y 2 respectivamente. En este caso, la aeronave de tipo 2 requiere de un tiempo de turno más largo, razón por la cual sus tiempos de llegada ocurren después que los del tipo 1.

El programa matemático basado en las redes de tiempo-espacio es el siguiente:

Maximizar:

$$\sum_{l \in L} \sum_{f \in F} c_{fl} x_{fl} \qquad 2a$$

Fórmula 3: Fórmula de maximización de beneficios en el modelo de red de tiempo-espacio

Sujeto a:

$$\sum_{f \in F} x_{fl} = 1 \quad \forall l \in L, \qquad 2b$$

$$\sum_{o \in S} x_{fost} + y_{fst^- t} - \sum_{d \in S} x_{fsdt} - y_{fstt^+} = 0 \ \ \forall \{fst\} \in N, \qquad 2c$$

$$\sum_{l \in O(f)} x_{fl} + \sum_{s \in S} y_{fst_n t_1} \leq A_f \ \ \forall f \in F, \qquad 2d$$

$$x \ binario \qquad 2e$$

Donde:

Fórmula 4: Limitaciones en el modelo de red tiempo-espacio.

Donde:

S	Conjunto de las estaciones, indexado por o, s ó d
F	Conjunto de tipos de flota, indexado por f
L	Conjunto de etapas de vuelo, indexado por l ó $\{odt\}$, donde $o, d \in S$ y t denota el tiempo en el cual la aeronave parte de o ó está listo para su siguiente salida en d.
N	Conjunto de nodos en una red, indexados por $\{fst\}$ donde $f \in F, s \in S$ y t denota el tiempo del evento
$O(f)$	Conjunto de arcos para el tipo de flota f que cruzan la línea de tiempo de conteo de aviones, $f \in F$
A_f	Número de aeronaves disponibles por tipo de flota $f, f \in F$
x_{fl}	$\begin{cases} 1, \text{ si el tipo de flota } f \text{ cubre la etapa de vuelo } l, f \in F, l \in L \\ 0, \text{ en caso contrario} \end{cases}$
$y_{fstt'}$	Flujo de la aeronave en el arco de tierra que va del nodo $\{fst\} \in N$ hacia el nodo$\{fst'\} \in N$ en la estación $s \in S$ en la red del tipo de flota f, para $f \in F$, donde $t' > t$ en general y $t' \leq t$ para arcos de madrugada.
t^-, t^+	El tiempo precediendo y sucediendo t, respectivamente, en la línea de tiempo.

Las ecuaciones (2b), (2c) y (2d) corresponden a las restricciones de cobertura, de balance y de disponibilidad de aeronaves respectivamente [SHERALI, BISH, ZHU, 2006].

Una de las mayores preocupaciones en la formulación de modelos de asignación de flota es mantener el seguimiento constante de las flotas en diferentes aeropuertos en cualquier punto dado del tiempo, por eso Massoud Bazargan describe un método de red de espacio temporal para formular el problema [*apud* CORTES, 2013].

1.4 Conceptos relacionados a la solución

1.4.1 Problema NP

Un problema NP es una clase de problema de decisión que puede ser resuelto por una máquina de Touring No Determinísticas en Tiempo Polinómico, es por eso que se les denomina Non-Deterministic Polynomial Time [COOK, 1971]. Estos problemas son clasificados como intratables, lo que significa que si se encuentra una solución a este, la solución obtenida no es la más óptima. Entre los problemas que pertenecen a esta clase están:

- Scheduling
- Agente viajero
- Timetabling

1.4.2 Algoritmos Heurísticos

Los algoritmos heurísticos son aquellos que encuentran soluciones "suficientemente buenas" a un problema sin importar si estas son correctas u óptimas. Los métodos heurísticos le dan más importancia a la rapidez y eficiencia de tiempo que a la precisión y la calidad [BROWNLEE, 2011].

Existen ciertas características que se deben considerar para decidir la aplicación de un método heurístico en la resolución de un problema [MOSCATO, 1996]:

- El problema no requiere una solución exacta.
- No existe un método exacto de resolución o de existir, requiere mucho tiempo y poder de procesamiento.
- Los datos son poco fiables o poco variados.
- Existen limitaciones de tiempo y poder de procesamiento.
- Es un paso intermedio en la aplicación de otro tipo de algoritmos.

1.4.3 Algoritmos Meta-heurísticos

Un algoritmo meta-heurístico es un proceso de generación iterativa que guía a una heurística subordinada combinando inteligentemente diferentes conceptos para explorar y explotar los espacios de búsqueda usando estrategias de aprendizaje en la estructura de información con el objetivo de encontrar soluciones casi óptimas [OSMAN, KELLY, 1996].
Las principales propiedades que caracterizan a los algoritmos meta-heurísticos son [BLUM, ROLI, 2003]:

- Los algoritmos meta-heurísticos son estrategias que guían el proceso de búsqueda.
- Su objetivo es explorar eficientemente el espacio de búsqueda para encontrar soluciones casi óptimas
- Las técnicas que constituyen a los algoritmos meta-heurísticos van desde una simple búsqueda local hasta complejos procesos de aprendizaje.

- Los algoritmos meta-heurísticos son aproximados y no determinísticos.
- Pueden incorporar mecanismos que permiten evitar que las soluciones se encuentren atrapadas en áreas confinadas del espacio de búsqueda.
- Los conceptos básicos de los algoritmos meta-heurísticos permiten que sean descritos en un nivel abstracto.
- Las meta-heurísticas no son específicas para un problema.

1.4.4 Algoritmos GRASP

El algoritmo GRASP (Greedy Randomized Adaptative Search Procedure) fue desarrollada por T. Feo y M. Resende a finales de los años 80. Este algoritmo consiste de un proceso iterativo, donde cada iteración consta de dos fases, una fase de construcción y una fase de búsqueda local. En la fase de construcción se construye una posible solución, elemento por elemento, en donde la decisión de cual agregar a la solución es determinada por el ordenamiento de una lista de candidatos respecto a una función codiciosa (greedy function), la cual mide el beneficio de seleccionar cada uno. En la fase de búsqueda local se hace uso de un algoritmo que trabaja de forma iterativa reemplazando la solución creada por una mejor en su vecindad. Al final de todo proceso la mejor solución es la que se presenta como el resultado [FEO, RESENDE, 1995].

Este algoritmo ha tenido aplicaciones en varias áreas como [RESENDE, RIBEIRO, 2010]:

- Ruteo
- Lógica
- Cobertura y segmentación
- Optimización en grafos
- Asignación
- Telecomunicaciones
- Biología
- Dibujado de grafos y mapas

Aquí se presenta la estructura del algoritmo GRASP [TUPIA, 2009]:

Procedimiento GRASP (Instancia del Problema)
1. *Leer (Instancia)*
2. *Mientras < no se cumpla condición de parada > hacer*
   2.1.1 Procedimiento Construcción (S_k)
   2.1.2 Procedimiento Mejorar (S_k)
   Fin Mientras
3. *Retornar (Mejor S_k)*

Fin GRASP

Imagen 6: Seudocódigo general del algoritmo GRASP. Imagen recuperada de [TUPIA, 2009].

Y la estructura de la fase de construcción del algoritmo GRASP:

Procedimiento GRASP Construcción (N, c, E, F, S, a)
Inicializar c, E, a
S = Ø
N = E
1. Mientras <no se cumpla condición de parada> hacer
Inicio
1.1 RCL = Ø
1.2 β = Mejor {c(x): x ϵ N}
1.3 τ = Peor {c(x): x ϵ N }
1.4 RCL = { ∀ x ϵ N: β≤c(x)≤ β + α(τ- β)}
1.5 α = Aleatorio(RCL)
1.6 Si S ∪ {a} *ϵ F → S* ∪ {a}
1.7 N = N – {a}
1.8 Adaptar c
Fin Mientras
Fin GRASP Construcción

Imagen 7: Seudocódigo general del proceso de construcción del algoritmo GRASP. Imagen recuperada de [TUPIA, 2009].

1.4.5 Algoritmos Genéticos

Los algoritmos genéticos fueron inventados por Jhon Holland en 1960 y desarrollados por Holland, sus estudiantes y sus colegas en la universidad de Michigan. Holland presento a los algoritmos genéticos (AG) como una abstracción de la evolución biológica dando un marco de trabajo teórico para adaptaciones del AG [MITCHEL, 1999].

El AG consiste en pasar de una población de cromosomas, los cuales representan posibles soluciones, a una nueva población usando algún método de selección natural, como selección del más apto, junto con los operadores genéticos de cruce, mutación e inversión.

Se pueden señalar tres ventajas importantes de los AG:

- No poseen demasiados requerimientos matemáticos sobre la optimización de problemas. Esto se debe a su naturaleza evolutiva, el AG buscará soluciones sin importar el funcionamiento interno del problema.

- Los operadores de evolución hacen al AG efectivo en realizar búsqueda global. Los métodos tradicionales requieren que el problema posea cierta convexidad para garantizar que cualquier máximo local es un máximo global.

- El AG provee de gran flexibilidad para trabajar en conjunto con heurísticas dependientes para hacer una implementación eficiente para problemas específicos.

Se pueden definir los siguientes conceptos para los GA:

- Gen: es la mínima unidad de representar la instancia de un estado o “alelo” (ceros o unos).

- Cromosoma: Conjunto de elementos del problema, genes, que representan una solución al mismo. También se le denomina individuo.

- Población Inicial: Es una población de individuos con la cual comenzará la ejecución el AG. Está compuesta de individuos, donde cada uno es una cadena binaria de caracteres con una longitud exacta.

- Evaluación: En cada generación del AG, cada individuo de la población es evaluado contra un ambiente desconocido. El valor de aptitud está asociado con la de la función objetivo.

- Operadores Genéticos: Son mecanismos para relacionar a los cromosomas "padres" y generar cromosomas hijos en una nueva generación. Los operadores más conocidos son: cruce (crossover), mutación (mutation) e inversión (inversion).

- Aberración: Es una solución inválida al problema. Es decir que no cumple con los requerimientos o restricciones propias del problema.

- Función Fitness: Es una función aplicada a un cromosoma que permite evaluar la calidad de la solución.

Con el paso del tiempo se han desarrollado variaciones de los AG. Entre los más destacados se encuentran [GUO, WANG, HAN 2010]:

- **Algoritmo Genético Hibrido (AGH):** La optimización de los algoritmos AG tradicionales tienen defectos inherentes. La mayoría de los métodos de optimización tradicionales, aplicado al diseño de ingeniería, requiere un mejor conjunto de valores iniciales para las variables de diseño y, luego converger rápidamente para generar buenos resultados. Sin embargo, la mayoría de los algoritmos enfrentan las mismas dificultades, como un proceso de prueba y error en la búsqueda de un mejor conjunto de variables iniciales de diseño o una lenta convergencia. El conjunto de variables iniciales de diseño es determinada por la intuición de la ingeniería en general y los diferentes conjuntos de variables de diseño iniciales darán, en general, diferentes resultados óptimos. Por lo tanto, la forma de seleccionar los mejores valores iniciales de las variables de diseño es un paso crítico para los métodos tradicionales

- **Algoritmo Genético de Intervalo (AGI):** Los algoritmos de optimización han sido desarrollados en la ingeniería por un largo tiempo, sin embargo, la mayoría de estos tratan los métodos derivados del diseño variables para un desempeño óptimo. Sin embargo, no suele ser fácil fabricar variables de diseño exactas en ingeniería, debido a imprecisiones de medición o errores en el proceso de fabricación mismo. Además, este proceso de fabricación es a menudo muy costoso. La optimización del intervalo es uno de los algoritmos que puede superar estas dificultades. En la optimización tradicional de intervalo, el análisis del intervalo es presentado en el proceso de optimización de intervalo. El propósito del análisis del intervalo es proporcionar límites superior e inferior del efecto de todos estos errores en una cantidad calculada.

- **Algoritmo Genético Híbrido de Intervalo (AGHI):** Con el avance de la capacidad computacional, el análisis de intervalos y la optimización de intervalos son respetados y aplicados en la mayoría de ambientes en los últimos años, como la matemática, bioquímica, ingeniería y otros. En la ingeniería la optimización de intervalos ha sido aplicada extensivamente en el diseño de estructuras. Aun cuando la optimización de intervalos ha sido aplicada en la ingeniería por mucho tiempo, el análisis de intervalos y el cálculo matemático siguen siendo indispensables.

La forma básica del algoritmo genético es el siguiente [TUPIA, 2009]:

Inicio Algoritmo Genético
1. *Generar Población Inicial Po.*
2. *P_Anterior = Población Inicial.*
3. *Mientras <no se cumpla condición de parada> hacer*
   Inicio
   3.1 Aplicar Operador Casamiento (tasa, P_Anterior, P_Nueva).
   3.2 Aplicar Operador Mutación (tasa, P_Anterior, P_Nueva).
   3.3 Aplicar Operador Inversión (tasa, P_Anterior, P_Nueva).
   3.4 Eliminar Aberraciones (P_Anterior, P_Nueva).
   3.5 Aplicar Mecanismo Selección Aleatoria (P_Nueva, f_ fitness).
   3.6 Medir Convergencia (P_Nueva, mejor_individuo).
   3.7 P_Anterior = P_Nueva.
   3.8 P_Nueva = φ.

Fin Mientras 3.
Fin Algoritmo Genético.

Imagen 8: Seudocódigo general del algoritmo genético. Imagen recuperada de [TUPIA, 2009].

1.5 Marco regulatorio / legal

El Perú como cualquier otro país posee un conjunto de normas llamadas Regulaciones Aeronáuticas del Perú (RAP). En estas se indican los requerimientos mínimos necesarios para que se le otorgue a la aeronave el permiso de levantar vuelo. Entre las normas hay dos que pueden incurrir en la solución de la problemática:

- La extensión 43 de la RAP indica que las aeronaves deben pasar por una revisión antes de cada vuelo, además cada cierto espacio de tiempo se debe realizar un mantenimiento exhaustivo a todos los componentes de la aeronave, especialmente a los vitales, como los rotores [MTC, 2013].

El factor antes mencionado implica que se debe dedicar una cierta cantidad de tiempo entre los vuelos para hacer las verificaciones pertinentes. Además según la Ley Nº 27261 (Ley de Aeronáutica Civil del Perú) los explotadores de un aeródromo son los responsables de mantener sus aeronaves en condiciones óptimas [MTC, 2000].

En conclusión, se deberá tomar en cuenta el espacio de tiempo requerido entre vuelos para cumplir con la regulación en la solución a la problemática.

1.6 Conclusión

La gama de conceptos necesarios para la correcta comprensión del problema de asignación de flotas en aerolíneas es amplia. Se requiere conocer desde las ventajas y desventajas de los posibles modelos y algoritmos aplicables para generar una solución hasta las características importantes de los principales actores, que son los aviones y los vuelos. Por último, también se debe tener en cuenta el ámbito regulatorio, que varía de país en país, donde se asignan responsabilidades a las aerolíneas y se exige que determinadas condiciones se cumplan antes de realizar un vuelo. Todos estos factores influyen en la complejidad que posee el problema y permiten una mejor decisión en la elección de un algoritmo para su solución.

2 Estado del arte

2.1 Introducción

La revisión de la literatura para el problema de asignación de flotas en aerolíneas es necesaria para conocer las diversas formas de solución que se han desarrollado y que se usan en la actualidad.

2.2 Objetivos de la revisión del estado del arte

El objetivo principal de esta revisión es conocer las distintas soluciones que se han realizado o planteado hasta el momento para el problema u otros afines a este. Algunas soluciones son exactas mientras que otras son aproximadas, se desea evaluar sus ventajas y desventajas, así como detectar áreas no resueltas del problema.

2.3 Método usado en la revisión del estado del arte

El estado del arte se realizó en base a una revisión sistemática. Esta elección se debe a la ventaja que ofrece la misma para minimizar la parcialidad de la investigación mediante una búsqueda bibliográfica exhaustiva.

En primer lugar, se planteó una pregunta de investigación: ¿Cuáles son las iniciativas para resolver el problema de asignación de flotas en aerolíneas comerciales y que al mismo tiempo presentan un caso de estudio real? En base a esta pregunta, se elaboró una lista de términos usados para resolver la pregunta de investigación. La lista de términos está conformada por los siguientes elementos:

- Aircraft scheduling
- Airline fleet scheduling
- Algoritmo
- Software
- Meta-heurística

En segundo lugar, se realizó búsqueda en base a cadenas elaboradas a partir de la combinación de los elementos de la lista de términos. En la presente revisión, las cadenas de búsqueda fueron usadas en librerías digitales como Scopus, IEEE, Proquest, EBSCOT y Springer; también se buscó en Google Scholar.

Una vez recabada una gran cantidad de estudios primarios, se realizó un proceso de filtración, en el cual se analizó el título y el “abstract” del artículo para saber si este aportaba algún conocimiento sobre el tema. Los objetivos principales de estos artículos debían proponer una solución al problema de asignación de flotas en aerolíneas o presentar un resumen del estado del problema en la actualidad.

2.4 Formas aproximadas de resolver el problema

2.4.1 Asignaciones periódicas de flotas con ventanas de tiempo, limitaciones de espacio y utilidades tiempo dependiente

La técnica de ramificación y poda (branch and bound) se suele interpretar como un árbol de soluciones, donde cada rama nos lleva a una posible solución posterior a la actual. La característica de esta técnica es que el algoritmo se encarga de detectar en qué ramificación las soluciones dadas ya no están siendo óptimas, para "podar" esa rama del árbol y no continuar malgastando recursos y procesos en casos que se alejan de la solución óptima [CORTES, 2013].

Se utilizó una mejora del algoritmo de ramificación y poda que aumente el beneficio obtenido de la asignación. Esto se logró agregando variables para ventanas de tiempo, restricciones de espacio y ganancias dependientes del tiempo. Se realizó una experimentación numérica que usó tres conjuntos de datos.

Data set	Airports	Flights	Aircraft	Aircraft types	Spacing constraints	Overlapping flights
I	45	283	93	13	192	30
II	62	411	114	13	307	32
III	47	412	90	7	345	0

Imagen 9: Estadística de los datos de prueba. Imagen recuperada de [BÉLANGER *et all*, 2006].

Comparado con versiones tradicionales del algoritmo, la nueva solución logró obtener un mejor resultado y en un tiempo más corto, hasta una hora y media. Pero se notó que la solución del problema central acaparaba cerca del 95% de poder de procesamientos del CPU y este sería el siguiente problema a resolver. En general esta nueva aplicación permitió reducir el número de aviones requeridos en varias ocasiones, donde, en la mayoría de los casos, se aumentaron los beneficios económicos. Además la incorporación de variables penalizadoras a las variables de la red de flujo permitió un adecuado control de las restricciones de espacio [BÉLANGER *et all*, 2006].

2.4.2 Aplicación de un algoritmo recocido simulado al problema de asignación de flotas en aerolíneas

El algoritmo de recocido simulado (simulated annealing) es una de las meta-heurísticas más antiguas. Este algoritmo simula el proceso de recocido del metal y el cristal. Para evitar quedar atrapado en un óptimo local, el algoritmo permite elegir una solución peor que la solución actual. En cada iteración se elige a partir de la solución actual s una solución s' del vecindario $N(s)$. Si s' es mejor que s, se sustituye s por s'como la solución actual. Si la solución s' es peor, entonces es aceptada con una determinada probabilidad que depende de la temperatura actual T y de la variación de la función de ajuste [CORTES, 2013]. Con esto intenta lograr que la solución actual converja a la solución óptima.

Para asegurar una convergencia estable, el algoritmo recocido simulado debe elegir una temperatura t_0 la cual debe garantizar la misma probabilidad en cada estado dentro de la distribución de equilibrio. Entonces, se necesita determinar el valor de evaluación de los límites inferior y superior. En el peor de los casos el beneficio será 0 y en caso contrario será el máximo posible. Para demostrar la factibilidad de este método, se hizo una simulación en el aeropuerto de Haikou, en cual existían 130 vuelos diarios, 7 tipos de aeronaves y 22 aeronaves operando. El resultado de 10 simulaciones, que demoraron en promedio 04.18 segundos, indica que el método propuesto es eficiente y estable [WANG, SUN, 2010].

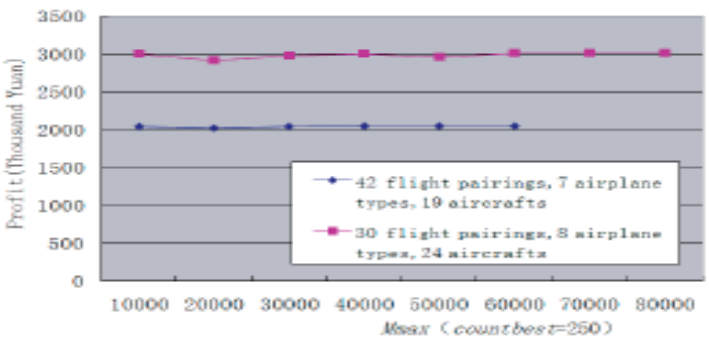


Imagen 10: Influencia del número de iteraciones en el algoritmo. Imagen recuperada de [WANG, SUN, 2010].

2.5 Productos comerciales para resolver el problema

2.5.1 Awery Aircraft Management

Es un módulo perteneciente al ERP ofrecido por la empresa Awery Aviation Solutions y está diseñado para controlar y gestionar la flota, seguir los detalles del vuelo, organizar una correcta planificación de aviones que maximice su utilización [AWERY, 2014].

Entre sus principales características tenemos:

- Base de datos de aeronaves y tipos de aeronaves con todos los detalles, especificaciones, galería de fotos y ratios.
- Gestión de aeronaves (arrendadas o propias), su documentación e itinerario de mantenimiento.
- Generación de reportes personalizables.
- Documentos de control de la aeronave y del operador, con alertas de expiración y notificaciones.

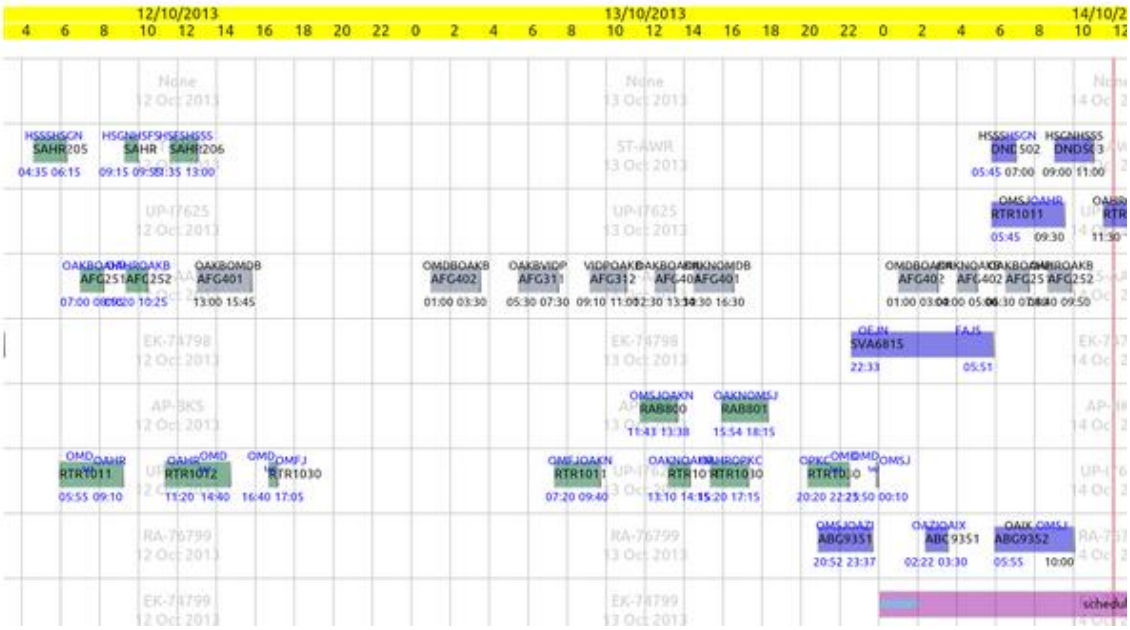


Imagen 11: Interfaz de planificación de Awey Aircraft Management. Imagen recuperada de [AWERY, 2014]

En la imagen superior se puede observar el itinerario de varias aeronaves. Cada bloque lleva la hora de inicio y fin, así como el código del aeropuerto de salida y del aeropuerto de llegada.

2.5.2 ARMS V2 - Aviation Resource Management System

Es un sistema de información ofertado por Sheorey Digital. Es una combinación de un sistema de planeamiento de recursos empresariales (ERP), un sistema de soporte a decisiones (DSS), solución a la automatización del flujo de trabajo (WFA), sistema de información de gestión ejecutiva (EMIS) y un sistema de manejo de la documentación (IDMS) [SHEOREY DIGITAL, 2014].

Entre los sub-sistemas que posee el programa, el sub-sistema de planeamiento y despacho de vuelos (FPDS) provee de todas las funcionalidades para un centralizado y remoto despacho de vuelos. Entre sus módulos se tiene [SHEOREY DIGITAL, 2014]:

- Módulo de planeamiento de vuelos computarizado
- Módulo de carga u recorte
- Documentación de pre-vuelo
- Módulo de auto-capacitación de la tripulación
- Bases de datos FPDS

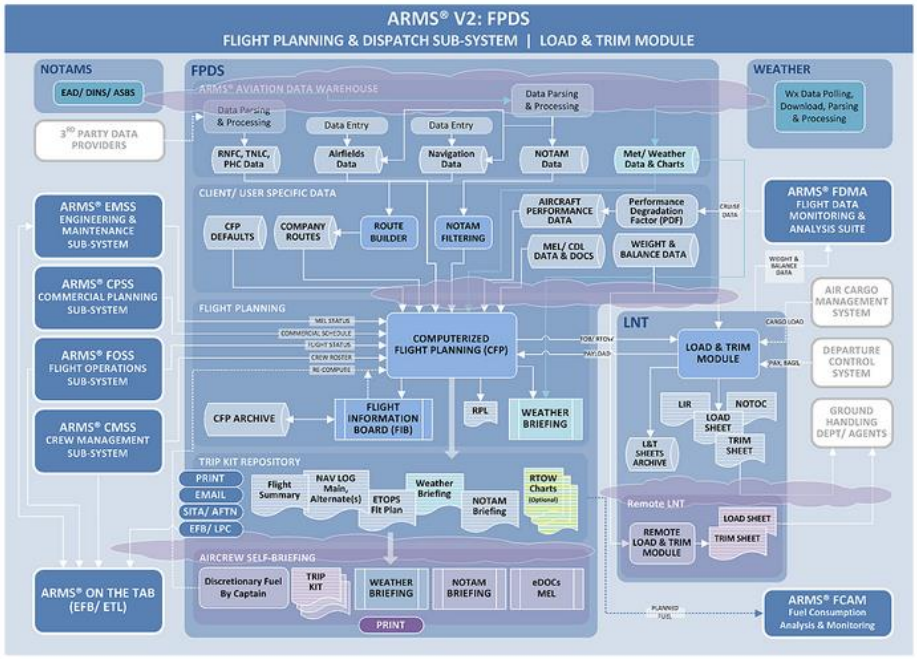


Imagen 12: Esquema de las funcionalidad del módulo FPDS. Imagen recuperada de [SHEOREY DIGITAL, 2014].

2.6 Conclusiones sobre el estado del arte

El contexto actual de las aerolíneas en muchos países ha vuelto necesario que estas comiencen a invertir en el desarrollo de algoritmos heurísticos o meta-heurísticos para la solución del problema de asignación de flotas. Esto se debe mayormente a la creciente complejidad del problema por el aumento de vuelos, aviones y la necesidad crítica de reducir el costo de la asignación y maximizar los posibles beneficios. Ante esta situación las soluciones por programación lineal o algoritmos exactos están dejando de ser usadas por su ineficiencia.

Además, las aerolíneas tiene a su disposición paquetes de software que les brindan solución no solo a este proceso, sino que poseen módulos enfocados a las áreas de la empresa. Pero a veces la adquisición de estos paquetes de software puede resultar muy costosa para aerolíneas pequeñas o se requiere un largo proceso de configuración para que el sistema se adapte a las necesidades de la empresa. Estas razones justifican que ciertas aerolíneas se decidan por usar un algoritmo diseñado y optimizado para entregar soluciones que se adecuen a los requerimientos su negocio.

CAPÍTULO 3: GENERACIÓN DE LA POBLACIÓN INICIAL

1 Estructuras de Datos

A continuación se presentarán las principales estructuras de datos a utilizar para el desarrollo del algoritmo que busca resolver la asignación de flotas en aerolíneas. La complejidad del problema recae en mantener el principio de conservación de flujo y maximizar el beneficio obtenido, en conjunto, por las asignaciones por lo que las estructuras a utilizar facilitarán estas tareas.

Entre las estructuras de datos que se van a presentar podemos distinguir dos grupos, el primero hace referencia a la principal estructura y la base de todo algoritmo genético, el *cromosoma*. Por otro lado encontramos las demás estructuras, conocidas como auxiliares, las cuales permitirán soportar las operaciones y cálculos que se tendrán que hacer sobre la estructura principal, así como simular un ámbito donde se pueda probar el algoritmo.

Es necesario destacar que estas son las estructuras principales para el desarrollo de ambos algoritmos, GRASP y genético. Sin embargo, dentro de cada algoritmo existen estructuras adicionales las que serán presentadas con su seudocódigo.

1.1 Cromosoma

1.1.1 Definición de la Estructura

Como se ha explicado en capítulos anteriores, un individuo representa una posible solución al problema planteado. Este individuo a su vez está conformado por unidades más pequeñas conocidas como *genes.*

Para el problema planteado se optó por usar la siguiente estructura: cada gen representa el par “etapa de vuelo - tipo de flota”, donde la etapa vuelo contendrá la distancia recorrida en el mismo, el aeropuerto de inicio con su respectiva hora de salida, la estación de llegada con su hora de arribo y el número de vuelo; el tipo de flota contendrá el código del tipo de flota, el nombre del tipo de flota, la cantidad de asientos disponibles, la velocidad de crucero y el consumo de combustible por hora de viaje. (Imagen 13).

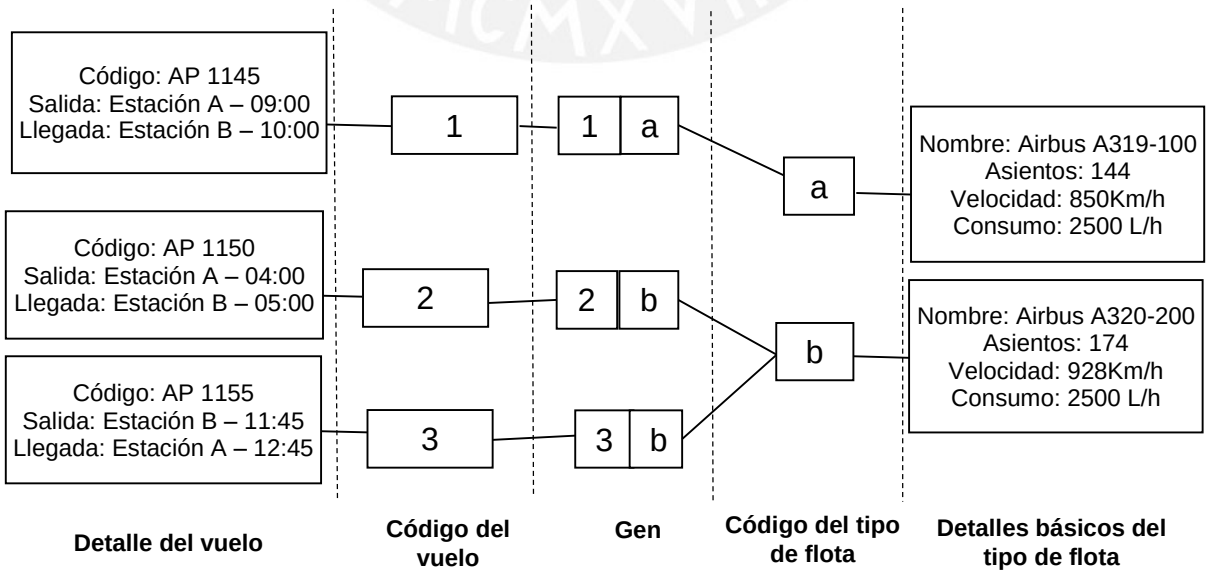


Imagen 13: Ejemplo de la formación de un gen. Imagen de autoría propia.

Como se muestra en la imagen 13 el gen contendrá los codigos de la etapa de vuelo y del tipo de flota, por ende un cromosoma se verá de la siguiente manera:

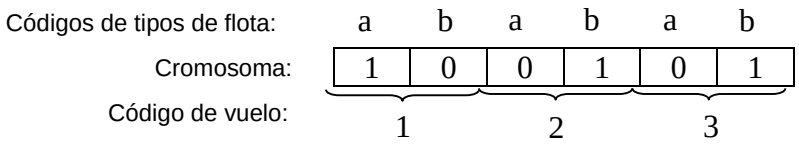


Imagen 14: Ejemplo de cromosoma. Imagen de autoría propia.

La imagen 14 muestra un conjunto de *genes* que conforman un cromosoma. En este caso cada gen ocupa dos bits, debido a que el ejemplo presentado toma en cuenta la existencia de solo dos tipos de aviones, pero el tamaño del gen estará determinado por la cantidad de tipo de aviones que se considerarán en el problema, por ejemplo si se han de considerar 5 tipos de aviones cada gen tendrá un tamaño de 5 bits. Este cromosoma representa las asignaciones "etapa de vuelo – tipo de flota" para la lista de vuelos que se ingresa como dato de entrada. Por ejemplo, para el vuelo 2 se deberá observar el segundo par de bits, en el cual se observa que este par el segundo bit es el que tiene valor "1", lo que idica que a este vuelo se le ha asignado el tipo de avion con codigo *b.* A nivel de estructuras de datos, este cromosoma será representado a travéz de una lista de *bits*. A continuacion un ejemplo mas complejo de un cromosoma.

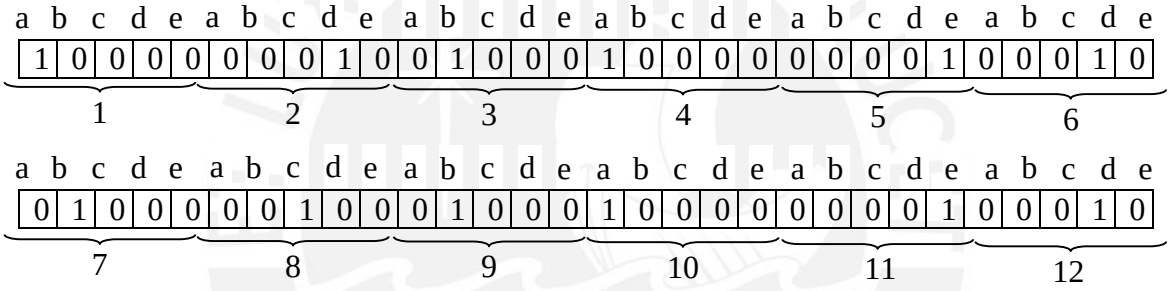


Imagen 15: Ejemplo de cromosoma. Imagen de autoría propia.

En la imagen 15 se muestra un cromosoma para doce vuelos tomando en cuenta cinco tipo de aviones, en este caso cada gen ocupa cinco bits, donde el bit que esta en "1" indica que tipo de avion ha sido asignado al vuelo. En este ejemplo tenemos que al vuelo *1* se le ha asignado el tipo de avion *a*, mientras que al vuelo *2* se le asignó el tipo de avion *d*.

Para comprender como funciona la estructura cromosomática mencionada en el ejemplo anterior, se procederá a contextualizar el cromosoma presentado en el ejemplo anterior. Se deben asignar tipos de flota a cada uno de los vuelos de la aerolinea, para ello se debe calcular cual es el consumo de combustible de cada modelo de aeronave para cada vuelo. Luego se procede a calcular la posible ganancia en la asignación de un modelo específico de aeronave a cada vuelo multiplicando los asientos del avion por un factor de demanda asignado a cada vuelo. Finalmente se divide la posible ganancia entre el consumo de combustible y se selecciona al tipo de flota que posea el mayor número resultante de la división. Esta selección debe estar restringida por los factores limitantes mencionandos en capítulos anteriores.

Durante la creación del cromosoma se agruparán los vuelos que sean atendidos por la misma aeronaves. Por cada avion se registrará una "ruta de vuelo", la cual será representada por una lista de los codigos de los vuelos que son atendidos por la aeronave, ordenados cronologicamente tomando en cuenta la hora de salida de cada vuelo. El orden de los genes en el cromosoma estará dado por la aeronave que atiende al vuelo repesentado en el gen y luego por la hora de salida del vuelo. Esta agrupación

facilitará la detección de abominaciones durante los procesos de casamiento y mutación usados en el algoritmo genético.

1.1.2 Representación

El cromosoma, que lo llamaremos individuo, será representado de la siguiente manera.

Lista<Gen> individuo

Donde *individuo* hace referencia a una lista de objetos del tipo *Gen* que conforman el cromosoma y que a nivel de código también tendrá ese mismo nombre. La lista de objetos tipo Gen tendrá como tamaño el número de vuelos a los cuales se les asignará un tipo de avión y cada gen contendrá un código de vuelo y un código de tipo de flota.

1.2 Lista de vuelos

1.2.1 Definición de la Estructura

Esta estructura de datos auxiliar permitirá almacenar los detalles de cada vuelo. Para ello se definirá una clase *vuelo* que guardará todos los detalles útiles de una etapa de vuelo, los cuales son:

- Código de vuelo
- Aeropuerto de partida
- Hora de partida
- Aeropuerto de llegada
- Hora de llegada
- Distancia del vuelo
- Factor demanda

Este último detalle es un valor numérico que representa si el vuelo es de alta, mediana o baja demanda y nos permitirá priorizar la atención de vuelos más rentables. Los demás datos servirán para restringir la elección del tipo de flota de acuerdo a la disponibilidad del mismo a la hora de salida en el aeropuerto de partida y el consumo de combustible dependiendo de la distancia a recorrer.

1.2.2 Representación

Para la representación de esta estructura se utilizará la siguiente clase, que será cada elemento de la lista.

```
Clase vuelo {
        codVuelo
        hPartida
        hLlegada
        aPartida
        aLlegada
        distancia
        factorDemanda
}
```

La lista de vuelos será representada de la siguiente manera:

Lista<vuelo> lVuelos

Donde *lVuelos* hace referencia a una lista de objetos de tipo *vuelo* que conforman la ruta que representa determinado cromosoma.

1.3 Lista de tipos de flota

1.3.1 Definición de la Estructura

En esta estructura se guardarán los datos necesarios de los tipos de flota. Para ello se define una clase tipo de flota denominada *tflota* la cual contendrá los siguientes datos:

- Código de tipo de flota
- Nombre de tipo de flota
- Cantidad de asientos
- Consumo de combustible por unidad de distancia
- Velocidad de crucero
- Número de aeronaves disponibles

1.3.2 Representación

Para la representación de esta estructura se utilizará la siguiente clase, que será cada elemento de la lista.

Clase tFlota {
codTFlota
nomTFlota
cantAsientos
consComb
velCrucero
NumDisp
}

La lista de tipos de flota será representada de la siguiente manera:

Lista<tFlota> lTFlota

Donde *lTFlota* hace referencia a una lista de objetos de tipo *vuelo* que conforman la ruta que representa determinado cromosoma.

1.4 Ruta de vuelo

1.4.1 Definición de la estructura

En esta estructura se almacenará la secuencia de vuelos que ha de seguir una aeronave. Su principal uso será en los operadores de casamiento y mutación a fin de evaluar si la ejecución de operador generará una o más abominaciones.

1.4.2 Representación

Clase ruta {
Avión
Lista<codigoVuelos> secuencia
}

Donde “Avión” contendrá los datos básicos de la aeronave, como son su código y su modelo, y secuencia tendrá la lista de los vuelos a atender por la aeronave en orden cronológico.

1.5 Red tiempo-espacio

1.5.1 Definición de la Estructura

Esta estructura permitirá representar la secuencialidad de los vuelos y controlar la restricción de conexión de flujo. Se representará por medio de un grafo dirigido con capacidades en los arcos. Cada nodo del grafo representará el punto de partida o punto de llegada de una etapa de vuelo, unidos por arcos con capacidad uno. Cada par de nodos *(partida i, llegada i)* unidos por un arco representara una etapa de vuelo. Además, para representar el carácter cíclico de la asignación, cada nodo que llegue o parta de un aeropuerto estará unido al siguiente nodo presente en ese mismo aeropuerto, en orden cronológico, por una conexión, la cual tendrá una capacidad infinita (imagen 16).

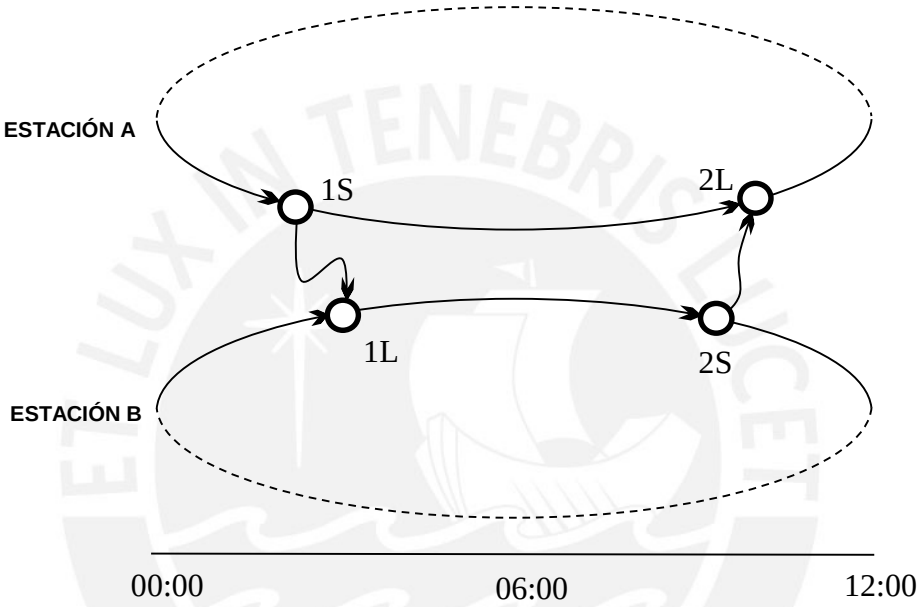


Imagen 16: Ejemplo de una red tiempo-espacio. Imagen de autoría propia.

En la imagen 16 se pueden observar 4 nodos, el par *(1S, 1L)* simboliza al vuelo con código *1* mientras que el par *(2S, 2L)* representa al vuelo con código *2*. La línea punteada simboliza las horas P.M. mientras que la línea continua lo hace con las horas A.M., de esta manera se da a conocer el aspecto cíclico de la asignación. El principal uso de esta estructura será para el control de flujo, verificar que si se quiere asignar un tipo de avión a un vuelo, este debe estar disponible en la estación de salida. Por ejemplo, si se asume que al vuelo 1 se le asigna el tipo de avión *a*, se necesita que exista al menos un avión de este tipo disponible a la hora de salida del vuelo en la estación A. Para que esta condición sea cumplida se requerirá que exista un avión del tipo *a* sin asignar o que el día anterior haya llegado a la estación A al menos un avión del tipo *a*. Siendo la primera asignación de la red de tiempo, se asignará un avión del conjunto de aeronaves no asignadas, es decir de la lista de tipos de flota se sustraerá 1 unidad al tipo de avión *a*. Luego se procede a hacer una asignación al vuelo *2*, debido a que existe una conexión entre el nodo *1L* y *2S* la aeronave del tipo *a* que llegó del vuelo *1* está disponible para atender al vuelo 2, también se puede seleccionar una aeronave no asignada. Teniendo en cuenta que para que el vuelo 1 se realice debe existir una aeronave del tipo *a* en la estación A, tomando en cuenta también que existe una conexión entre *2L* y *1L*, lo que permite que la aeronave que atienda al vuelo *2* pueda luego atender al vuelo *1*, se dará prioridad a la elección de este tipo de aeronave. Asumiendo que en la estación *B* ya existe una aeronave del tipo *a* disponible no es necesario seleccionar un avión no

asignado, entonces se procede a asignar a la aeronave que llegó del vuelo 1 a atender al vuelo *2*. Al final de este vuelo la aeronave se encontrará en la estación A lista para atender al vuelo 1 en el siguiente día.

En la imagen 17 se muestra cómo quedará la red de tiempo luego de la asignación descrita en el párrafo anterior. El tipo de avión a ha sido asignado a los arcos (1S, 1L), (1L, 1S), (2S, 2L) y (2L, 1S).

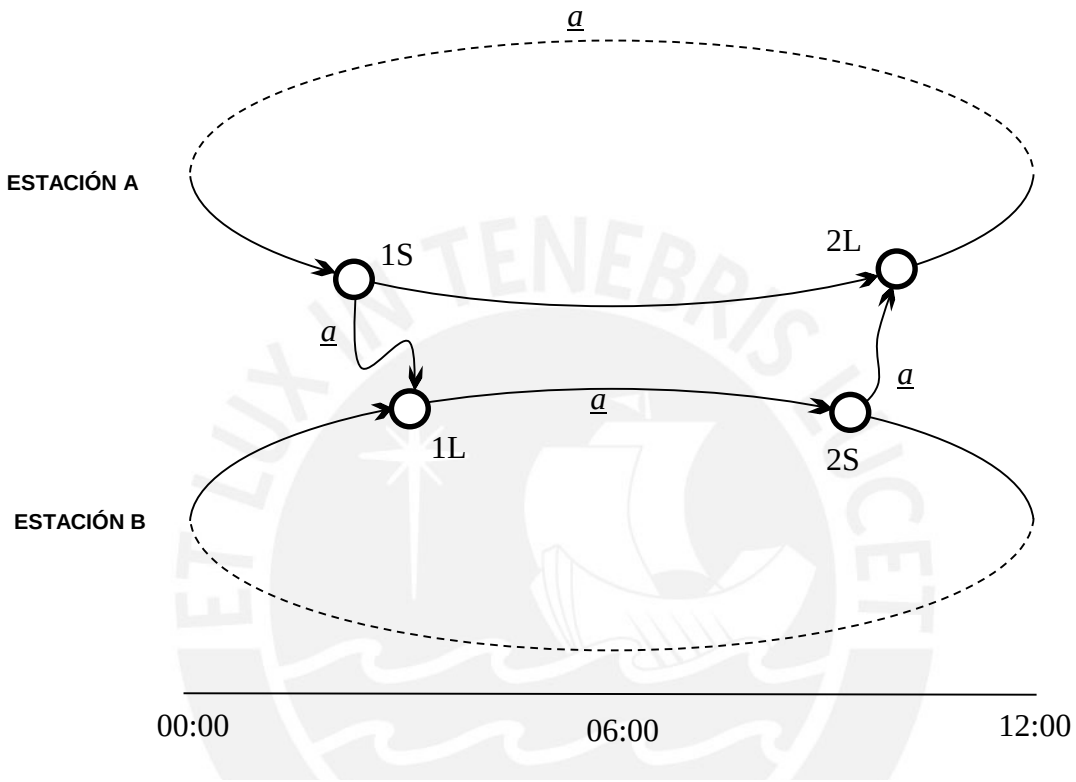


Imagen 17: Ejemplo de una red tiempo-espacio. Imagen de autoría propia.

1.5.2 Representación

Para representar esta estructura en código se usará la siguiente clase:

Lista< Conexión > redTiempoEspacio

Donde la clase Conexiones es:

```
Clase Conexión {
        EtapaVuelo
        Lista<Arco> conexiones
}
```

Donde *EtapaVuelo* es el nodo del cual se listarán las conexiones y *Lista<Arco>* contiene a los nodos con los cuales se tiene alguna conexión y aviones asignados a este.

1.6 Población

1.6.1 Definición de la Estructura

El algoritmo genético utiliza varios cromosomas como posibles soluciones para cada iteración que da, por ello se maneja un conjunto de cromosomas conocido como *población*. Esta *población* va evolucionando en cada iteración con el objetivo mejorar entre cada una de ellas.

La *población* no es más que una lista de individuos que se va actualizando según los operadores a utilizar durante la ejecución del algoritmo.

1.6.2 Representación

A nivel de código la representación de la población es la siguiente.

Lista<individuo> población

Donde *individuo* es la estructura de datos definida anteriormente, y *población* hace referencia a la población que esta representa.

1.7 Fitness de la Población

1.7.1 Definición de la Estructura

Para poder determinar qué individuo dentro de la población es el mejor, o tiene un mejor grado de bondad que otro, debemos evaluar la función fitness en cada uno de estos individuos y almacenarlo en alguna estructura para su posterior uso.

Esta estructura utilizará una lista de números reales que representarán el fitness obtenido por cada individuo de la población. Esto va a ir variando según la población va evolucionando. Su función es básicamente poder utilizar el valor del fitness para hacer la selección de individuos que pasarán a la siguiente población.

1.7.2 Representación

A nivel de código la representación del fitness de cada individuo de la población es la siguiente.

Lista<numReales> fitPob

Donde *numReales* hace referencia a un tipo de datos de números reales, ya que el *fitness* de cada individuo no necesariamente es un número entero. Asimismo *fitPob* hace referencia al *fitness* de la población, almacenando en la misma variable (lista) pero en elementos distintos el *fitness* de cada individuo.

1.8 Red tiempo-espacio de la Población

1.8.1 Definición de la Estructura

Para poder obtener la representación del individuo cuya población es mejor y mantener un registro de las soluciones alternas se almacenarán las representaciones en forma de red tiempo-espacio de la última población activa.

Esta estructura utilizará una lista de redes de tiempo-espacio la cual irá variando según evolucione la población.

1.8.2 Representación

A nivel de código la representación de la red tiempo-espacio de cada individuo es la siguiente:

Lista< redTiempoEspacio > redPoblacion

Donde *redTiempoEspacio* hace referencia a la representación en red de tiempo-espacio para un individuo. La clase *redPoblacion* hace referencia a la red tiempo-espacio de la población, almacenando en la misma variable, en forma de lista, pero en elementos distintos la representación en red tiempo-espacio de cada individuo.

2 Función Fitness

2.1 Definición

En el presente apartado se procederá a presentar la función fitness u objetivo que busca determinar el grado de bondad de una solución, de manera que se pueda discernir qué solución es mejor que otra.

A continuación se muestra dicha función objetivo:

$$FO = Max\left(\sum_{i=0}^{n}\sum_{j=0}^{m} X_{ij} * \left(\frac{CDP_i * CAA_j}{CC_j * \frac{D_i}{VC_j}}\right)\right)$$

2.2 Detalle de variable

A continuación se detallará cada variable de la función objetivo.

X_{ij} : Indica si el tipo de avión *j* ha sido asignado al vuelo i. En caso se haya producido la asignación tomará el valor 1, caso contrario tomará el valor 0.

CDP_i : Es un coeficiente de demanda para el vuelo i y tomará 3 valores distintos: 10 en caso el vuelo posea una alta demanda, 5 en caso el vuelo tenga una demanda regular y 2 en caso el vuelo tenga una baja demanda.

CAA_j : Es la cantidad de asientos que posee una aeronave del tipo *j*. Este es un número entero.

CC_j : Consumo de combustible por hora de vuelo del avión *j*.

D_i : Distancia entre el punto de origen y el punto de destino del vuelo *i*.

VC_j : Representa la velocidad de crucero de la aeronave *j*.

La operación $(CDP_i * CAA_j)$ representa el ingreso a obtener en caso se asigne el avión *j* al vuelo *i*. La operación $(\frac{D_i}{VC_j})$ tendrá como resultado el tiempo requerido para atender el vuelo por el avión seleccionado, el cual será multiplicado CC_j con lo cual se obtiene el gasto de combustible. Se ha hecho uso del coeficiente de demanda de vuelo para

maximizar la cantidad de pasajeros que pueden ser transportados en vuelos de alta demanda y se ha seleccionado como gasto principal el consumo de combustible debido a que este es uno de los principales componentes de los costos operativos de una aerolínea.

El valor de la variable X_{ij} esta directamente ligado al cromosoma, debido a que en esta estructura se almacena que tipo de aeronave ha sido asignado a cada vuelo. Entonces la variable X_{ij} tendrá valor 1 si en el gen correspondiente al vuelo *i* el bit *j* esta prendido, caso contrario tendrá valor 0.

En el presente caso, la solución que tenga el mayor valor de la función objetivo, será más óptima y por lo tanto más elegible a ser la solución final. Por esto se buscará maximizar la función objetivo.

CAPÍTULO 4: MODELO COMPUTACIONAL GRASP

1 Algoritmo GRASP

En el presente apartado se presentará el algoritmo GRASP a desarrollar, tanto como para el input del algoritmo genético como para la comparación de ambos a través de la experimentación numérica. Este algoritmo ha sido seleccionado tomando en cuenta dos principales razones:

1. El algoritmo ya ha sido adaptado al problema en conjunto con el algoritmo de recocido simulado. En esta adaptación el algoritmo GRASP logró obtener resultados comparables al del algoritmo de recocido simulado e incluso los superó en algunos casos [SHERALI, BISH, ZHU, 2006]. Por esto se puede concluir que el algoritmo es capaz de obtener soluciones buenas al problema.

2. La complejidad no muy alta en la adaptación del algoritmo GRASP hace posible su uso como algoritmo de comparación dentro de los límites de tiempo del proyecto.

1.1 Variables y estructuras

Antes de mostrar el seudocódigo del algoritmo GRASP se presentarán las variables y estructuras propias de este algoritmo. Ya que si bien se utilizarán como base las estructuras presentadas en el capítulo anterior, existen variables que son propias del este algoritmo y serán presentadas en la tabla a continuación.

Tabla 3: Variables y estructuras del algoritmo GRASP

Representación	Descripción
CS	Hace referencia al conjunto de soluciones que va a brindar el algoritmo GRASP. Al utilizarse como input para el algoritmo genético, no solo es necesario obtener una solución, sino un conjunto de ellas.
MI	Esta variable hace referencia a la cantidad de iteraciones que se ejecutará el algoritmo GRASP.
S	Almacena una solución obtenida por el algoritmo. Esta estructura tomara la forma de un cromosoma, siendo una lista de pares “códigos de vuelo-código de tipo de flota”. Además contendrá una lista de rutas de vuelo para que serán utilizadas en el algoritmo genético.
CA	Representa los tipos de flota que pueden atender un vuelo. Es una lista conteniendo los posibles candidatos que conformarán el RCL, donde cada tipo de flota aparecerá tantas veces como aeronaves de este queden disponibles.
RCL	Es una versión restringida de CA utilizando el valor de alfa, el mejor elemento y el peor.
V	Representa al vuelo al cual se le va a asignar un tipo de flota. Contiene los datos necesarios del vuelo para obtener el CA.

Representación	Descripción
A	Representa un tipo de flota elegido aleatoriamente del RCL para ser asignado a un vuelo y que formará parte de la solución de determinada iteración del algoritmo.
CAA	Representa la lista de aviones, disponibles en la estación de salida, que ha sido filtrada por las restricciones disponibilidad respecto al tiempo de salida del vuelo.
LTF	Representa la lista de aviones disponibles en la aerolínea.
LTFR	Representa la lista de aviones no asignados disponibles.
MXB	Representa el valor máximo de bondad en una lista de candidatos para ser asignados a un vuelo
MNB	Representa el valor mínimo de bondad en una lista de candidatos para ser asignados a un vuelo.
RTS	Representa la red de tiempo-espacio que irá variando entre cada iteración.

1.2 Seudocódigo del algoritmo GRASP

```
Algoritmo_GRASP (LTF, RTS)
Inicio
1.      CS ← {0}
2.      Inicializar(RTS)
3.      Mientras (num_iteraciones < MI) hacer
        Inicio
3.1.        V ← obtener _siguiente_vuelo (RTS)
3.2.        Mientras (solución <> completa) hacer
            Inicio
3.2.1.          CA = buscar_candidatos(LTF, RTS,V)
3.2.2.          RCL <- generar_RCL(CA,alfa)
3.2.3.          A <- selección_aleatoria(RCL)
3.2.4.          S <- S U {A}
3.2.5.          S <- actualizar_ruta(A)
            Fin
3.3.        HallarBondad(S)
3.4.        Actualizar_solucion(CS, S)
3.5.
        Fin
Fin
```

Imagen 18: Seudocódigo del algoritmo GRASP. Imagen de autoría propia.

1.3 Explicación del algoritmo GRASP

El algoritmo GRASP presenta como parámetros las siguientes variables: lista de tipos de flota (LTF), red de tiempo-espacio (RTS), las cuales ya han sido definidas en el capítulo anterior.

A continuación se explicaran una por una las líneas presentadas en el seudocódigo del algoritmo GRASP.

Línea 1: Debido a que el algoritmo genético recibe como población inicial el resultado del algoritmo GRASP, se debe adaptar la estructura obtenida por el GRASP a la estructura cromosomática que utiliza el algoritmo genético. Por ello se utiliza un conjunto de soluciones como resultado, el cual será guardado en la variable *CS,* y que luego, dentro del algoritmo genético, se convertirá en cromosomas.

Línea 2: Se inicializa la red tiempo-espacio, asignando un valor vacío a cada arco de la red, simbolizando que aún no se ha asignado ningún tipo de flota a los vuelos.

Línea 3: El algoritmo GRASP será ejecutado varias veces de manera que se obtenga un buen conjunto de soluciones. En cada iteración se genera y evalúa una solución, si esta es mejor que alguna de las anteriores se almacena. La variable *MI* representa el número máximo de iteraciones.

Línea 3.1: Se selecciona un vuelo, al cual se le asignará un tipo de flota, se priorizarán los vuelos por la hora de salida y en caso existan varios vuelos que tengan la misma hora de salida se elegirá uno de manera aleatoria.

Línea 3.2: Se entra en una iteración que genera una asignación de vuelo a tipo de flota hasta que todos los vuelos tengan asignado un tipo de flota o no existan aeronaves disponibles.

Línea 3.2.1. Se hallan los posibles tipos de flota que pueden atender un vuelo, tomando en cuenta restricciones como la disponibilidad de la aeronave a la hora de salida de un vuelo, la distancia a recorrer, el consumo de combustible y cantidad de aeronaves de cada tipo de flota. Como resultado se tendrá una lista que contiene los códigos de los tipos de flota, que aparecerán tantas veces como aeronaves disponibles del tipo de flota existan. Para obtener la lista de tipos de flota disponible se hará uso de la red tiempo-espacio (RTS) y de la lista de tipos de flota (LTF). Tomando en cuenta los datos de los vuelos, específicamente el aeropuerto de salida se verificará con el RTS que aeronaves están disponible en esa locación. En caso no exista ninguna nave disponible se procederá a verificar LTF donde estarán las aeronaves que no han sido asignadas a ningún vuelo. Cada una de las verificaciones tomará en cuenta las limitaciones propias del problema para la asignación de tipos de flota a vuelos.

Línea 3.2.2: Se genera la *RCL* que un subconjunto de la lista *CA*, la cual fue limitada por el mejor valor (a), peor valor (b) y el valor del alpha (α). El RCL se obtiene seleccionando los candidatos que cuya bondad mayor a $(a - \alpha \times (a - b))$.

Línea 3.2.3: Se elige al azar un tipo de flota dentro del *RCL* y se almacena, junto al código del vuelo, en la variable *A*.
Línea 3.2.4: Se une la asignación “tipo de flota – vuelo” a la solución que esta guardada en la variable S.

Línea 3.2.5: En la función actualizar ruta se evalúa si el tipo de flota seleccionado era un avión no asignado o uno disponible en la estación de salida. En el primer caso se le

genera un nuevo código de avión, así como una nueva ruta de vuelo la cual será asignada al avión recién designado y se agrega a esta ruta el código de vuelo actual. En el segundo caso se busca la ruta de vuelo del avión seleccionado y se agrega el código del vuelo actual.

Línea 3.3: Una vez obtenida una asignación para cada vuelo, se considerará que se tiene una solución completa. Para esta solución se halla su valor de bondad usando la función de bondad definida en el apartado anterior. Se hace uso de la misma función con el objetivo de poder comparar las soluciones presentadas por el algoritmo GRASP con las que generará el algoritmo genético.

Línea 3.4: Usando el valor de bondad encontrado, se verifica si la solución es mejor que alguno de los contenidos en *CS*, en caso sea mejor que alguna se agrega a *CS* y se descarta la peor solución del conjunto.

1.4 Seudocódigo de la función buscar candidatos

```
buscar_candidatos(LTF, RTS,V)
Inicio
1.      CAA ← ObtenerAeronavesDisponibles(RTS,V)
2.      LTFR ← FiltrarAviones(LTF,V)
3.      CAA ← Actualizar(CAA, LTFR)
4.      Mientras (CAA <> vacío) hacer
        Inicio
4.1.        C ← obtener _siguiente_candidato (CAA)
4.2.        C ← CalcularBondad(C,V)
4.3.        CA ← CA U {C}
4.4.        Remover_candidato(CA,C)
        Fin
4.5.    Ordenar (CA)
4.6.    Retornar(CA)
Fin
```

Imagen 19: Seudocódigo de la función buscar candidatos. Imagen de autoría propia.

1.5 Explicación de la función buscar candidatos

Esta función se encarga de generar la lista de candidatos filtrando los aviones disponibles en la estación de salida y los aviones que no han sido asignados.

Línea 1: Se recorren todas las conexiones que se dirijan al nodo de salida del vuelo. Para cada conexión se extraen las aeronaves disponibles. Para cada aeronave en esta lista se verifica si cumple con la limitación de tiempo y distancia. Las aeronaves que pasan el filtro son asignadas a la lista de candidatos auxiliar (*CAA*).

Línea 2: Se recorre la lista de tipos de aviones (*LTF*) obteniendo estos con una disponibilidad mayor 0 y se verifica si cumplen con las restricciones de distancia. Los tipos de aviones que pasan el filtro son asignados a la lista de tipos de vuelos restringida (*LTFR*).

Línea 3: Se una la lista de candidatos auxiliares con la lista de tipos de vuelo restringida

Línea 4: Se recorre cada elemento de la lista *CAA.*

Línea 4.1: Se obtiene el siguiente candidato en la lista *CAA*, asumiendo que esta lista esta desordenada.

Línea 4.2: Para cada candidato se calcula la bondad, la cual es obtenida dividiendo la posible ganancia, representada por la multiplicación de la cantidad de asientos del avión con el factor demanda del vuelo, entre los costos de operación, representado por la multiplicación del consumo de combustible del avión con la distancia de vuelo. La fórmula es la siguiente:

$$bondad = \frac{cantidad\ asientos\ del\ avion * factor\ demanda\ del\ vuelo}{consumo\ de\ combustible\ del\ avion * distancia\ de\ vuelo}.$$

Esta bondad es luego guardada como parte de la información del candidato.

Línea 4.3: Se agrega el candidato a la lista de candidatos (*CA*).

Línea 4.4: Se retira el candidato de la lista de candidatos auxiliar (*CAA*).

Línea 4.4: Al finalizar la iteración se ordena la lista de candidatos, de acuerdo al valor de bondad de manera descendente.

Línea 4.5: Se retorna la lista de candidatos que está en la variable *CA*.

1.6 Seudocódigo de la función genera RCL

```
generar_RCL(CA, alfa)
Inicio
1.      MXB ← max_bondad(CA)
2.      MNB ← min_bondad(CA)
3.      LI ← MXB- alfa×(MXB-MNB)
4.      Mientras (CA <> vacío) hacer
        Inicio
4.1.        C ← obtener _siguiente_candidato (CA)
4.2.        Si (bondad(C) > LI) hacer
            Inicio
4.2.1.          RCL←RCL U {C}
            Fin
4.3.        Remover_candidato(CA,C)
        Fin
4.4.        Retornar(RCL)
Fin
```

Imagen 20: Seudocódigo de la función generar RCL. Imagen de autoría propia.

1.7 Explicación de la función general RCL

En esta función se procede a genera el RCL para su posterior uso en la selección de tipo de avión asignado al vuelo.

Línea 1: Se obtiene el mayor valor de bondad en la lista de candidatos y se asigna el valor a la variable *MXB*.

Línea 2: Se obtiene el menor valor de bondad en la lista de candidatos y se asigna el valor a la variable *MNB*.

Línea 3: Se obtiene el límite inferior para los valores que pueden pertenecer al RCL y se asigna a la variable LI. Se usa la siguiente formula: *mayor valor bondad – alfa × (mayor valor bondad – menor valor bondad).*

Línea 4: Se recorre toda la lista de candidatos (CA).

Línea 4.1: Se obtiene el siguiente candidato de la lista para ser evaluado. Se asume que la lista esta desordenada.

Línea 4.2: Si el candidato posee un valor de bondad mayor al límite inferior (LI) es agregado al RCL.

Línea 4.2.1: Se agrega al candidato al RCL.

Línea 4.3: Se remueve al candidato evaluado de la lista CA para asegurarnos que no vuelva a ser evaluado.

Línea 4.3: Se retorna el RCL con los candidatos cuya bondad es mayor al límite inferior (LI).

1.8 Calibración del alfa

A continuación se procede a la calibración del valor del alfa que tendrá el algoritmo GRASP presentado.

Se realizó un primer experimento haciendo variar el valor de alfa desde 0.3 hasta 0.75 en un intervalo de 0.05. A continuación se muestra la tabla resumen de estos resultados, sin embargo en el anexo 1 se puede observar el detalle de esta prueba. Se muestran los valores de bondad para cada uno de los valores de alfa de un muestreo de 40 pruebas.

Tabla 4: Calibración del alfa – nivel 1

	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75
Promedio	7.08	7.33	7.60	7.61	8.52	8.53	8.61	8.61	8.72	8.75
Mínimo	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35
Frec. Mínima	1	1	1	1	1	1	1	1	1	1
Máximo	14.68	14.68	19.10	19.11	19.11	19.13	19.14	19.14	19.14	19.16
Frec .Máximo	3	3	1	1	1	1	1	1	1	1
Desviación	5.2826	5.2009	5.2855	5.2738	5.7363	5.743	5.7241	5.7243	5.7464	5.766

Durante la ejecución de las pruebas se notó que para valores de alfa que van desde 0.30 hasta 0.50 se presentaban casos en los que no se generaban soluciones viables. Los valores mayores a 0.75 presentaban mucha diferencia entre sus soluciones y

aunque algunas de estas eran mayores a las que obtuvieron con un menor valor de alfa también se encontraban casos donde no se llegaba a una solución viable. Como se buscar maximizar el valor de la función de bondad, se puede notar que con los valores 0.65, 0.7 y 0.75 de alfa se obtienen en promedio los valores más óptimos. Por ello se decide realizar otra prueba, esta vez entre los rangos de 0.65 y 0.75 con un intervalos de 0.1 y, al igual que el caso anterior, para 40 pruebas. Para mayor detalle revisar el anexo 2.

Tabla 5: Calibración del alfa - nivel 2

	0.65	0.66	0.67	0.68	0.69	0.70	0.71	0.72	0.73	0.74	0.75
Promedio	8.62	8.62	8.67	8.72	8.72	8.72	8.72	8.73	8.73	8.75	8.75
Mínimo	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35
Frec. Mínima	1	1	1	1	1	1	1	1	1	1	1
Máximo	19.16	19.16	19.16	19.16	19.16	19.16	19.16	19.16	19.16	19.16	19.16
Frec .Máximo	1	1	1	1	1	1	1	1	1	1	1
Desviación	5.7295	5.7295	5.7575	5.7526	5.7518	5.7515	5.7513	5.7548	5.7555	5.7692	5.771

Como se puede observar los valores alfa que nos dan el mayor promedio son 0.74 y 0.75. Ambos valores nos entregan en promedio el mismo valor de bondad, pero 0.74 muestra una desviación estándar menor, por lo que las soluciones generadas con este alfa estarán más cercas al promedio mostrado que las soluciones generadas con un valor alfa de 0.75

Ante estas razones se decide usar el valor alfa de 0.74, por lo que se concluye en la calibración.

1.9 Calibración de iteraciones

A continuación se procede a realizar la calibración de la cantidad de iteraciones que tendrá el algoritmo GRASP. Para esto se procede a ejecutar una variedad de pruebas, camiando el número de iteraciones desde 1000 hasta 15000 con un incremento de 500. Para cada valor que toma el un número de iteraciones se ejecutarán 40 pruebas, las cuales harán uso del valor alfa calibrado anteriormente, que es 0.74. Durante las pruebas se observará el comportamiento de la función bondad promedio, el cual se busca maximizar, a la par con el tiempo de ejecución, buscando un balance entre un valor lo suficientemente optimo y un tiempo de ejecución bajo.

A continuación se muestra la tabla resumen y un gráfico representando los resultados. Para mayor detalle revisar el anexo 3.

Tabla 6: Calibración del Número de Iteraciones

Bondad Promedio	Iteraciones	Tiempo Promedio	Bondad por unidad de tiempo
8.20	1000	198.63	0.0413
8.71	1500	267.93	0.0325
8.73	2000	325.33	0.0268
8.73	2500	329.53	0.0265
8.73	3000	407.75	0.0214
8.74	3500	453.43	0.0193

Bondad Promedio	Iteraciones	Tiempo Promedio	Bondad por unidad de tiempo
8.74	4000	523.25	0.0167
8.74	4500	576.63	0.0152
8.75	5000	629.13	0.0139
8.75	5500	699.48	0.0125
8.75	6000	765.98	0.0114
8.75	6500	824.60	0.0106
8.75	7000	883.05	0.0099
8.75	7500	950.43	0.0092
8.75	8000	1021.13	0.0086
8.75	8500	1115.10	0.0078
8.75	9000	1141.00	0.0077
8.75	9500	1202.08	0.0073
8.76	10000	1266.48	0.0069
8.76	10500	1321.60	0.0066

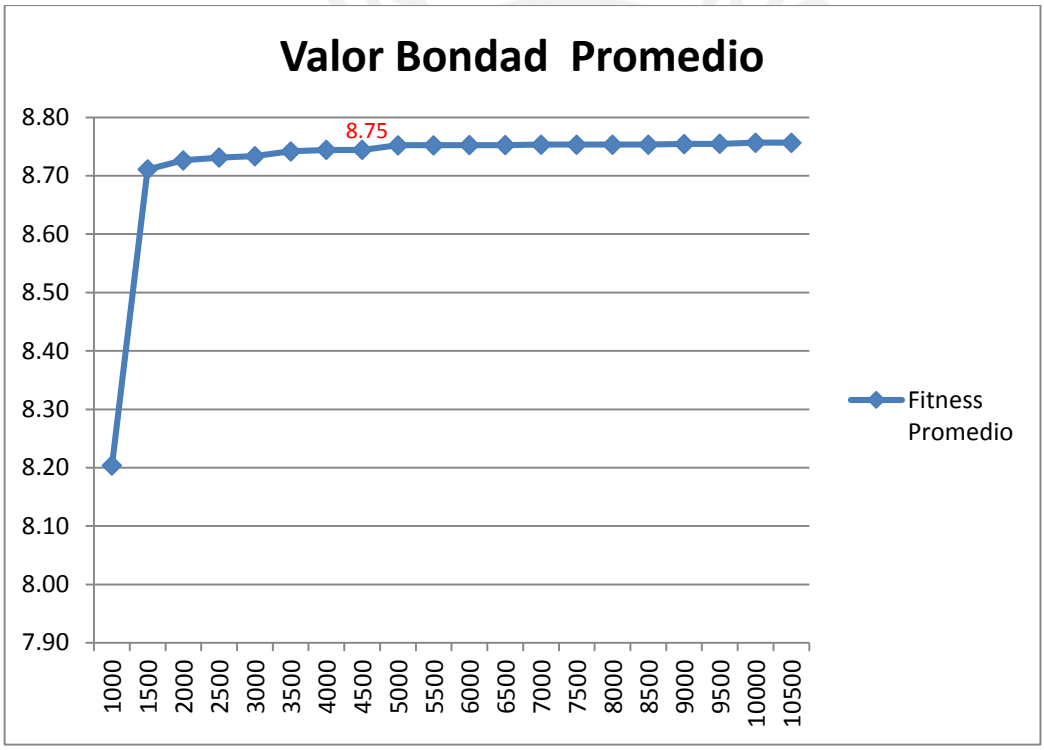


Imagen 21: Representación gráfica de la variación del valor de bondad. Imagen de autoría propia.

Luego de observar los resultados, se concluye que el valor para el número de iteraciones será el de 5000. Esto se debe que permite obtener un alto valor de bondad y brinda una buena relación entre este valor y el tiempo de ejecución requerido. Además como se puede observar tanto en el gráfico y en la tabla, los valores posteriores no mejoran significativamente el valor de bondad, mostrándose una mínima mejora recién a las 10000 iteraciones, donde el tiempo de ejecución es casi es doble.

CAPÍTULO 5: MODELO COMPUTACIONAL GENÉTICO

1 Algoritmo Genético

En el presente apartado se presentarán las variables del algoritmo genético a desarrollar, así como el seudocódigo y explicación del mismo, en conjunto con los operadores a utilizar. El algoritmo genético ha sido seleccionado para ser el algoritmo principal debido a tres importantes ventajas [GEN, CHENG, LIN, 2008]:

1. Adaptabilidad: Los algoritmos genéticos pueden manejar cualquier tipo de función objetivo y cualquier tipo de restricciones.

2. Robustez: El uso de operadores de evolución hace que los algoritmos genéticos sean muy efectivos en búsquedas globales.

3. Flexibilidad: Los algoritmos genéticos proveen de gran flexibilidad para hibridar con algoritmos heurísticos específicos para ciertos ambientes para obtener una implementación eficiente para un problema específico.

Estas tres ventajas permiten que el algoritmo genético entregue una solución buena, más cercana a la solución óptima global.

1.1 Variables y estructuras

Antes de poder visualizar el seudocódigo del algoritmo genético se definirán las variables y estructuras propias del algoritmo. Ya que si bien se utilizan como base principal las estructuras definidas en el capítulo anterior, existen variables que son propias de este algoritmo y serán presentadas en la tabla a continuación.

Tabla 7: Variables y estructuras del algoritmo genético

Representación	Descripción
CS	Hace referencia al conjunto de soluciones que va a brindar el algoritmo GRASP. Al utilizarse como input para el algoritmo genético es necesario tener un conjunto y no una sola solución.
P	Representa a la población.
TP	Esta variable hace referencia al tamaño de la población.
NA	Esta variable hace referencia a la cantidad de aberraciones que existen en la población actual del algoritmo genético.
MA	Esta variable hace referencia a la cantidad de aberraciones máximas permitidas en el algoritmo genético.
CP	Representa un conjunto de poblaciones y sirve para determinar si se tiene un comportamiento del tipo meseta.
NI	Esta variable hace referencia a la cantidad de iteraciones reales que ejecuta el algoritmo genético.
MI	Esta variable hace referencia a la cantidad máxima de iteraciones que podrá ejecutar el algoritmo genético.
TT	Esta variable representa el tiempo usado, medido en segundos, por el algoritmo.
TM	Esta variable indica el tiempo máximo que puede utilizar el algoritmo.
S	Representa a una solución, forma parte del CS que el algoritmo GRASP brinda como resultado.

Representación	Descripción
P_CAS	Esta variable representa el porcentaje de individuos en una población que será sometida al operador de casamiento.
P_MUT	Esta variable representa el porcentaje de individuos en una población que será sometida al operador de mutación.
LC	Representa un subconjunto de la población, este grupo de individuos serán sometidos a los operadores de casamiento y mutación.
Padre	Hace referencia a un cromosoma, el cual tomará el papel de padre, que será sometido al operador de casamiento
Madre	Hace referencia a un cromosoma, el cual tomará el papel de madre, que será sometido al operador de casamiento
PC1,PC2	Estas variables representarán los puntos de corte donde se va a realizar el intercambio entre los genes del par de cromosomas sometidos al operador de casamiento.
HijoA	Representa a un cromosoma resultante de aplicar el operador de casamiento a un Padre y una Madre.
HijoB	Representa a un cromosoma resultante de aplicar el operador de casamiento a un Padre y una Madre.
G	Representa una asignación aleatoria que será elegida para realizar el operador de mutación.
CM	Hace referencia al cromosoma resultante del operador de mutación.

1.2 Seudocódigo principal

```
Algoritmo_Genético (CS)
Inicio
1.      P ← convertir_cromosomas(CS)
2.      TP ← obtener_tamaño(P)
3.      Mientras (NA < MA && !esMeseta(CP) && NI<MI &&TT<TM) hacer
        Inicio
3.1.        CalcularFitness(P)
3.2.        P ← casamiento(P)
3.3.        P ← mutacion(P)
3.4.        P ← seleccionRuleta (P, TP)
3.5.        CP ← actualizarCP(P)
        Fin
4.  S ← mejorElemento(P)
Fin
```

Imagen 22: Seudocódigo principal del algoritmo genético. Imagen de autoría propia.

1.3 Explicación del seudocódigo principal

A continuación se explicarán en detalle el seudocódigo principal.

Línea 1: Al recibir como población inicial el conjunto de solución *CS* obtenido por el algoritmo GRASP se debe adaptar su estructura a la cromosomática que utilizará el

algoritmo genético. Luego de su conversión se almacenará en la variable *P*. El detalle de este proceso será explicado más adelante.

Línea 2: Se obtiene el tamaño de la población original al verificar que las poblaciones posteriores tengan el mismo tamaño.

Línea 3: Se ingresa en la iteración, la cual posee tres condiciones de salida. La primera controla que no exista un exceso de aberraciones. Por ello se tiene la variable *MA,* que indica el número máximo de aberración que puede existir en una población, y la variable NA, que indica el número de aberraciones existente, la cual no debe ser mayor a *MA*. La segunda controla la condición de meseta, esta condición se da cuando durante cierto número de iteraciones las poblaciones no han logrado obtener un mejor rendimiento que la anterior o las mejoras han sido mínimas. Ante esta condición se concluye que las siguientes poblaciones seguirán el mismo comportamiento. La tercera se refiere al número de iteraciones máximas que se ejecutarán en el algoritmo. Este está determinado por la variable *MI*, mientras que la variable *NI* indica el número de iteraciones que han sido ejecutados. Por último se tiene el tiempo máximo que se debe demorar la ejecución, el cual está definido por la variable *TM*, mientras que el tiempo que algoritmo ha consumido se almacena en la variable TT. Esta última limitación se impone debido a que el objetivo principal de un algoritmo genético es entregar un buen resultado en el menor tiempo posible, limitando el tiempo de ejecución podremos controlar la calidad del resultado de acuerdo a las necesidades y limitación de tiempo del usuario.

Línea 3.1: Para cada iteración, debido a que la población se va actualizando, se debe actualizar el valor de fitness que posee cada uno de los cromosomas de la población *P*. Para ello se usará la fórmula de la función fitness definida anteriormente.

Línea 3.2: Esta línea invoca al operador de casamiento, el cual recibe la población P y por cada par de cromosomas que se “casen” se generaran dos nuevos individuos, que en caso no sean aberraciones, se incorporarán a la población. Este operador será explicado más adelante.

Línea 3.3: En esta línea se aplica un segundo operador conocido como mutación. Se elige un pequeño porcentaje de la población *P* y sobre estos cromosomas se procede a aplicar dicho operador. El detalle de este operador será explicado más adelante.

Línea 3.4: Se selecciona un subconjunto de la población *P* cuyo tamaño será determinado por la variable *TP* y se almacena en la variable *P*. Para ello se hace uso de un operador conocido como ruleta, el cual funciona de la siguiente manera. A cada uno de los cromosomas se le da un peso correspondiente a su valor de fitness, de manera que aquel con mejor fitness tenga mayor peso. Luego se procede a “girar” la ruleta un número de veces determinado, en este caso dependiendo de la variable *TP*. De esta forma los cromosomas con mayor peso poseen mayor probabilidad de ser elegidos, sin embargo aún existe un componente aleatorio inducido por el operador ruleta.

Línea 3.5: Acá se procede a almacenar la población P en la variable CP, la cual representa las poblaciones obtenidas en iteraciones anteriores con el fin de utilizarlas para determinar si nos encontramos en una meseta.

Línea 4: Finalmente luego de terminar la iteración principal del algoritmo genético, se procede a seleccionar al mejor elemento dentro de la población final eligiendo el individuo con mayor valor de fitness y se almacena el individuo elegido en la variable *S*.

1.4 Seudocódigo del procedimiento de conversión a cromosomas

```
Convertir_cromosoma (CS)
Inicio
1.  Mientras(CS<>{})
        Inicio
1.1.        S ← obtenerSolucion(CS)
1.2.        C ← convertirSolucion(R)
1.3.        P ← agregarCromosoma(C)
        Fin
2.  Retornar(P)
Fin
```

Imagen 23: Seudocódigo del procedimiento de conversión a cromosomas. Imagen de autoría propia.

1.5 Explicación del procedimiento de conversión a cromosomas

En este apartado se explicará el detalle del procedimiento para convertir a cromosomas.

Línea 1: El procedimiento posee una iteración principal, la cual es controlada por el conjunto de soluciones *CS* obtenidas por el algoritmo GRASP. La iteración recorrerá cada uno de los elementos que pertenecen a la variable *CS*.

Línea 1.1: Se almacenará en la variable *S* un conjunto de asignaciones para la lista de vuelos de la aerolínea, el cual es un elemento de la lista de conjunto de solución *CS*.

Línea 1.2: Utilizando el conjunto de asignaciones *S* obtenida en el paso anterior, se procede a convertirla en la estructura cromosomática y almacenarla en la variable *C*. Para esto se procederá a obtener la asignación de tipo de avión para cada vuelo y se guardará su valor en el gen respectivo. Además se obtienen otros datos importantes para los procesos de mutación y casamientos, así como las rutas de vuelo recorridos por cada aeronave.

Línea 1.3: El cromosoma *C* obtenido en el paso anterior es almacenado en el conjunto de cromosomas conocido como población, el cual es representado por la variable *P*

Línea 2: Una vez ya se ha procesado *CS* y se ha generado la población *P* se retorna esta última como resultado del procedimiento.

1.6 Seudocódigo del operador de casamiento

```
Casamiento (P, P_CAS)
Inicio
1.      LC <- seleccionRuleta(P, P_CAS)
2.      Mientras(LC <> {})
        Inicio
2.1.        Padre = obtenerCromosoma(LC)
2.2.        Madre = obtenerCromosoma(LC)
2.3.        PC1= obtenerPunto(Padre)
2.4.        PC2= obtenerPunto(Padre)
2.5.        HijoA = casar(padre, madre, PC1,PC2)
2.6.        HijoB = casar(madre, padre, PC1,PC2)
2.7.        Si (!esAbominacion(HijoA))
2.7.1.          P = agregarSolucion(P, HijoA)
2.8.        Si (!esAbominacion(HijoB))
2.8.1.          P = agregarSolucion(P, HijoB)
        Fin
3.      Retornar(P)
Fin
```

Imagen 24: Seudocódigo del operador de casamiento. Imagen de autoría propia.

1.7 Explicación del operador de casamiento

A continuación se muestra la explicación para el seudocódigo del operador de casamiento.

Línea 1: Se selecciona un subconjunto de la población *P* determinado por el porcentaje *P_CAS* y se almacena en la variable *LC*. Para ello se hace uso del operador conocido como ruleta. A cada uno de los cromosomas se le da un peso correspondiente a su valor de fitness, de manera que aquel con mejor fitness tenga mayor peso. Luego se procede a "girar" la ruleta un número de veces determinado, en este caso dependiendo de la variable *P_CAS*. De esta manera los cromosomas con mayor peso posee mayor probabilidad de ser elegidos, sin embargo aún existe un componente aleatoria inducido por el operador ruleta.

Línea 2: Se procede a ejecutar una iteración para recorrer todo el subconjunto de la población, *LC*.

Línea 2.1: Se selecciona un cromosoma dentro del conjunto *LC*, de manera aleatoria, que se almacena en la variable *padre.*

Línea 2.2: Se selecciona un cromosoma dentro del conjunto *LC*, de manera aleatoria, que se almacena en la variable *madre.*

Línea 2.3: Se procede a ubicar el primer punto de corte para el casamiento. Este punto es seleccionado de manera aleatoria y el resultado es almacenado en la variable *PC1*.

Línea 2.4: Se procede a ubicar el segundo punto de corte para el casamiento. Este punto es seleccionado de manera aleatoria y el resultado es almacenado en la variable *PC2*. Se eligen dos puntos de corte para obtener un segmento el cual estará delimitado por ambos puntos.

Línea 2.5: Debido a la necesidad de mantener la conservación de flujo se procederá a intercambiar rutas de vuelo enteras. Estas serán identificadas a partir de los genes entre los puntos de corte PC1 y PC2, a partir de los cuales obtendremos los aviones afectados. A partir de estos aviones se obtendrá la lista de vuelos realizados por estos y a toda esta lista se le hará el intercambio, por lo cual toda la secuencia pasará a ser atendida por una aeronave diferente. Una vez realizado el cambio se obtiene un nuevo cromosoma que se almacena en la variable *HijoA.*

Línea 2.6: Se realiza el mismo procedimiento solo que se intercambia el orden de las variables *padre* y *madre* de manera que se genera un nuevo cromosoma y este es almacenado en la variable *HijoB.*

Línea 2.7: Se procede a verificar si el cromosoma *HijoA* es una abominación. Esto quiere decir que se valida que dicha variable realmente es una solución al problema y si el intercambio realizado es posible tomando en cuenta la disponibilidad de las aeronaves asignadas y no asignadas. En caso sea posible se intercambiarán las secuencias de vuelos entre 2 aeronaves para cumplir con el intercambio. En caso no sea posible realizar el intercambio de secuencias se declarará al *HijoA* como abominación.

Línea 2.7.1: En el caso de que el cromosoma *HijoA* no sea una abominación, se procede a agregar al cromosoma *HijoA* a la población *P*.

Línea 2.8: Al igual que 2.7, se verifica si el cromosoma *HijoB* es una abominación.

Línea 2.8.1: En el caso de que el cromosoma *HijoB* no sea una abominación, se procede a agregar al *HijoB* a la población *P*.

Línea 3: Finalmente se retorna la población *P* actualizada.

1.8 Seudocódigo del operador de mutación

```
Mutacion (P)
Inicio
1.      LC <- seleccionarCromosomas(P, P_MUT)
2.      Mientras(LC <> {})
        Inicio
2.1.        C <- obtenerCromosoma(LC)
2.2.        G <- SeleccionarGen(C)
2.3.        G <- Mutar(G)
2.4.        Si (!esAbominacion(C))
2.4.1.              P = agregarSolucion(P, CM)
        Fin
3.      Retornar(P)
Fin
```

Imagen 25: Seudocódigo del operador de mutación. Imagen de autoría propia.

1.9 Explicación o del operador de mutación

A continuación se muestra la explicación del seudocódigo del operador de mutación.

Línea 1: Se selecciona un subconjunto de la población *P* determinado por la variable *P_MUT*, que representa un porcentaje de mutación. Esta selección es de manera aleatoria y el resultado es almacenado en la variable *LC*.

Línea 2: Se procede a ejecutar una iteración para recorrer todo el subconjunto de cromosomas *LP* a ser mutados.

Línea 2.1: Se obtiene un cromosoma que pertenece a la lista *LC* y este es almacenado dentro de la variable *C*.

Línea 2.2: Se procede a seleccionar un gen del cromosoma *C* de manera.

Línea 2.3: Se procede a mutar el gen eligiendo un nuevo tipo de flota, diferente al actual, para que atienda este vuelo. Debido a la necesidad de mantener la conservación de flujo este cambio afectará a toda la ruta de vuelo a la cual pertenezca el vuelo contenido en el gen elegido.

Línea 2.4: Se procede a realizar una validación si el cromosoma *CM* es una abominación. Esto quiere decir que se valida que dicha variable realmente es una solución al problema y si es posible efectuar el intercambio de rutas necesario para atender la mutación realizada.

Línea 2.4.1: En el caso que el cromosoma *CM* no se trate de una aberración, se procede a agregar el cromosoma *CM* a la población *P*.

Línea 3: Finalmente se procede a retornar la población actualizada *P*.

1.10 Calibración del porcentaje de casamiento

A continuación se procede a realizar la calibración del valor del porcentaje de la población que será sometida al operador de casamiento. Esto se realizará variando el valor del porcentaje de casamiento desde el valor de 5% hasta 95%, con un incremento de 5% en cada una de las iteraciones. Para cada uno de estos valores de realizarán 40 pruebas de manera que se pueda observar el comportamiento del fitness promedio obtenido por el pre procesado GRASP y el fitness obtenido por el algoritmo genético. Asimismo se podrá observar el porcentaje de casamientos exitosos realizados para cada una de estas pruebas.

A continuación se muestra la tabla resumen con los resultados. Para mayor detalle revisar el anexo 4.

Tabla 8: Calibración del porcentaje de casamiento

Porcentaje mutación	Promedio GRASP	Promedio Genético	Porcentaje Mejora	Promedio Casamiento Exitoso	Mínimo Casamiento Exitoso	Máximo Casamiento Exitoso	Desviación Estándar
5%	8.50	8.94	5.23%	30.21%	0.00%	89%	5.902871
10%	8.51	8.97	5.44%	27.16%	1.50%	84%	5.892819
15%	8.73	9.02	3.30%	26.83%	0.00%	93%	5.923545
20%	8.50	8.99	5.85%	44.38%	0.00%	100%	5.920983
25%	8.73	9.00	3.03%	49.11%	7.99%	100%	5.923358
30%	8.49	8.99	5.86%	52.58%	9.99%	100%	5.906973
35%	8.73	8.99	2.91%	54.99%	12.49%	100%	5.914772
40%	8.73	9.01	3.19%	55.83%	22.07%	100%	5.923209
45%	8.73	8.97	2.73%	59.57%	11.98%	94%	5.892894
50%	8.74	9.00	2.97%	59.95%	17.37%	95%	5.935131
55%	8.73	8.96	2.70%	61.88%	16.07%	96%	5.897602
60%	8.73	8.97	2.83%	61.14%	19.02%	96%	5.913960
65%	8.50	8.93	5.10%	68.76%	38.43%	96%	5.857052
70%	8.74	8.92	2.02%	67.22%	21.00%	96%	5.892167
75%	8.73	8.90	1.88%	71.25%	29.33%	95%	5.885871
80%	8.50	8.87	4.34%	70.77%	29.23%	95%	5.864944
85%	8.74	8.89	1.77%	72.28%	31.75%	96%	5.866839
90%	8.49	8.86	4.33%	70.12%	31.32%	95%	5.854994
95%	8.74	8.82	0.92%	77.21%	29.04%	98%	5.862693

Los datos mostrados en la tabla 12 muestran como se ha ido comportando el algoritmo para los distintos valores del porcentaje de la población a casar. El valor más relevante es el porcentaje de mejora, este indica cuanto progresa el algoritmo genético frente a los resultados del pre procesado del algoritmo GRASP. Es necesario mencionar que el porcentaje de mejora tiene relación directa con el juego de datos de pruebas con el que se realiza la calibración, la cual nos da una idea de cómo se comporta la mejora, sin ser exclusiva.

A continuación se muestra un gráfico con la variación del porcentaje de mejora obtenido.

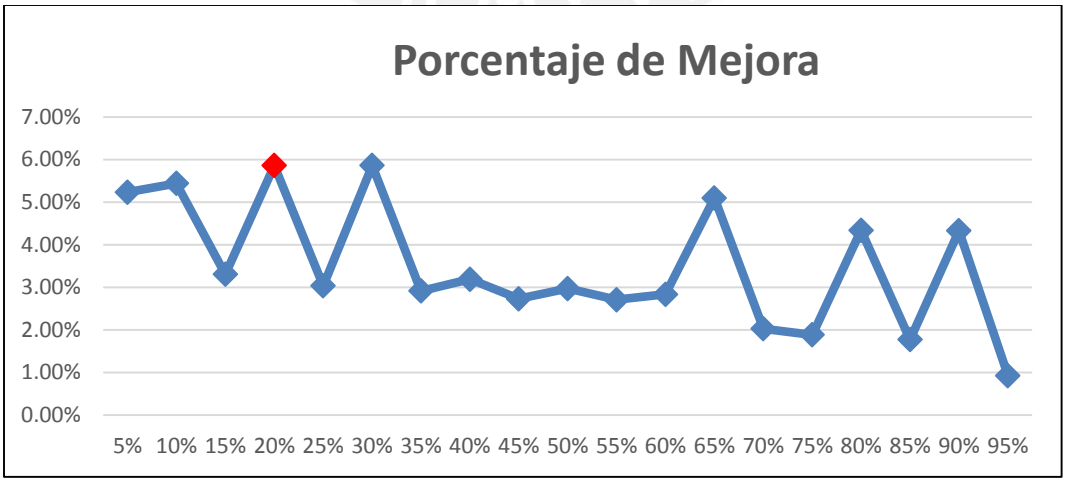


Imagen 26: Gráfico de porcentaje de mejora según el porcentaje de casamiento. Imagen de autoría propia.

Se observa que el porcentaje de casamiento frente al número total de la población que brinda mejores resultados es de 20%. Este porcentaje también tiene un porcentaje de casamientos exitosos alto, de 44.38%, lo cual reafirma la elección de este porcentaje.

1.11 Calibración del porcentaje de mutación

De manera similar con la calibración del porcentaje de casamiento, se procederá a calibrar el porcentaje de individuos que van a ser sometidos al operador de mutación dentro de la población.

Esto se realizará variando el valor del porcentaje de mutación desde el valor de 4% hasta 15%, con un incremento de 1% en cada una de las iteraciones. Para cada uno de estos valores de realizarán 40 pruebas de manera que se pueda observar el comportamiento del fitness promedio obtenido por el pre procesado GRASP y el fitness obtenido por el algoritmo genético.

A continuación se muestra la tabla resumen con los resultados. Para mayor detalle revisar el anexo 5.

Tabla 9: Calibración de porcentaje de mutación

Porcentaje mutación	Promedio GRASP	Promedio Genético	Porcentaje Mejora	Promedio Mutación Exitoso	Mínimo Mutación Exitoso	Máximo Mutación Exitoso	Desviación Estándar
4%	8.74	9.03	3.38%	80.82%	6.50%	99.95%	5.930418
5%	8.71	9.00	3.29%	81.01%	6.47%	99.99%	5.928302
6%	8.74	9.02	3.18%	82.07%	6.01%	99.95%	5.938447
7%	8.74	9.02	3.25%	83.20%	54.63%	99.95%	5.915205
8%	8.73	8.99	3.03%	82.35%	6.46%	99.97%	5.924798
9%	8.73	9.03	3.36%	84.15%	55.66%	99.98%	5.927331
10%	8.73	8.99	2.99%	80.91%	6.25%	99.97%	5.896416
11%	8.49	9.00	6.01%	83.30%	6.66%	99.94%	5.925964
12%	8.71	9.01	3.35%	82.99%	5.90%	99.96%	5.930588
13%	8.49	9.03	6.26%	82.05%	6.36%	99.99%	5.929865
14%	8.73	9.01	3.22%	83.50%	53.54%	99.99%	5.929322
15%	8.73	9.01	3.18%	81.45%	6.89%	99.97%	5.932564

Los datos mostrados en la tabla 9 muestran como se ha ido comportando el algoritmo para los distintos valores del porcentaje de la población a mutar. El valor más relevante es el porcentaje de mejora.

A continuación se muestra un gráfico con la variación del porcentaje de mejora obtenido.

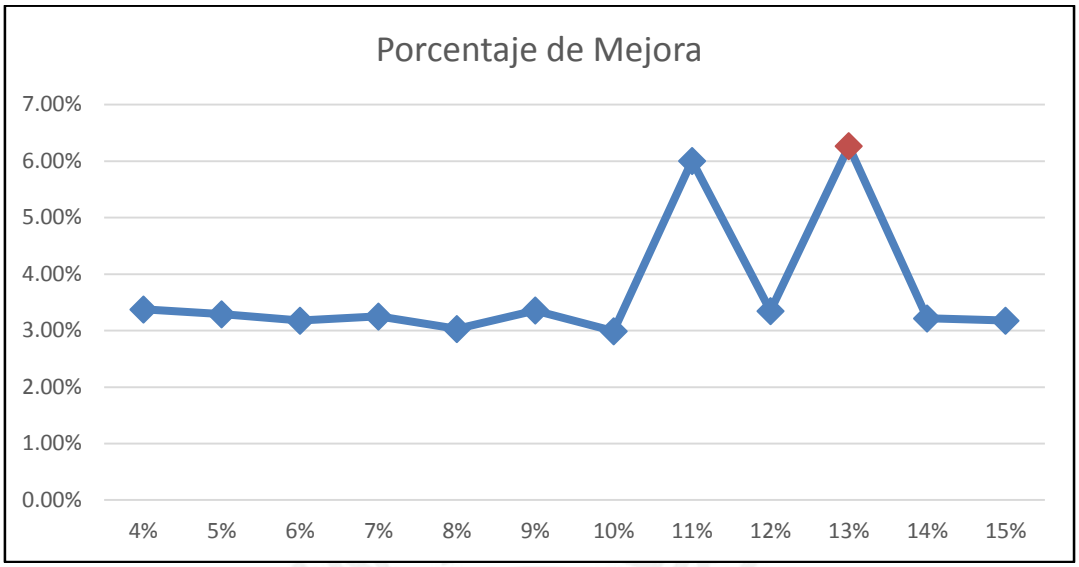


Imagen 27: Gráfico de porcentaje de mejora según el porcentaje de mutación. Imagen de autoría propia.

Se observa que el porcentaje de mutación frente al número total de población que brinda mejores resultados es el de 13%. Este porcentaje también tiene un porcentaje de mutaciones exitosos alto, de 82.05%, lo cual reafirma la elección de este porcentaje.

CAPÍTULO 6: INTERFAZ GRÁFICA

1 Ventanas de la herramienta

A continuación se mostrarán cada uno de los paneles que conformarán la aplicación que contendrá a los dos algoritmos, genético y GRASP, permitiendo la carga de datos de entrada, ejecutar los algoritmos y obtener los resultados

1.1 Panel carga de datos generados

Este panel tendrá como única función la selección del archivo que contendrá los datos de prueba para la simulación, este archivo es del tipo CSV y contendrá la información de los tipos de aviones y los vuelos separados en dos bloques. El primer bloque contendrá los siguientes datos para cada tipo de avión:

- Nombre del tipo de avión
- Cantidad de asientos del tipo de avión
- Máxima distancia que puede recorrer
- Aviones disponibles.

Los cuales deben aparecer separados por el carácter “,”. Cada línea tendrá los datos de un tipo de avión. Luego se tiene el carácter “--”, que indicará el cambio de bloque, luego del cual habrá un cambio de línea. Por último se tiene el bloque de la lista de vuelos. Para cada vuelo se listarán los siguientes datos:

- Cantidad de pasajeros del vuelo
- Ciudad del aeropuerto de salida
- Ciudad del aeropuerto de llegada
- Hora de salida
- Hora de llegada
- Distancia recorrida en el vuelo

Los que también estarán separados por el carácter “,” .Cada línea tendrá los datos de un vuelo. Una vez seleccionado el archivo se procederá a cargar la información del mismo en la memoria de la aplicación. Además luego de la carga de datos se puede usar el botón “Mostrar datos cargados” para revisar en una tabla los que han sido leídos y guardados en la memoria de la aplicación.

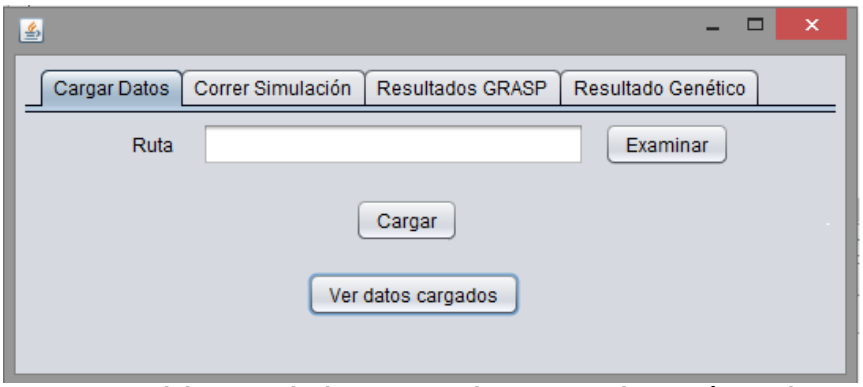


Imagen 28: Panel de carga de datos generados. Imagen de autoría propia.

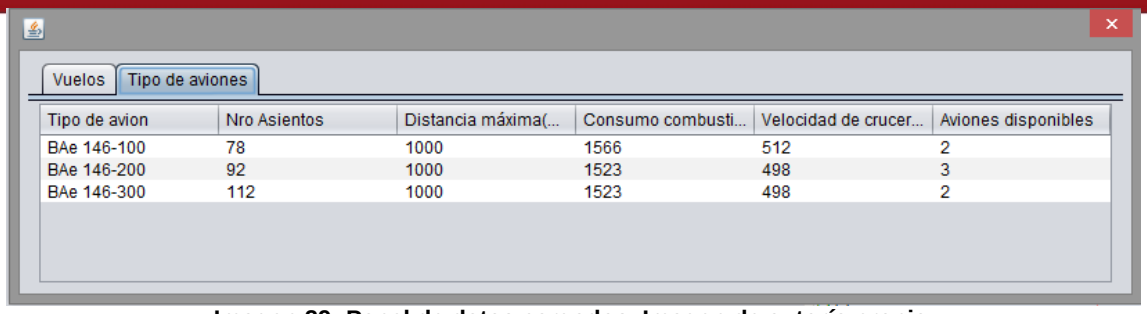


Imagen 29: Panel de datos cargados. Imagen de autoría propia.

1.2 Panel de simulación de asignación de flotas en aerolíneas

Este panel nos muestra algunos parámetros configurables de los algoritmos a ejecutar los cuales son:

- Algoritmo GRASP
  - Alfa
- Algoritmo genético
  - Porcentaje de casamiento
  - Porcentaje de mutación

Cada uno de estos valores mostrará un valor por defecto, el cual ha sido hallado mediante la calibración de cada una de las variables mostradas. La modificación de cada una de estas variables puede afectar negativamente o positivamente al resultado de la simulación.

Por último se presenta un botón que permitirá ejecutar la simulación.

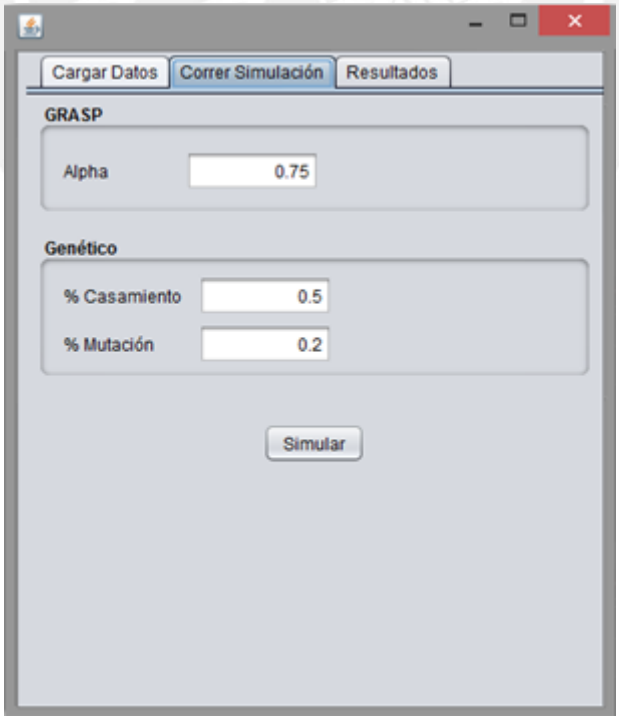


Imagen 30: Panel de ejecución de simulación. Imagen de autoría propia.

1.3 Panel de visualización de resultados de simulación

En este último panel se mostrarán los resultados de la asignación, debido a que cada algoritmo produce un resultado se han puesto dos pestañas, una para mostrar el resultado de cada algoritmo. Este panel contiene una tabla en la cual se mostrarán los datos mínimos necesarios para mostrar la solución, lo cuales son:

- Código de vuelo: Código asignado al vuelo durante la simulación.
- Nombre de vuelo : Código de vuelo asignado por la aerolínea
- Código del tipo de avión: Código asignado al tipo de avión durante la simulación.
- Nombre de tipo de avión: Modelo del tipo de avión.
- Diferencia de asientos: Cantidad de asientos resultante de la asignación.

Aparte de estos datos se mostrará el valor de la función objetivo para cada resultado. También se permitirá exportar los resultados en un archivo CSV, cuyo formato será definido en el siguiente capítulo, para su uso en la experimentación numérica.

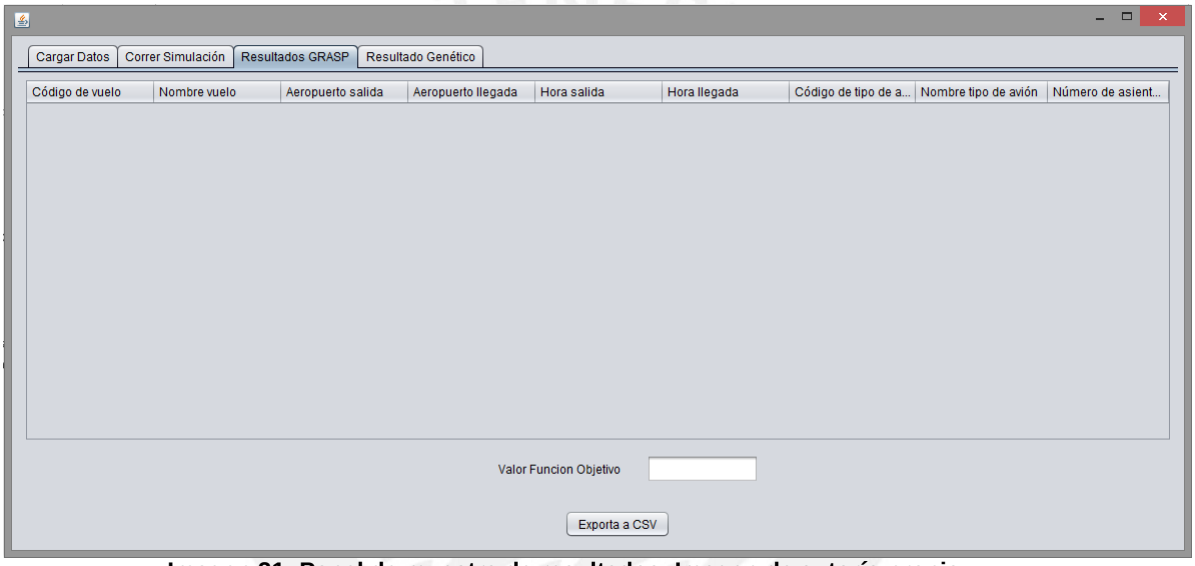


Imagen 31: Panel de muestra de resultados. Imagen de autoría propia.

2 Herramienta principal

La herramienta estará desarrollada en el lenguaje JAVA y será una aplicación de escritorio. A continuación se procederá a detallar el funcionamiento y el proceso de simulación de la herramienta.

2.1 Funcionamiento de aplicación principal

La herramienta tiene tres módulos principales. El primer módulo es de recolección de datos, en el cual se recibirá como entrada un archivo de datos separados por comas (CSV), a partir del cual se recibirán las características de los tipos de aeronaves, las características de los vuelos y la lista de vuelos. Esta información será guardada en las estructuras definidas en el capítulo 3 para su posterior uso.

El segundo módulo es el de simulación donde se recibirán las estructuras preparadas por el módulo de recolección de datos. El usuario podrá configurar tanto el alpha del algoritmo GRASP y los porcentajes de casamiento y mutación del algoritmo genético. Luego este módulo ejecutara secuencialmente cada uno de los algoritmos, de manera

que cada algoritmo aproveche al máximo el poder de procesamiento de la PC donde se ejecuta la aplicación. Luego de la ejecución de cada algoritmo se guardará en memoria los resultados de cada uno.

El último módulo presentará los resultados de la simulación de cada algoritmo en una tabla listando los vuelos con su respectivo tipo de flota asignado. Estos resultados podrán ser exportados e un CSV para su posterior uso en la experimentación numérica.

2.2 Simulación de la herramienta

El proceso de simulación de la herramienta, encargado de ejecutar los algoritmos GRASP y genético, hace uso de múltiples hilos de ejecución de manera que se cada algoritmo se pueda ejecutar por lo menos dos veces en simultaneo. De esta manera se obtiene el mejor resultado de entre las ejecuciones simultáneas de cada algoritmo. El número de hilos de ejecución a usar depende directamente del número de núcleos de procesamiento que posee el computador donde se ejecuta el problema.

Debido a que existe la probabilidad que el algoritmo no pueda encontrar una solución buena en el tiempo establecida, el modulo fuerza la parada de ejecución del algoritmo devolviendo el ultimo resultado obtenido. Este procedimiento será utilizado en ambos algoritmos.

Por último, los resultados obtenidos del algoritmo GRASP son almacenados en un conjunto de soluciones y usados para la ejecución del algoritmo genético, facilitando la comparar la bondad de la solución obtenida del algoritmo GRASP con la mejora realizada sobre la misma solución aplicada por el algoritmo genético.

CAPÍTULO 7: MÉTODOS ESTADÍSTICOS Y RESULTADOS

1 Recolección de datos

En el proceso de generación de datos se usó como base data real, esta data consiste en la composición de la flota perteneciente a aerolíneas peruanas. Entre las aerolíneas utilizadas para la recolección de datos tenemos: Avianca, Peruvian Airlines, LC Perú y Star Perú.

La composición de la flota de cada aerolínea se obtuvo mediante la consulta a la página web de CH-AVIATIONS, una proveedora de información para aerolíneas. En la imagen 32 se muestran los tipos de flota actualmente activos en Avianca. Esta lista luego paso a ser filtrada para obtener las aeronaves utilizadas en territorio nacional.

Aircraft Type	Active
A318-100	10
A319-100	27
A319-100 (sl)	6
A320-100/-200	52
A320-200 (sl)	6
A321-100/-200	3
A321-200 (sl)	5
A330-200	12
ATR 42-300/-320/-400	4
ATR 72-200/-500/-600	12
B787-8	
EMB-190	12
Total	149

Imagen 32: Lista de aviones de Avianca. Imagen recuperada de [CH AVIATIONS, 2014]

Con respecto a la lista de vuelos se hizo un levantamiento de datos usando como fuente a la página de consulta correspondiente de las mismas aerolíneas y de google.com la cual ofrece información sobre las horas de salida de los vuelos, así como su duración promedio. En el levantamiento de datos de los vuelos se recabaron los vuelos para todos los destinos de cada aerolínea durante una semana con el objetivo de obtener varios conjuntos de datos de prueba iniciales.

Respecto a la obtención de los datos sobre las características propias de cada tipo de aeronave se recurrió a las páginas web de los fabricantes, las cuales muestran información lo suficientemente detallada como se muestra en la imagen 33.

Go to key figures | Metric | Imperial

Dimensions	
Overall length	33.84 m
Cabin length	23.78 m
Fuselage width	3.95 m
Max cabin width	3.70 m
Wing span (geometric)	35.80 m
Height	11.76 m
Track	7.59 m
Wheelbase	11.04 m

Capacity		
Pax	Typical seating	124 (2-class)
	Max	156
Freight	LD3 capacity underfloor	4 LD3-45W + 1 LD3/40-45W
	Max pallet number underfloor	4
	Bulk hold volume	7.20 m³
	Total volume (Bulk loading)	27.70 m³

Performance	
Range	6 850 km with Sharklets
Mmo	M0.82
Max ramp weight	64.4 (75.9) tonnes
Max take-off weight	64.0 (75.5) tonnes
Max landing weight	61 (62.5) tonnes
Max zero fuel weight	57.0 (58.5) tonnes
Max fuel capacity	up to 24 210 (30 190) litres

Imagen 33: Características de un avión A319-100. Imagen recuperada de [AIRBUS, 2014].

Debido a que las aeronaves pueden ser configuradas para poder llevar una cantidad variable de asientos de manera que se puedan adaptar a las necesidades particulares de cada aerolínea, se tuvo que consultar la página web de cada una de estas con el objetivo de tener la configuración de asientos usada en cada tipo de aeronave.

Imagen 34: Características de avión A319-100 perteneciente a la aerolínea Avianca. Imagen recuperada de [Avianca, 2014]

Como se puede observar en la imagen 34, Avianca ha decidido usar la configuración de 120 asientos la cual difiere de la configuración típica de 124 asientos mostrada en la imagen 33.

2 Generación de datos

Para la generación de más datos de prueba se usaron como base 28 conjuntos de datos iniciales obtenidos. Para que los datos generados tengan cierto nivel de realismo se ha decidido no alterar ciertas características de la información original, estas son:

- Características de los tipos de aviones
- Duración del vuelo
- Distancia de los vuelos

Los cambios que se le pueden hacer a la información original pueden ser:

1. Eliminar uno o más aeropuertos, así como los vuelos que partan o que lleguen al aeropuerto.
2. Eliminar uno o más vuelos.

Durante el proceso de generación de data se controlará la presencia de aberraciones en los datos, los cuales serán descartados. De esta manera se podrá simular el comportamiento del algoritmo ante una posible baja en la demanda de servicios de transporte aéreo y también ante una subida en la demanda.

La generación de nuevos datos será manual debido a la complejidad presente en la automatización de esta tarea. Todos los datos generados serán guardados en un archivo de datos separados por coma (CSV). A continuación de definirá el formato del archivo CSV.

El archivo tendrá dos bloques de datos. El primer bloque de datos contiene para cada tipo de avión disponible, los siguientes datos:

- El nombre del tipo de avión
- Cantidad de asientos del tipo de avión
- Máxima distancia que puede recorrer
- Consumo de combustible por hora
- Velocidad de crucero
- Aviones disponibles.

Para cada tipo de avión se escriben estos datos en una línea en el mismo orden de la lista y separados por una coma. A continuación aparece la cadena de caracteres "--", la cual actuará como separador entre los dos bloques. Por último se tiene el bloque la lista de vuelos, para cada vuelo se listaran los siguientes datos:

- El nombre del vuelo
- Ciudad del aeropuerto de salida
- Ciudad del aeropuerto de llegada
- Hora de salida
- Hora de llegada
- Duración
- Distancia recorrida en el vuelo
- Factor de demanda

De manera similar al primer bloque, para cada vuelo se escriben los datos mencionados, en el mismo orden de la lista. A continuación se muestra un ejemplo.

```
BAe 146-100,78,1000,1566,512,2
BAe 146-200,92,1000,1523,498,3
BAe 146-300,112,1000,1523,498,2
--
Star Peru 1112,CUZ,LIM,08:45,10:05,01:20,586,5
Star Peru 1114,CUZ,LIM,09:50,11:10,01:20,586,5
Star Peru 1118,CUZ,LIM,10:55,12:15,01:20,586,5
Star Peru 1182,CUZ,LIM,13:40,15:00,01:20,586,5
```

Imagen 35: ejemplo de contenido del archivo CSV. Imagen de autoría propia.

En la imagen 35 se puede observar que el primer bloque contiene las características para tres tipos diferentes de aviones. Luego se ve la cadena de caracteres "--" que indica el fin del bloque de datos de tipos de aviones. A continuación aparecen los datos de 4 vuelos con origen en Lima y con destino hacia Cuzco.

Este archivo será leído por el módulo de carga de datos generados que pertenece a la aplicación principal.

3 Experimentación numérica

El principal objetivo del proyecto es implementar un algoritmo genético que genere una solución buena al problema presentado. Para que la implementación tenga valor agregado debe ser superior a soluciones anteriormente presentadas. Entre los algoritmo meta-heurísticos de optimización global usados para resolver el problema se tiene al algoritmo GRASP y al algoritmo recocido simulado [SHERALI, BISH, ZHU, 2006]. Ambos algoritmos fuero adaptados para resolver el problema de asignación de flotas en aerolíneas y los resultados no mostraron una superioridad significativa de uno sobre otro [SOSNOWSKA, 2000]. Teniendo en cuenta el límite de tiempo para el desarrollo del proyecto se ha elegido al algoritmo GRASP para la comparación, debido a que tiene un rendimiento similar al algoritmo recocido simulado y posee una menor complejidad en su implementación.

Para probar la superioridad de un algoritmo sobre otro se deberían hacer pruebas con todos los posibles casos que pueda presentar el problema. Debido al límite del tiempo del proyecto esta verificación no se puede realizar, por lo que se usará una muestra de

los casos, usando como base casos reales. La muestra estadística a usar en la experimentación numérica tendrá 40 elementos con el objetivo de tener una cantidad suficiente de datos para la realización de las pruebas planificadas. Cada elemento será el promedio del valor de la función objetivo obtenido en 10 ejecuciones para cada juego de datos. La gran mayoría de los juegos de datos necesarios se obtendrán de planes de vuelo de aerolíneas peruanas, el monto restante se obtendrá mediante la alteración de la data recabada.

Para la elección de la comparación a usar se tomó en cuenta la limitación impuesta en el tiempo de ejecución para ambos algoritmos, la complejidad de implementación del algoritmo genético es mayor a la del algoritmo GRASP ya que el principal objetivo del desarrollo del algoritmo es la maximización del beneficio obtenido por las asignaciones de las flotas. En base a estos criterios se descartaron las comparaciones de tiempo de ejecución y de complejidad de algoritmos y se seleccionó la comparación de la calidad de la solución devuelta por ambos algoritmos. La principal métrica a usar en esta comparación se basa en la desviación que tienen los resultados de la solución óptima, es decir cuan alejados están los resultados devueltos por cada algoritmo de la solución óptima global [SILBERHOLZ, GOLDEN, 2010]. En el caso actual la solución óptima tendrá el mayor valor posible de la función objetivo.

Debido a que la solución óptima para cada caso no está disponible, no se puede medir con exactitud la desviación de la solución propuesta de la óptima. Teniendo en cuenta que la solución óptima tiene el valor máximo de la función objetivo, se puede asumir que a mayor sea el valor de la función objetivo para la solución propuesta, esta estará más cerca a la solución óptima. Debido a que no es posible comparar el resultado para cada posible caso se ha tomado una muestra estadística aleatoria, la cual será sometida a pruebas para medir que juego de resultados es significativamente mayor. La prueba a utilizar se denomina ANOVA, análisis de varianza, la cual fue seleccionada debido a que nos permite comparar varias muestras con varianzas desconocidas.

La prueba ANOVA requiere que cada muestra tenga un comportamiento parecido a la distribución normal y que las muestras tengan una varianza significativamente homogéneas. Dependiendo del número de muestras se pueden aplicar algunas de estas pruebas en la comparación de medias [SPRINTHALL, 2011]:

- Prueba Z
- Prueba t de student

Debido a que el número de elementos es mayor a 30 y que la varianza de la población es desconocida se eligió a la prueba Z en vez de la prueba t de student. Esta prueba Z tiene como requerimiento que las muestras a comparar deben tener un comportamiento normal, además las muestras deben tener una varianza conocida. [BERENSON, LEVINE, KREHBIEL 2006].

Con el objetivo de verificar que las muestras tengan un comportamiento normal, se aplicará la prueba Kolmogorov-Smirnov sobre estas, la cual nos indicará si cumplen con la distribución normal o no. La elección de esta prueba se centra en la experiencia previa en su ejecución y que no se han encontrado desventajas significativas en su uso respecto a otras pruebas de normalidad.

Por último se hace uso de la prueba F de Fisher para verificar la homogeneidad de la varianza de las muestras, según lo requerido por la prueba ANOVA. Esto se hace con el objetivo de verificar la necesidad de realizar la prueba de igualdad entre ambas muestras. En caso las varianzas sean homogéneas se deberá verifica primero que las muestras no sean estadísticamente iguales.

Como último paso se procede a realizar la prueba Z para comparar las muestras y obtener cual es mayor, permitiendo saber que algoritmo entrega mejores resultados.

A continuación se explicará cada prueba, las hipótesis a usar en cada una y una explicación más detallada

3.1 Prueba de Kolmogorov-Smirnov

Como ya se mencionó anteriormente el objetivo de esta prueba es verificar que ambas muestras, la del algoritmo GRASP y la del algoritmo genético, tengan una distribución normal. Por ello se plantean las siguientes hipótesis:

$$H0: La\ muestra\ sigue\ una\ distribución\ normal$$

$$H1: La\ muestra\ no\ sigue\ una\ distribución\ normal$$

Las hipótesis mencionadas siguen el esquema para plantear una prueba de dos colas ya que para la hipótesis nula se presenta el caso de igualdad, mientras que en el caso de la hipótesis alternativa se presenta el caso de una desigualdad. Además cada muestra debe tener por lo menos 30 elementos con el objetivo de poder utilizar la prueba Z.

Para las pruebas realizadas se utilizaron 40 elementos, donde cada uno era el máximo valor entre 10 ejecuciones de cada algoritmo para un conjunto de datos. En el caso del algoritmo GRASP se obtuvieron los resultados que se muestran en la imagen 36.

Resultados Grasp	
Valor del estimador	0.101
Grados de libertad	40
Significancia	0.05
Valor crítico	0.21012
Resultado	Estimador fuera de la Zona crítica.
	Se ACEPTA la hipótesis nula

Imagen 36: Resultados de la prueba Kolmogorov-Smirnov para el algoritmo GRASP. Imagen de autoría propia.

Como se puede observar, el valor del estimador obtenido es de 0.101, el cual es menor al valor crítico 0.2101, por lo que el valor estimado se encuentra fuera del rango de la zona crítica, aceptando así la hipótesis nula y concluyendo que este modelo sigue una distribución normal.

De manera análoga para el algoritmo genético se obtuvieron los siguientes resultados:

Resultados Genético	
Valor del estimador	0.102
Grados de libertad	40
Significancia	0.05
Valor crítico	0.21012
Resultado	Estimador fuera de la Zona crítica.
	Se ACEPTA la hipótesis nula

Imagen 37: Resultados de la prueba Kolmogorov-Smirnov para el algoritmo genético. Imagen de autoría propia.

En este caso el valor del estimador también es menor al valor crítico, por lo que podemos aceptar la hipótesis nula y afirmar que este modelo también sigue una distribución normal.

3.2 Prueba F de Fisher

La presente prueba tiene como propósito determinar si las varianzas de ambas muestras son significativamente homogéneas o significativamente diferentes. Para poder realizar esta prueba se debe cumplir con la condición de que la muestra que va a ser analizada cumpla la distribución normal, razón por la cual se eligió la prueba de Kolmogorov-Smirnov. Para cada una de las muestras dadas, se presentan las siguientes hipótesis.

$$H0: La\ varianza\ de\ los\ resultados\ d\ algoritmo\ GRASP\ es\ igual\ a\ la\ varianza\ de\ los\ resultados\ del\ algoritmo\ genético.$$

$$H1: La\ varianza\ de\ los\ resultados\ d\ algoritmo\ GRASP\ es\ diferente\ a\ la\ varianza\ de\ los\ resultados\ del\ algoritmo\ genético$$

Las cuales también se pueden expresar como:

$$H0: \sigma^2_{GRASP} = \sigma^2_{Genético}$$
$$H1: \sigma^2_{GRASP} \neq \sigma^2_{Genético}$$

Las hipótesis planteadas intentan determinar el comportamiento de las varianzas de ambas muestras. La hipótesis nula plantea que las varianzas de ambas muestras son significativamente homogéneas, mientras que la hipótesis alterna propone que las varianzas de las muestras son significativamente distintas.

	Grasp	Genético
Media	11.4898922	12.34662427
Varianza	3.3624	4.20
Observaciones	40	40
Grados de libertad	39	39
F	1.248314351	
P(F<=f) una cola (derecha)	0.245906386	
Valor crítico para F (una cola)	1.704465067	

Imagen 38: Resultado de la prueba F de Fisher. Imagen de autoría propia.

Luego de realizar los cálculos correspondientes se determina que el valor de F para ambas muestras es de 1.2483 y el valor crítico es de 1.7044. Dado estos valores se puede apreciar que el valor de F obtenido está fuera de la región crítica, razón por la cual se acepta la hipótesis nula, concluyendo que para los modelos presentados se tiene varianzas significativamente homogéneas.

3.3 Prueba Z

Finalmente la última prueba que será aplicada a los modelos presentados, busca determinar cuál de los dos modelos tiene la media más baja. Esta prueba es vital ya que para el problema planteado busca demostrar que los valores obtenidos por el algoritmo Genético son más óptimos que los obtenidos por el algoritmo GRASP. Ello se ve relejado en las medias de los fitness obtenidos en ambas muestras. Como se trata de una minimización, el modelo que contenga el valor de media más bajo, será aquel que tenga los resultados más óptimos.

Para poder determinar lo planteado se realizarán dos pruebas de hipótesis. La primera consiste en diferenciar si las medias de ambos modelos son iguales o distintas. Para ello se presentan las siguientes hipótesis:

H0: La media de los resultados del algoritmo GRASP es igual a la media de los resultados algoritmo genético.

H1: La media de los resultados del algoritmo GRASP es diferente a la media de los resultados algoritmo genético.

Las cuales también se pueden expresar como:

$$H0: \mu_{GRASP} = \mu_{Genético}$$
$$H1: \mu_{GRASP} \neq \mu_{Genético}$$

Las hipótesis planteadas son de igualdad para la hipótesis nula y de desigualdad para la hipótesis alterna. Por ello se utilizará una prueba de dos colas para determinar la aceptación o negación de las hipótesis.

A continuación, en la imagen 39 se muestran los resultados de dicha prueba

	Grasp	Genético
Media	11.4898922	12.34662427
Varianza	3.36239223	4.197322475
Observaciones	40	40
Diferencia Hipotética de medias	0	
z	1.97071	
P(Z ≤ z) dos colas	6.66134E-15	
Valor crítico de z (dos colas)	1.959963985	

Imagen 39: Resultado de la prueba Z para dos colas. Imagen de autoría propia. Imagen de autoría propia.

Una vez realizados los cálculos se obtiene el valor de z, que es 1.9707, el cual contrastamos con el valor crítico de 1.9599. Como se puede apreciar, el valor obtenido cae dentro de la región crítica. Ello nos lleva a concluir que la media obtenida del algoritmo GRASP es distinta a la media del algoritmo genético.

Luego de saber que las medias de las muestras dadas son distintas, es necesario determinar cuál de las dos es menor a la otra. Para ello se plantean las siguientes hipótesis:

$$H0: La\ media\ de\ los\ resultados\ del\ algoritmo\ GRASP\ es\ igual\ a\ la\ media\ de\ los\ resultados\ del\ algoritmo\ genético.$$

$$H1: La\ media\ de\ los\ resultados\ del\ algoritmo\ GRASP\ es\ diferente\ a\ la\ media\ de\ los\ resultados\ del\ algoritmo\ genético.$$

Las cuales también se pueden expresar como:

$$H0: \mu_{GRASP} > \mu_{Genético}$$
$$H1: \mu_{GRASP} < \mu_{Genético}$$

Por el tipo de hipótesis presentadas, donde la nula indica que la media del algoritmo GRASP es menor a la media del algoritmo Genético, y la alterna que la media del algoritmo GRASP es mayor a la del Genético, se procede a realizar la prueba de una cola. El resultado esperado de la prueba es representado por la hipótesis nula, donde la media del algoritmo genético es mayor a la media del algoritmo GRASP. A continuación se muestran los resultados en la imagen 40.

	Grasp	Genético
Media	11.4898922	12.34662427
Varianza	3.362	4.197
Observaciones	40	40
Diferencia Hipotética de medias	0	
z	1.97071	
P(Z ≤ z) una cola	3.33067E-15	
Valor crítico de z (una cola)	1.645	

Imagen 40: Resultado de la prueba Z para una cola. Imagen de autoría propia.

Vemos que el valor de z es de 1.9707, mientras que el valor crítico para una cola es de 1.645. Esto demuestra que el valor de z cae dentro de la región crítica, lo cual nos lleva a rechazar la hipótesis nula y aceptar la hipótesis alterna. Eso quiere decir, que concluye que la media obtenida por la muestra del algoritmo genético es mayor a la media obtenida por el algoritmo GRASP. Como se trata de una maximización, se puede concluir que el algoritmo Genético permite obtener resultados más óptimos que un algoritmo GRASP.

CAPÍTULO 8: CONCLUSIONES Y TRABAJOS FUTUROS

1 Resultados Finales

1.1 Observaciones

Cabe señalar que en el presente proyecto se han restringido ciertos aspectos de la aplicación desarrollada como ha sido mencionado en el alcance debido a que el proyecto está orientado al desarrollo y adaptación de un algoritmo genético más que a un sistema o aplicativo en particular.

El manejo de la información de aviones y de los vuelos se está simplificando en la carga del archivo de modo que su ingreso sea más eficiente y consistente al momento de manejar esta información.

Por otro lado, es importante mencionar que la optimización del beneficio toma en cuenta solamente el consumo de combustible, como representante de los costos operacionales, y va priorizando la atención a vuelos de alta demanda ya que para realizar una completa simulación de las variables que intervienen en la decisión de la asignación de un avión a un vuelo se requeriría de información adicional a la cual no se tiene acceso.

Debido a la importancia de encontrar un balance entre la eficiencia del algoritmo y los recursos utilizados, se ha tenido un cuidado especial en el diseño de las estructuras de datos y la forma que estas manejan la información contenida.

Por último, para los casos en los que la cantidad de vuelos era muy baja, 4 a 8, el algoritmo genético no podía mejorar la solución ya obtenida por el GRASP. Para casos con una cantidad de vuelos superiores el algoritmo genético era capaz de entregar una mejor solución que el algoritmo GRASP.

1.2 Conclusiones

A partir de la investigación realizada se puede concluir que el problema de asignación de flotas en aerolíneas es un problema complejo que posee diversas restricciones y consideraciones. Para el caso de estudio, algunas de las restricciones más importantes son la conservación de flujo y las restricciones de temporales debido al tiempo que debe permanecer una aeronave en tierra antes de atender otro vuelo.

Por otro lado, luego de haber realizado la implementación de los algoritmos GRASP y Genético para resolver el problema específico de asignación de flotas y aerolíneas peruanas así como la comparación de resultados en la experimentación numérica, se pueden establecer las siguientes conclusiones:

- Los algoritmos desarrollados efectivamente resuelven el problema planteado en el presente trabajo de modo que permiten obtener asignación que maximiza las ganancias de cada recorrido y minimiza los costos de operación en general, representado por el consumo de combustible.

- Los resultados del algoritmo Genético superan los resultados del algoritmo GRASP para las mismas instancias (número de aviones disponibles y cantidad de vuelos programados). Es decir, que utilizando el algoritmo Genético desarrollado se obtiene una óptima asignación de tipos de flota y mayor utilidad por cada vuelo respetando las restricciones de tiempo y la disponibilidad general de las aeronaves a comparación de las soluciones obtenidas por el algoritmo GRASP.

Para haber establecido esta última conclusión se realizó la experimentación numérica mencionada anteriormente. Previamente a esto, para obtener resultados muchos más precisos y cercanos al óptimo por parte de cada algoritmo, se realizó una calibración de los distintos parámetros que maneja cada metodología. De este modo, para el caso en particular del problema que se pretende resolver, se obtuvo un valor de alfa de 0.74 y número de iteraciones de 5000 para el algoritmo GRASP, mientras que un valor de tasa de casamiento de 0.20 y tasa de mutación de 0.13 para el algoritmo Genético.

De este modo, se obtuvieron los resultados por cada algoritmo desarrollado en la experimentación numérica percibiendo, en promedio, una mejora de resultados del 14% de la función fitness del algoritmo Genético respecto al algoritmo GRASP.

Algoritmo GRASP		Algoritmo Genético propuesto	
(α = 0.74, iteraciones = 5000)	**11.45**	(tCasamiento = 0.20, tMutación = 0.13)	**13.18**

Imagen 41: Comparación algoritmo GRASP vs Genético propuesto. Imagen de autoría propia.

Las ejecuciones fueron realizadas para instancias de 40 conjuntos de vuelos y diferentes modelos de aeronaves. Por lo tanto, la mejora señalada confirma la optimización de resultados esperados en este proyecto.

1.3 Recomendaciones y trabajos futuros

En el presente trabajo, se desarrolló un algoritmo genético para poder obtener la más óptima asignación de tipos de flota a vuelos programados de modo que se obtenga una solución en un tiempo adecuado de acuerdo a los recursos computacionales disponibles.

Entre los trabajos futuros que se recomiendan realizar para el presente algoritmo implementado se encuentran:

- Confrontar el algoritmo desarrollado con otras meta-heurísticas que han tenido éxito en asignación de tareas y en particularmente para este problema como pueden ser Algoritmo Memético, Búsqueda Tabú, entre otras, considerando su desarrollo bajo las mismas condiciones.

- Realizar la fase mejoría del algoritmo GRASP para afinar la asignación obtenida por este.

- Tomar en cuenta una mayor cantidad de variables que puedan afectar la decisión de asignar un avión a un vuelo, como disponibilidad de la tripulación o restricciones en la capacidad de carga con el objetivo de acercar aún más la solución obtenido por los algoritmos a la solución óptima global.

Referencias bibliográficas

Artículos

BÉLANGER, Nicolas; DESAULNIERS, Guy; SOUMIS, François; DESROSIERS, Jacques
2006 "Periodic airline fleet assignment with time windows, spacing constraints, and time dependent revenues". European Journal of Operational Research, Volume 175, Issue 3, pp 1754-1766.

BLUM, Christian; ROLI, Andrea.
2003 "Metaheuristics in combinatorial optimization: Overview and conceptual comparison". ACM Computing Survey. No. 35, pp. 268 – 308.

CHEN, Chiu-Hung; LIU, Tung-Kuan; CHOU, Jyh-Horng; WANG, Chuen-Ching
2012 "Multiobjective Airline Scheduling: An Integrated Approach". 2012, SICE Annual Conference 2012. 2012. pp 1266 - 1270.

COOK, Stephen
1971 "The complexity of theorem proving procedures".3rd annual ACM symposium on Theory of computing.1971, pp 151 – 158

FEO, Thomas; RESENDE, Mauricio
1995 "Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization. 1995. Vol 6. pp 109-134

GROSS, Sven; SCHRÖDER, Alexander
2012 "Handbook of Low Cost Airlines: Strategies, Business Processes and Market Environment". Edición ilustrada. Indianapolis: Addison-Wesley.

GUO, Pengfei; WANG, Xuezhi; HAN, Yingshi
2010 "The Enhanced Genetic Algorithms for the Optimization Design". IEEE 3rd International Conference on Biomedical Engineering and Informatics. 2010, vol.7, pp.2990-2994.

OSMAN, Ibrahim H.; KELLY, James P.
1996 "Meta-Heuristics: Theory and Applications". Kluwer, Boston. Kluwer academic publishers.

RESENDE, Mauricio; RIBEIRO, Celso;
2010 "Greedy Randomized Adaptive Search Procedures. International Series in Operations Research & Management Science. 2010. Vol 146. pp 283-319

SHERALI, Hanif D.; BISH, Ebru K., ZHU, Xiaomei
2006 "Airline fleet assignment concepts, models, and algorithms". European Journal of Operational Research. 2006, vol 172, issue 1, pp 1-30

SILBERHOLZ, John; GOLDEN, Bruce
2010 "Comparison of metaheuristics". Hnadbook of Metaheuristics International Series in Operations Research & Management Science. 2010. Vol 146. Pp 625-640

WANG, Yu; SUN, Hong
2010 "Application of Simulated Annealing Algorithm in Airline Fleet Assignment Problem". International Conference of intelligent System Design and Engineering Application (ISDEA) 2010, vol 1, pp.364-367.

SOSNOWSKA, D.
2000 "Optimization of a simplified fleet assignment problem with metaheuristics: Simulated annealing and GRASP". Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems. pp. 477–488.

Libros

BERENSON, Mark; LEVINE, David; KREHBIEL, Timothy
2006 *"Estadística para Administración"*. Pearson Educación. USA.

BROWNLEE, Jason.
2011 "Clever Algorithms: Nature-Inspired Programming Recipes". Primera Edición. USA: Lulu Enterprises.

DIESTEL, Reinhard
2000 "Graph Theory". Edición electrónica. York: Springer-Verla.

BECK, Kenta; ANDRES, Cynthia
2004 "Extreme Programming Explained: Embrace Change". USA: Addison Wesley Professional.

MITCHEL, Melanie
1999 "An Introduction to Genetic Algorithms". Cambridge, Massachusetts. MIT Press

MOSCATO, P.
1996 "Optimización heurística y Redes Neuronales". España: Editorial Paraninfo.

PMBOK
2009 "A guide to The Project Management Body of Knowledge (PMBOK Guide)". Fourth Edition. PMI Publications, USA.

TUPIA, Manuel
2009 "Fundamentos de Inteligencia Artificial". Tupia Consultores y Auditores S.A.C, PERU.

GEN, Mitsuo; CHENG, Runwei; LIN, Lin
2008 "Network models and optimization: multiobjective genetic algorithm approach". London: Springer-Verlag.

SPRINTHALL, R.C.
2011 "Basic Statistical Analysis". 9th Edition. Pearson Education Group.

Referencias Web

AIRBUS
2014 "A319 - Dimensions & key data"
Consultado: 25 de Septiembre del 2014.
<http://www.airbus.com/aircraftfamilies/passengeraircraft/a320family/a319/specifications/>

ATM
2013 "Europe low cost carrier growth outstrips legacy airline rivals"
Consultado: 18 de Marzo de 2014.
<http://www.airtrafficmanagement.net/2013/05/europe-lcc-growth-outstrips-legacy-rivals/>

AVIANCA
2014 “Nuestra flota – más de 140 años a su servicio”
Consultado: 25 de Septiembre del 2014.
< http://www.avianca.com/es-us/informacion-viaje/antes-vuelo/flota.aspx>

AWERY
2014 “Aircraft Management”
Consultado: 18 de Marzo de 2014.
< http://awery.aero/products/erp-modules/aircraft-management/pdf >

BOEING
2013 “Current Market Outlook 2013 –2032”.
Consultado: 18 de Noviembre de 2013.
<http://www.boeing.com/assets/pdf/commercial/cmo/pdf/Boeing_Current_MarketOutlook_2013.pdf>

CH AVIATION
2014 “Airline Information – LAN Peru”.
Consultado: 25 de Septiembre del 2014.
< http://www.ch-aviation.com/portal/airline/LP#al_profile_tab_fleet >

JAVA
2013 “Caracteristicas y ventajas de Java”.
Consultado: 30 de Mayo del 2014.
< http://www.oracle.com/es/technologies/java/features/index.html >

MICROSOFT OFFICE
2013 “Microsoft Office Products”
Consultado: 30 de Mayo del 2014.
< http://office.microsoft.com/es-es/products >

NETBEANS
2013 “Netbeans IDE - The Smarter and Faster Way to Code”.
Consultado: 30 de Mayo del 2014.
< https://netbeans.org/features/index.html# >

OECD
2008 “The impacts of globalization on international air transport”
Consultado: 18 de Marzo de 2014.
< http://www.oecd.org/greengrowth/greening-transport/41373470.pdf >

RETURETA, Elsa
2010. *“Estadística Inferencial: Planteamiento de Hipótesis en más de dos poblaciones”*. Facultad de Administración, Universidad Veracruzana.
Consultado: 1 de Junio del 2014.
<http://www.slideshare.net/lizgc/fplanteamiento-de-hiptesis-en-mas-de-dos-poblaciones>

SHEOREY DIGITAL
2014 “ARMS® V2”
Consultado: 18 de Marzo de 2014.
< http://www.sds.co.in/prarmsv201.htm>

STUFF
2014 "The rise of China's penny-pinching budget airlines"
Consultado: 18 de Mayo de 2014.
<http://www.stuff.co.nz/travel/news/9943102/The-rise-of-Chinas-penny-pinching-budget-airlines Reglamentos>

UNIVERSIDAD DE VALENCIA
2010 "SPSS: Pruebas no paramétricas"
Consultado: 31 de Mayo del 2014.
< http://www.uv.es/innomide/spss/SPSS/SPSS_0802A.pdf >

Reglamentos

MTC
2000 Ley 27261. Ley de Aeronáutica Civil del Perú.09 de Mayo.

MTC
2002 Resolución Directoral 039-2002-MTC/12. Definiciones y Abreviaturas. 07 de Abril.

MTC
2013 Resolución Directoral 061-2013-MTC/12. Mantenimiento, Mantenimiento Preventivo, Reconstrucción, Alteraciones. 05 de Abril.

Tesis

CALDERON, Imanaida
2002 "*Desarrollo y Prueba de una metodología de Simulación Basada en Algoritmos Genéticos*". Tesis para optar por el Título de Ingeniero de Petróleo. Universidad central de Venezuela, VENEZUELA.
Consultado: 01 de Junio del 2014.

CORTES, Andrés
2013 "*Caracterización del problema del transporte aéreo*". Tesis para optar por el Título de Ingeniero Industrial. Especialidad de Ingeniería Industrial. Universidad Tecnológica De Pereira, COLOMBIA.
Consultado: 20 de Abril del 2014.

GONZÁLEZ, Miguel
2012 *"Programación de itinerarios de líneas aéreas bajo incertidumbre en los tiempos de operación mediante optimización robusta".* Tesis para optar por el grado de Magister en Ciencias e Ingeniería. Pontificia Universidad Católica De Chile, CHILE.
Consultado: 20 de Abril del 2014.

GANOZA, Dante; SOLANO, Úrsula
2004 "*Un algoritmo de búsqueda adaptativa aleatoria y golosa para la resolución del problema de cortes*". Tesis para optar por el Título Profesional de Ingeniero de Sistemas e Informática. Lima: Universidad Nacional Mayor de San Marcos, PÉRU.
Consultado: 01 de Junio del 2014.